Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів гейміфікації та їх програмна реалізація в програмних застосунках підрахунку калорійності та БЖВ продуктів
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗздм-22-1

Ієвлєв І. В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Вечур О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ієвлєву Іллі Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів геймифікації та їх програмна реалізація в програмних застосунках підрахунку калорійності та БЖВ продуктів»
Затверджена наказом по університету від №60Стз від 22.04.2024
2. Термін подання студентом роботи до екзаменаційної комісії 14.06.2024
3. Вихідні дані до роботи методи гейміфікації, розробка додатку для підрахунку БЖВ та калорійності, мова програмування Java, Spring Framework
4. Перелік питань, що потрібно опрацювати в роботі
аналіз та порівняння існуючих методів гейміфікації, вибір підходящих технологій для розробки програмного застосунку підрахунку калорійності та БЖВ продуктів, розробка програмного рішеня для підрахунку калорійності та БЖВ продуктів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	23.01 – 14.02.23	*виконано*
2	Аналіз та моделювання предметної області	17.02 – 28.02.23	*виконано*
3	Програмна реалізація застосунку	25.02 – 01.04.23	*виконано*
4	Аналіз результатів досліджень	22.04 – 10.05.24	*виконано*
5	Написання та оформлення статті та тез доповіді	11.05 – 30.05.24	*виконано*
6	Підготовка пояснювальної записки	11.05 – 10.06.24	*виконано*
7	Підготовка презентації та доповіді	10.06 – 14.06.24	*виконано*
8	Нормоконтроль, перевірка на плагіат	15.06 – 20.06.24	*виконано*
9	Рецензування	20.06 – 21.06.24	*виконано*
10	Занесення диплома в електронний архів	22.06.2024	*виконано*
11	Попередній захист	21.06.2024	*виконано*
12	Допуск до захисту у зав. кафедри	22.06.2024	*виконано*

Дата видачі завдання 20 січня 2023р.

Студент (ка) _____ Ієвлєв І.В.
(підпис)

Керівник роботи _____ доц. Вечур О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 51 с., 5 рис., 1 табл., 9 джерел.

ГЕЙМІФІКАЦІЯ, ПІДРАХУНОК КАЛОРІЙНОСТІ, JAVA, SPRING

Об'єктом дослідження є методи гейміфікації та їх програмна реалізація в програмних застосунках підрахунку калорійності та БЖВ продуктів.

Метою роботи є проведення дослідження методів гейміфікації у програмних застосунках підрахунку калорійності та БЖВ продуктів.

У результаті кваліфікаційної роботи було досліджено використання методів геймифікації у розробці застосунків для підрахунку калорійності та БЖВ продуктів. Також запропоновано метод визначення індексу насиченості їжею.

Практична цінність даного дослідження полягає у вливі на просуванні концепції здорового способу життя. Використання запропонованого додатку знижує ризики ожиріння та пов'язаних хвороб за рахунок формуваня здорових харчових звичок у користувачів.

GAMIFICATION, CALORIE CALCULATION, JAVA, SPRING

The object of research is gamification methods and their software implementation in software applications for calculating calorie content and PFC of products.

The purpose of the work is to conduct a study of gamification methods in software applications for calculating calorie content and nutritional value of products.

As a result of the qualification work, the use of gamification methods in the development of applications for calculating the caloric content and PFC of products was investigated. A method of determining satiety index is also proposed.

The practical value of this study lies in its influence on the promotion of the concept of a healthy lifestyle. The use of the proposed application reduces the risks of obesity and related diseases due to the formation of healthy eating habits in users.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Ієвлєв Ілля Вячеславович, студент гр. ІПЗздм-22-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів гейміфікації та їх програмна реалізація в програмних застосунках підрахунку калорійності та БЖВ продуктів», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БЖВ – Білки Жири Вуглеводи

СУБД – Система Управління Базами Даних

API – Application programming interface

HQL – Hibernate query language

HTTP – Hypertext transfer protocol

JDBC – Java database connectivity

JPA – Java persistence API

ORM – Object-relational mapping

PCF – Protein, fat, carbohydrates

POM – Project object model

SQL – Structured query language

XML – Extensible markup language

ВСТУП

У сучасному світі додатки для підрахунку БЖВ та калорійності продуктів відіграють визначальну роль у підтримці здорового способу життя та інформуванні користувачів про їхні харчові звички. Гейміфікація цих додатків є необхідним елементом для забезпечення зацікавленості користувачів у постійному використанні застосунку та досягненні їхніх цілей. Інтеграція гейміфікованих елементів, таких як досягнення, нагороди, та інтерактивні завдання, дозволяє забезпечити позитивний досвід користувача та зберегти їхній інтерес протягом тривалого періоду користування.

Актуальність дослідження гейміфікації у додатках для підрахунку БЖВ обумовлена сучасними тенденціями у сфері здорового способу життя та технологічного розвитку. Зростаючий інтерес до підтримки оптимальної фізичної форми та контролю над раціоном харчування стимулює розробку нових інструментів, що полегшують цей процес.

Для досягнення мети роботи передбачається аналіз сучасних підходів до гейміфікації у застосунках з підрахунку БЖВ, а також розробка додатку на мові програмування Java з використанням Spring Framework що дозволяє ввести елементи гейміфікації в процес підрахунку БЖВ та калорійності продуктів.

В процесі виконання роботи було вдосконалено методи гейміфікації у застосунках для підрахунку БЖВ та калорійності продуктів шляхом введення нового показнику, а саме індексу насиченості.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

"Гейміфікація - це використання елементів дизайну ігор та ігрових принципів у неігрових контекстах. Вона використовує природні бажання людей до змагання, досягнень, статусу та самовираження, інтегруючи ці мотивації в процеси, які зазвичай не асоціюються з іграми."[1]

Актуальність дослідження гейміфікації у додатках для підрахунку БЖВ обумовлена сучасними тенденціями у сфері здорового способу життя та технологічного розвитку. Зростаючий інтерес до підтримки оптимальної фізичної форми та контролю над раціоном харчування стимулює розробку нових інструментів, що полегшують цей процес.

1.1 Аналіз існуючих методів гейміфікації для підрахунку БЖВ

Існують багато різних методів гейміфікації які створюються для підвищення залученості користувачів. Найбільш популярні методи гейміфікації наведено нижче:

- бейджі та досягнення;
- прогрес бари;
- бали та нагороди;
- соціальні челенджі;
- щоденні завдання;
- відстеження цілей;
- соціальні функції;
- щоденні звіти.

Далі роздивимось кожен з методів гейміфікації.

Досягнення та бейджі є одним з найпоширеніших методів гейміфікації. Користувачі отримують віртуальні нагороди за виконання певних завдань, таких як досягнення цілей по споживанню калорій або виконання фізичних вправ. Ці

нагороди створюють відчуття досягнення і стимулюють користувачів до продовження використання додатка. Переваги:

- підвищують мотивацію та інтерес користувачів;
- стимулюють досягнення нових цілей;
- покращують користувацький досвід.

Прогрес бари візуально показують користувачам їхній прогрес у досягненні певних цілей. Вони надають можливість візуально оцінювати свої досягнення та роблять процес досягнення мети більш прозорим і зрозумілим.

- збільшують прозорість процесу досягнення цілей;
- мотивують користувачів за рахунок візуального відображення прогресу;
- знижують відчуття перевантаженості завдяки розподілу завдань на етапи.

Бали та нагороди передбачають нарахування балів за виконання певних завдань або досягнення цілей. Набрані бали можуть бути обміняні на реальні або віртуальні нагороди, що додатково мотивує користувачів до активного використання додатка.

- створюють систему винагород, що стимулює активність;
- підвищують рівень залученості користувачів;
- можуть бути налаштовані під різні види активності.

Соціальні челенджі дозволяють користувачам змагатися між собою, виконуючи певні завдання або досягаючи спільних цілей. Це створює відчуття спільноти та додає елемент конкуренції, що додатково стимулює до активності.

- підвищують рівень взаємодії між користувачами;
- створюють відчуття спільноти;
- стимулюють досягнення кращих результатів через змагання.

Щоденні завдання надають користувачам можливість виконувати невеликі щоденні завдання, що допомагають досягати їхніх довгострокових цілей. Це робить процес більш структурованим та допомагає формувати корисні звички.

- допомагають формувати корисні звички;
- роблять процес досягнення цілей більш структурованим;

– мотивують користувачів за рахунок регулярних невеликих досягнень.

Цілі та відстеження передбачають встановлення персоналізованих цілей та відстеження прогресу у їх досягненні. Користувачі можуть встановлювати власні цілі щодо споживання калорій, фізичної активності тощо, а додаток відстежує їхній прогрес.

– забезпечують персоналізований підхід до досягнення цілей;
– роблять процес досягнення цілей більш керованим;
– мотивують користувачів через можливість відстеження власного прогресу.

Соціальні функції дозволяють користувачам взаємодіяти один з одним, обмінюватися досягненнями, підтримувати один одного та ділитися своїм прогресом. Це створює відчуття спільноти та підвищує рівень залученості.

– підвищують рівень соціальної взаємодії;
– стимулюють користувачів через підтримку спільноти;
– підвищують рівень залученості користувачів.

Щоденні звіти надають користувачам детальну інформацію про їхню активність та досягнення за день. Це допомагає краще розуміти свій прогрес та коригувати поведінку для досягнення цілей.

– підвищують обізнаність користувачів про власний прогрес;
– допомагають коригувати поведінку для досягнення цілей;
– забезпечують регулярний зворотний зв'язок.

1.2 Необхідність впровадження індексу насиченості

У сучасному світі, де ожиріння та надмірна вага стали глобальними проблемами, ефективне управління харчуванням набуває критичне значення. Одним із ключових аспектів успішного зниження ваги є вибір продуктів, які забезпечують тривале відчуття насичення. Розглянемо, чому це так важливо і як такий підхід сприяє досягненню цілей зниження ваги.

Вживання їжі, яка довго насичує, допомагає краще контролювати апетит та запобігати переїданню. Коли людина споживає їжу, що забезпечує тривале відчуття ситості, зменшується ймовірність частих перекушуваннь та надмірного споживання калорій. Це особливо важливо, оскільки переїдання є однією із головних причин набору ваги. Продукти з високим вмістом білка та клітковини, такі як нежирне м'ясо, риба, яйця, бобові та овочі, ефективно збільшують почуття ситості.

Їжа, яка довго насичує, допомагає стабілізувати рівень цукру на крові. Швидкі вуглеводи, що містяться в солодощах і випічці, викликають різкий стрибок цукру в крові, за яким слідує його різке падіння, що призводить до відчуття голоду та втоми. На відміну від цього, продукти з низьким глікемічним індексом, такі як цільнозернові, овочі та бобові, забезпечують більш рівномірне та тривале вивільнення енергії, що допомагає уникати різких коливань рівня цукру та, отже, голоду. "Дослідження показують, що споживання їжі з низьким глікемічним індексом сприяє більш стабільному рівню глюкози в крові, що може зменшити відчуття голоду та допомогти в підтримці енергетичного балансу, необхідного для ефективного регулювання ваги тіла."[2]

Вибір продуктів, що забезпечують тривале насичення допомагає зменшити загальну калорійність раціону без відчуття голоду. Це можливо завдяки тому, що такі продукти дозволяють з'їдати менше їжі, зберігаючи при цьому почуття ситості на тривалий час. Таким чином, можна створити дефіцит калорій, необхідний для зниження ваги, не відчуваючи дискомфорту і не вдаючись до жорстких дієтичних обмежень.

Продукти, багаті на білок, не тільки сприяють тривалому відчуттю ситості, але й підтримують метаболізм і м'язову масу. Білок вимагає більше енергії для перетравлення та засвоєння в порівнянні з вуглеводами та жирами, що збільшує загальну витрату калорій. Крім того, збереження м'язової маси під час зниження ваги важливо для підтримки високого рівня метаболізму та запобігання зниженню рівня енергії.

Зниження ваги вимагає як короткострокових змін у харчуванні, так й стійких, довгострокових звичок. Вибір продуктів, що забезпечують тривале насичення, сприяє формуванню здорових харчових навичок, які допомагають підтримувати вагу після досягнення мети. Поступове включення таких продуктів до щоденного раціону робить процес зниження ваги більш керованим і менш стресовим, підвищуючи ймовірність довгострокового успіху.

Вживання їжі, яка довго насичує, є ключовим елементом ефективного та сталого зниження ваги. Такі продукти допомагають контролювати апетит, стабілізувати рівень цукру в крові, знизити загальну калорійність раціону, підтримувати метаболізм та м'язову масу, а також сприяють формуванню здорових харчових звичок. Включення цих продуктів до щоденного раціону дозволяє створити сприятливі умови для досягнення та підтримки оптимальної ваги, роблячи процес зниження ваги більш керованим та ефективним.

У світі фітнесу та бодібілдингу, де мета полягає у наборі м'язової маси та збільшенні ваги, правильне харчування відіграє ключову роль. Одним із головних аспектів дієти для набору маси є споживання їжі з низьким коефіцієнтом насичення. Далі розглядається, чому це важливо і як такий підхід допомагає людям, які прагнуть набрати вагу та побудувати м'язову масу. "Для атлетів, які прагнуть набрати масу, важливо споживати їжу з високою калорійністю, але низькою здатністю до насичення. Це дозволяє збільшувати кількість споживаної їжі без надмірного відчуття ситості, що є критичним для досягнення енергетичного балансу та збільшення м'язової маси."[3]

Для успішного набору ваги необхідно підтримувати позитивний енергетичний баланс, що означає споживання більшої кількості калорій, ніж витрачається. Продукти з низьким коефіцієнтом насичення дозволяють з'їдати більше їжі, не відчуваючи сильного почуття ситості. Це спрощує досягнення необхідного рівня калорій, особливо для людей зі швидким метаболізмом або для тих, хто тренується з високою інтенсивністю.

Інтенсивні тренування потребують значної кількості енергії. Для забезпечення достатнього рівня енергії спортсменам потрібно споживати більше

калорій. Їжа з низьким коефіцієнтом насичення дозволяє збільшити обсяг споживаної їжі, що допомагає підтримувати високий рівень енергії та покращує витривалість під час тренувань. Вуглеводи, особливо швидкі вуглеводи, такі як фрукти, рис та хліб, є чудовим джерелом енергії для тренувань високої інтенсивності.

Для набору ваги рекомендується часте харчування – 5-6 разів на день. Їжа з низьким коефіцієнтом насичення допомагає зберегти апетит між їжею, що робить процес харчування більш керованим і комфортним. Це особливо важливо для тих, хто зазнає труднощів зі споживанням великих обсягів їжі за один прийом. Швидкі вуглеводи та легкозасвоювані білки, такі як протеїнові коктейлі, йогурти та горіхові пасти, дозволяють збільшити частоту прийомів їжі без відчуття дискомфорту.

Продукти з низьким коефіцієнтом насичення часто багаті на вуглеводи та жири, які необхідні для підтримки високого рівня енергії та забезпечення організму необхідними макро- та мікроелементами. Це включає вітаміни, мінерали та амінокислоти, які відіграють важливу роль у відновленні м'язів і загальному здоров'ї. Продукти, такі як авокадо, горіхи, насіння та жирна риба, забезпечують необхідні поживні речовини, сприяючи загальному зростанню та відновленню організму.

Для нарощування м'язової маси важливим є не тільки загальна кількість калорій, але й розподіл макроелементів. Продукти з низьким коефіцієнтом насичення, багаті на білки та вуглеводи, допомагають забезпечити необхідні будівельні матеріали для зростання м'язів. Білкові продукти, такі як курка, яловичина, риба та молочні продукти, сприяють синтезу м'язового білка, що критично важливо для відновлення та зростання м'язової тканини.

Процес набору ваги може бути психологічно важким для деяких людей, особливо, якщо вони звикли до обмеження калорій. Продукти з низьким коефіцієнтом насичення дозволяють насолодитися їжею, не викликаючи почуття перенасичення або дискомфорту. Це допомагає підтримувати мотивацію та позитивне ставлення до процесу харчування та набору маси. Продукти, такі як

паста, піца, та десерти, можуть бути включені в раціон для забезпечення необхідної кількості калорій та задоволення від їжі.

Споживання їжі з низьким коефіцієнтом насичення є важливим елементом стратегії для набору ваги та збільшення м'язової маси. Такі продукти дозволяють збільшити загальне калорійне споживання, підтримувати високий рівень енергії, зберігати апетит та часті прийоми їжі, а також забезпечувати організм необхідними макро- та мікроелементами. Оптимізація харчування з акцентом на низький коефіцієнт насичення робить процес набору маси більш ефективним і комфортним, сприяючи досягненню поставлених цілей у сфері фітнесу та бодібілдингу.

1.3 Постановка задачі

Після того як ми провели аналіз проблемної області дослідження, можна сформувати список задач для проведення дослідження.

Отже, для проведення дослідження необхідно вирішити наступні задачі:

– провести аналіз існуючих підходів до гейміфікції додатків для підрахунку БЖВ;

запропонувати метод обчислення індексу насиченості їжею;

– спроектувати базу даних для обраної області;
– спроектувати та реалізувати додаток для гейміфікації підрахунку БЖВ та калорійності продуктів;

- провести аналіз ефективності запропонованого рішення.

2 ПІДХІД В РОЗРОБЦІ ГЕЙМІФІКОВАНОЇ СИСТЕМИ ПІДРАХУНКУ КАЛОРІЙ

У цій роботі пропонується підхід до вирішення завдання щодо гейміфікації у додатку для підрахунку калорійності та БЖВ продуктів. Зокрема вивчаються методи гейміфікації та вплив методів гейміфікації на користувачів застосунку. Також аналізується ефективність використання гейміфікації для процесу підрахунку БЖВ та калорійності продуктів.

Завдання полягає в ідентифікації найбільш поширених проблем, пов'язаних з використанням гейміфікації для підрахунку БЖВ та калорійності продуктів, та визначення шляхів їх вирішення. Крім того, метою є запропонувати певні підходи до використання гейміфікації, які можуть бути єфективними у сфері здорового харчування. Робота має на меті дослідження впливу гейміфікації на мотивацію та успішність користувачів у досягненні своїх цілей по зміні чи збереженню ваги. Досліджуються, які елементи гейміфікації можуть бути найбільш стимулюючими для користувачів та можуть допомагати зберігати цікавість на довгому проміжку часу у підрахунку БЖВ раціону.

Отже, головною метою системи навчання з гейміфікацією є підвищення мотивації та морального стану користувачів які прагнутнуть змінити чи залишити вагу, або досягти інших цілей у здоровому харчуванні. Цільовою аудиторією системи є люди, що прагнуть змінити свій раціон або досягти поставлених цілей за допомогою здорового харчування. Для досягнення цієї мети, розглядається використання різних методів гейміфікації у сфері підрахунку бжв та калорійності продуктів.

2.1 Аналіз додатків для підрахунку БЖВ

У сучасному світі здоровий спосіб життя та правильне харчування стали надзвичайно популярними. У зв'язку з цим, на ринку з'явилося багато мобільних додатків для підрахунку калорій, таких як MyFitnessPal, Yazio, FatSecret та інші. Ці додатки допомагають користувачам контролювати своє харчування, слідкувати за

споживанням макронутрієнтів (білків, жирів, вуглеводів) та калорій. Важливим аспектом, який підвищує залученість користувачів до використання таких додатків, є гейміфікація.

Для того щоб побачити, як додатки використовують методи гейміфікації було створено таблицю (див. табл. 2.1).

Таблиця 2.1 – Використання популярними додатками методів гейміфікації (таблиця виконана самостійно)

Додаток	Досягнення та бейджі	Прогрес бари	Бали та нагороди	Соціальні челенджі	Щоденні завдання	Цілі та відстеження	Соціальні функції	Щоденні звіти
MyFitnessPal	так	так	ні	так	ні	так	так	так
Cronometer	ні	ні	ні	ні	ні	так	ні	так
Lose It!	так	ні	так	так	ні	так	так	ні
Yazio	так	так	ні	так	ні	так	ні	ні
Lifesum	так	ні	ні	так	так	так	ні	ні
MyPlate by Livestrong	так	так	ні	так	ні	так	так	ні
SparkPeople	так	ні	так	так	ні	так	так	ні
FatSecret	ні	ні	ні	ні	ні	так	так	так

MyFitnessPal є одним з найпопулярніших додатків для підрахунку калорій та БЖВ. Він пропонує велику базу даних продуктів харчування, що налічує понад 11 мільйонів найменувань. Користувачі можуть сканувати штрих-коди продуктів для швидкого додавання їх до свого щоденника. Додаток також дозволяє відстежувати фізичну активність, інтегрується з такими пристроями, як Fitbit та Apple Health. Персоналізовані цілі щодо калорій та макроелементів розраховуються на основі індивідуальних параметрів користувача. Гейміфікаційні елементи включають значки за досягнення, щоденні нагадування та соціальні функції для підтримки друзів.

Cronometer спеціалізується на детальному аналізі спожитих поживних речовин, включаючи вітаміни та мінерали. Він дозволяє користувачам відстежувати не лише калорії та макроелементи, а й мікроелементи, що робить його корисним для тих, хто хоче мати повний контроль над своїм харчуванням. Додаток пропонує персоналізовані плани харчування на основі індивідуальних потреб користувача. Візуалізація даних у вигляді графіків та діаграм допомагає краще розуміти свій прогрес. Cronometer інтегрується з іншими фітнес-додатками, що робить його зручним інструментом для підтримки здорового способу життя.

Lose It! фокусується на простоті використання та ефективності. Користувачі можуть легко вводити дані про харчування та фізичну активність, а додаток автоматично розраховує калорії та макроелементи. Унікальною функцією є групові виклики, що підвищують мотивацію та створюють відчуття спільноти. Додаток також пропонує віртуальні нагороди та значки за досягнення певних результатів. Інтеграція з іншими фітнес-додатками та пристроями, такими як Google Fit та Apple Health, дозволяє Lose It! надавати повний огляд здоров'я та фізичної активності користувачів.

Yazio надає користувачам детальні плани харчування та відстеження калорій. Він пропонує понад 1000 рецептів та порад щодо здорового харчування. Унікальна особливість додатку – можливість підніматися рівнями та отримувати досягнення за виконання завдань. Персоналізовані виклики допомагають досягати індивідуальних цілей, таких як скорочення споживання цукру. Щоденні мотиваційні повідомлення підтримують користувачів у зусиллях щодо здорового харчування. Yazio також відстежує фізичну активність, надаючи повний огляд здоров'я.

Lifesum пропонує індивідуальні плани харчування та здорового способу життя, враховуючи вподобання та цілі користувачів. Оцінка харчування кожного прийому їжі за шкалою здоров'я мотивує користувачів вибирати корисні продукти. Додаток інтегрується з іншими фітнес-додатками та пристроями для відстеження фізичної активності та прогресу. Функція планування прийомів їжі допомагає підтримувати

здоровий режим харчування. Щоденні нагадування та мотиваційні повідомлення підвищують залученість користувачів.

MyPlate by Livestrong пропонує користувачам інструменти для відстеження калорій, макро- та мікроелементів. Додаток надає персоналізовані плани харчування та тренувань, що враховують індивідуальні цілі користувачів. Унікальна функція додатку – підтримка соціальної спільноти, де користувачі можуть ділитися успіхами та підтримувати один одного. MyPlate також пропонує рецепти та поради щодо здорового харчування, що допомагає користувачам залишатися мотивованими та зацікавленими у досягненні своїх цілей.

SparkPeople є комплексним додатком для відстеження харчування, фізичної активності та здоров'я. Він пропонує велике співтовариство користувачів, які підтримують один одного у досягненні здорових звичок. Додаток надає індивідуальні плани харчування та тренувань, що допомагають користувачам досягати своїх цілей. Унікальна функція додатку – можливість брати участь у викликах та змаганнях, що підвищує мотивацію. SparkPeople також надає рецепти, поради щодо здорового харчування та тренувань, що робить його ефективним інструментом для підтримки здорового способу життя.

FatSecret пропонує простий та зручний інтерфейс для відстеження калорій та макроелементів. Додаток дозволяє користувачам сканувати штрих-коди продуктів для швидкого додавання їх до щоденника харчування. Персоналізовані цілі щодо калорій та макроелементів допомагають користувачам досягати своїх індивідуальних цілей. Унікальна функція додатку – спільнота користувачів, де можна ділитися успіхами та отримувати підтримку. FatSecret також пропонує рецепти та поради щодо здорового харчування, що підтримує користувачів у досягненні здорових звичок.

2.2 Розрахунок індексу насиченості

Індекс насиченості враховує такі показники їжі, як глікемічний індекс страви, кількість клітковини на 100 грамів, кількість складних та простих углеводів на 100 грамів. Індекс насиченості розраховується за формулою 2.1:

$$F = \frac{A * \frac{B}{C}}{D} \quad (2.1)$$

де F – індекс насиченості,

A– кількість клітковини на 100 грамів,

B – кількість складних вуглеводів,

C – кількість простих вуглеводів,

D – глікемічний індекс.

Глікемічний індекс – показник впливу вуглеводів у продуктах харчування на зміну рівня глюкози в крові щодо впливу чистої глюкози.

Індекс насиченості повинен бути присутній у кожного доданого продукту в додатку.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Опис використовуваних засобів розробки

В першу чергу, якість програмного продукту складається з набору функцій та характеристик програмного забезпечення, які можуть задовольнити потреби та запити зацікавлених сторін[9]. Тому важливо відповідально віднестись до вибору програмного забезпечення що буде використовуватися при створенні проекту.

Для розробки застосунку було використано мову програмування Java з використанням Spring Framework.

У сучасному світі, де веб-розробка швидко розвивається, Java позиціонує себе як універсальний і надійний інструмент для створення веб-рішень.

Мова програмування Java, розроблена в 1995 році компанією Sun Microsystems (зараз Oracle), має широкий спектр функцій, що роблять його придатним для вирішення завдань різної складності.

Об'єктно-орієнтована парадигма Java дозволяє створювати програми, що базуються на концепції об'єктів, що забезпечує модульність, простоту розробки та повторне використання коду[5].

Платформна незалежність Java-коду, яка досягається завдяки компіляції в байт-код, який може виконуватися на будь-якій віртуальній машині Java незалежно від апаратної платформи або операційної системи, робить його універсальним інструментом.

Безпека Java, забезпечена перевіркою типів, керуванням пам'яттю та захистом від винятків, робить його надійним вибором для розробки веб-застосунків.

Висока продуктивність Java-додатків досягається завдяки оптимізації коду віртуальною машиною Java, що забезпечує масштабованість.

Велика і активна спільнота розробників Java надає доступ до великих ресурсів, документації та інструментів.

Це все робить мову програмування Java гарним варінтом для розробки необхідного додатку.

Spring Framework – це один із найбільш затребуваних фреймворків для розробки на Java. Він використовується мільйонами розробників у всьому світі для створення складних, масштабованих та надійних додатків[4]. Ключова перевага Spring Framework полягає в його універсальності та гнучкості, що робить його придатним для різних типів додатків, чи то веб-додатків, мікросервісів чи корпоративних систем. Далі буде розглянуто, чому використання Spring Framework є обґрунтованим вибором для проекту на Java, а також буде показано його плюси та мінуси.

Spring Framework пропонує потужний набір інструментів та бібліотек, які значно спрощують розробку, тестування та розгортання програм. Одним із головних компонентів Spring є Inversion of Control контейнер, який управляє створенням та життєвим циклом об'єктів, дозволяючи розробникам зосередитися на бізнес-логіці, а не на управлінні залежностями. Цей підхід спрощує код і робить його більш модульним та тестованим.

Ще одним важливим аспектом Spring є підтримка Aspect-Oriented Programming, яка дозволяє відокремити бізнес-логіку від системних аспектів, таких як логування, управління транзакціями та безпека. Це сприяє створенню більш чистого та підтримуваного коду. Spring також надає потужні засоби для впровадження крос-зрізів, що покращує структуру та читаність програми.

Одним із найвідоміших модулів Spring є Spring MVC, який використовується для створення веб-додатків. Він забезпечує гнучку та розширювану архітектуру, що дозволяє легко інтегруватися з різними видами уявлень, такими як Thymeleaf або FreeMarker. Завдяки цьому розробники можуть створювати складні веб-застосунки з мінімальними зусиллями. Spring також підтримує RESTful веб-сервіси, що робить його чудовим вибором для розробки сучасних веб-додатків.

Spring Boot, частина екосистеми Spring, є ще один значимий компонент, який значно спрощує процес розробки. Spring Boot пропонує попередньо налаштовані шаблони та вбудований сервер, що дозволяє швидко розгорнути програму без необхідності налаштування складної інфраструктури. Він також підтримує автоматичну конфігурацію, що мінімізує кількість налаштувань, необхідних для

запуску проекту. Це особливо корисно для стартапів та невеликих команд, які прагнуть швидко випустити продукт на ринок.

Незважаючи на численні переваги, Spring Framework має деякі недоліки. Одним із головних мінусів є висока крива навчання. Для розробників-початківців може знадобитися значний час, щоб освоїти всі можливості та концепції фреймворку. Крім того, Spring включає безліч модулів та компонентів, що може створювати зайву складність при розробці невеликих проектів. Ще одним недоліком є обсяг абстракцій та конфігурацій, які можуть уповільнити виконання програми при неправильному використанні.

Однією з головних причин вибору Spring Framework для проекту є його час на ринку та надійність. Spring існує на ринку більше десяти років та зарекомендував себе як перевірене часом рішення. Багато великих компаній і корпорацій використовують Spring для своїх критично важливих додатків, що підтверджує його надійність та продуктивність[6]. Крім того, Spring має величезну спільноту розробників і велику документацію, що полегшує пошук рішення для проблем і питань, що виникають.

Ще однією значною перевагою Spring є його підтримка інтеграції з іншими технологіями та фреймворками. Наприклад, Spring легко інтегрується з Hibernate для роботи з базами даних, Apache Kafka для обробки потоків даних, а також з різними хмарними платформами, такими як Google Cloud і Microsoft Azure. Це робить Spring універсальним інструментом, який можна адаптувати до конкретних потреб проекту.

Важливим аспектом є тестування додатків на основі Spring. Фреймворк підтримує різні інструменти для модульного та інтеграційного тестування, такі як JUnit та Mockito. Це дозволяє розробникам створювати надійні тести та гарантувати якість коду на всіх етапах розробки. Spring також підтримує тестування крос-зрізів та аспектів, що спрощує перевірку коректності роботи різних частин програми.

Масштабованість додатків на основі Spring. Фреймворк дозволяє легко створювати мікросервісні архітектури, де кожна програма є незалежним модулем,

що взаємодіє з іншими через легковажні протоколи, такі як HTTP. Це спрощує розгортання та керування програмами, а також дозволяє масштабувати їх залежно від навантаження.

MySQL - це одна з найбільш широко використовуваних реляційних систем управління базами даних, розроблена компанією MySQL і нині підтримувана Oracle Corporation. Ця СУБД широко застосовується у різних додатках, починаючи з малих веб-сайтів і закінчуючи великими корпоративними системами. MySQL відомий своєю продуктивністю, надійністю та простотою у використанні, що робить його кращим вибором для багатьох проектів. Далі буде розглянуто, чому використання MySQL є оптимальним рішенням для вашого проекту, а також буде обговорено переваги та недоліки реляційних баз даних.

MySQL – це реляційна СУБД, що означає зберігання даних у таблицях, які можуть бути взаємопов'язані. Це дозволяє ефективно керувати великими обсягами даних та виконувати складні запити. MySQL використовує мову структурованих запитів (SQL), стандарт для керування реляційними базами даних, що дозволяє створювати, змінювати та керувати даними.

MySQL також виділяється своєю надійністю та стабільністю. СУБД пропонує засоби для резервного копіювання та відновлення даних, що мінімізує ризик втрати інформації[7]. Важливим аспектом є підтримка різних рівнів ізоляції транзакцій, що забезпечує цілісність даних та запобігає конфліктам при одночасному доступі.

Один з ключових плюсів MySQL полягає у його відкритому вихідному коді. Це дозволяє безкоштовно використовувати базу даних і модифікувати її для задоволення конкретних потреб проекту. Відкритий вихідний код також сприяє активному розвитку та підтримці MySQL спільнотою розробників, що гарантує регулярні оновлення та виправлення, покращуючи продуктивність та безпеку СУБД.

MySQL відрізняється високою масштабованістю. Ця база даних може ефективно працювати як у малих серверах, і великих кластерних системах. Можливість горизонтального та вертикального масштабування дозволяє легко

адаптувати MySQL під зростаючі потреби вашого проекту. Наприклад, шардування даних дозволяє розподілити навантаження на кілька серверів, підвищуючи продуктивність.

Ще однією значною перевагою MySQL є її сумісність з різними операційними системами та платформами. MySQL підтримує Windows, Linux, macOS та інші системи, забезпечуючи гнучкість у виборі інфраструктури для розгортання бази даних. Крім того, MySQL легко інтегрується з різними мовами програмування та фреймворками, такими як Java, PHP, Python та .NET, що спрощує розробку та підтримку програм.

MySQL надає потужні інструменти для забезпечення безпеки даних. СУБД підтримує автентифікацію користувачів, управління правами доступу та шифрування даних, що захищає інформацію від несанкціонованого доступу. Важливим аспектом є також можливість налаштування аудиту та моніторингу активності бази даних, що допомагає виявляти та запобігати потенційним загрозам безпеці.

Незважаючи на численні переваги, MySQL, як і будь-яка реляційна СУБД, має свої обмеження. Одним із основних недоліків реляційних баз даних є їх обмежена гнучкість при роботі з неструктурованими даними. Сучасні програми часто вимагають зберігання та обробки великих обсягів неструктурованої інформації, такої як документи, зображення та мультимедійні файли. У таких випадках бази даних NoSQL можуть бути більш ефективним рішенням.

Ще одним недоліком реляційних баз даних є складність масштабування. Хоча MySQL підтримує різні методи масштабування, такі як шардування та реплікація, налаштування та управління масштабованими кластерами може бути складним завданням. Для деяких проектів, особливо тих, які вимагають горизонтального масштабування на велику кількість вузлів, бази даних NoSQL можуть запропонувати більш прості та ефективні рішення.

Реляційні бази даних, включаючи MySQL, можуть стикатися з проблемами продуктивності при виконанні складних запитів на великих обсягах даних. Незважаючи на оптимізацію та використання індексів, виконання складних join-

операцій та агрегацій може займати значний час та ресурси. У таких випадках потрібне ретельне налаштування та оптимізація бази даних для забезпечення необхідного рівня продуктивності.

Hibernate є одним із найпоширеніших фреймворків для роботи з об'єктно-реляційним відображенням (ORM) в екосистемі Java. Він створений для спрощення взаємодії між об'єктно-орієнтованими додатками та реляційними базами даних. Основна функція Hibernate – автоматичне перетворення даних між об'єктною моделлю програми та реляційною моделлю бази даних. Це дозволяє розробникам працювати з базами даних, використовуючи об'єктно орієнтований підхід, мінімізуючи при цьому кількість необхідного коду SQL.

Hibernate – це ORM-фреймворк, що підтримує об'єктно-реляційне відображення. Він дозволяє автоматично зберігати об'єкти Java у реляційній базі даних та витягувати їх назад. Це досягається за допомогою конфігураційних файлів та анотацій, які дозволяють мапірувати класи Java на таблиці бази даних. Hibernate підтримує безліч різних баз даних, включаючи MySQL, PostgreSQL, Oracle та інші, що робить його універсальним інструментом розробки на Java.

Однією з ключових особливостей Hibernate є підтримка HQL - мови запитів, яка схожа з SQL, але працює з об'єктами, а не з таблицями. Це дозволяє розробникам писати запити, використовуючи об'єктну модель, що спрощує процес розробки та робить код більш читабельним та підтримуваним. HQL також підтримує всі основні операції, такі як вибірка, вставка, оновлення та видалення даних.

Головна перевага Hibernate полягає у його здатності скорочувати обсяг коду, необхідного для взаємодії з базою даних. Замість написання великої кількості SQL-коду для виконання операцій з базою даних, розробники можуть використовувати методи фреймворку, що робить код більш чистим та простим у супроводі. Це особливо важливо у великих проектах, де керування складними SQL-запитами може стати значною проблемою.

Hibernate також забезпечує автоматичне керування транзакціями, що спрощує роботу з базою даних та знижує ймовірність помилок. Фреймворк

підтримує транзакції лише на рівні об'єкта, що дозволяє виконувати операції над кількома об'єктами у межах однієї транзакції, забезпечуючи цілісність даних. Це особливо корисно в корпоративних додатках, де критично важлива надійність та узгодженість даних.

Ще однією значною перевагою Hibernate є кешування. Фреймворк підтримує кешування двох рівнях: перший рівень (L1) – кеш лише на рівні сесії, і другий рівень (L2) – кеш лише на рівні фабрики сесій. Кешування дозволяє зменшити кількість звернень до бази даних, що підвищує продуктивність програми. Завдяки цьому, Hibernate може значно прискорити виконання запитів, що часто повторюються.

Hibernate також надає потужні засоби для валідації даних. За допомогою вбудованих механізмів можна визначити правила валідації, які автоматично застосовуються при збереженні об'єктів у базі даних. Це дозволяє підвищити якість даних та зменшити кількість помилок, пов'язаних із некоректними даними.

Незважаючи на численні переваги, Hibernate має свої недоліки. Одним із основних мінусів є висока крива навчання. Для розробників-початківців може знадобитися значний час, щоб освоїти всі можливості та концепції фреймворку. Hibernate включає безліч функцій і конфігураційних параметрів, що може створювати складність при його вивченні та використанні.

Ще одним недоліком є потенційне зниження продуктивності при неправильному використанні. Некоректне настроювання кешування або неефективні запити можуть призвести до значних витрат ресурсів та уповільнення роботи програми. Для досягнення оптимальної продуктивності потрібне ретельне налаштування та моніторинг роботи фреймворку.

Також варто відзначити, що Hibernate додає рівень абстракції між програмою та базою даних, що може ускладнити налагодження. У деяких випадках, особливо при складних запитах або при виникненні проблем із продуктивністю, може знадобитися глибоке розуміння як Hibernate, так і реляційної бази даних для ефективного вирішення проблем.

Існують різні альтернативи Hibernate для роботи з базами даних Java. Серед них найбільш відомі наступні.

MyBatis – це інший популярний ORM-фреймворк, який пропонує більш гнучкий та низькорівневий підхід порівняно з Hibernate. MyBatis дозволяє розробляти SQL-запити вручну і мапувати їх на об'єкти Java, що дає більше контролю над процесом взаємодії з базою даних.

JDBC – це низькорівневий API для прямої взаємодії з базами даних із використанням SQL-запитів. Хоча JDBC вимагає більше коду та зусиль для керування з'єднаннями та транзакціями, він надає максимальну гнучкість та контроль над взаємодією з базою даних.

Вибір Hibernate як ORM-фреймворку для проекту на Java обумовлений його потужними можливостями та широкими функціональними можливостями. Hibernate надає високий рівень абстракції, що спрощує розробку та підтримку додатків, знижуючи кількість коду та полегшуючи керування транзакціями. Підтримка кешування та валідації даних робить Hibernate ефективним та надійним інструментом для роботи з базами даних.

Крім того, Hibernate є однією з найбільш старих та широко використовуваних реалізацій JPA, що гарантує його сумісність із різними базами даних та інтеграцію з іншими технологіями. Завдяки активній спільноті розробників та великої документації, Hibernate залишається актуальним і постійно розвивається, пропонуючи нові можливості та покращення.

Незважаючи на наявність альтернатив, таких як MyBatis та JDBC, Hibernate пропонує оптимальний баланс між простотою використання та функціональністю. MyBatis може бути кращим для проектів, де потрібний повний контроль над SQL-запитами, а JDBC – для низькорівневої взаємодії з базою даних. Однак для більшості проектів Hibernate забезпечує найбільш зручне та ефективне рішення для роботи з реляційними базами даних, дозволяючи зосередитися на бізнес-логіці програми.

Таким чином, вибір Hibernate для вашого проекту на Java дозволить вам створити надійну, масштабовану та продуктивну програму, яка буде легко підтримувати і розвивати в майбутньому.

Apache Maven є одним із найпоширеніших інструментів для управління проектами та автоматизації складання в екосистемі Java. Він надає розробникам широкі можливості для управління залежностями, складання, тестування та розгортання проектів. Основне завдання Maven – спростити та стандартизувати процес розробки програмного забезпечення, забезпечуючи при цьому високу гнучкість та масштабованість.

Maven використовує XML для опису структури проекту, залежностей, плагінів та цілей складання. Це дозволяє керувати проектами за допомогою єдиного конфігураційного файлу, що значно спрощує процес налаштування та підтримки проекту. Однією з основних переваг Maven є централізоване керування залежностями. Maven використовує репозиторії для зберігання артефактів (бібліотек та плагінів), що дозволяє автоматично завантажувати та оновлювати залежності проекту. Це звільняє розробників від необхідності вручну керувати бібліотеками та їх версіями, мінімізуючи ризик виникнення конфліктів та спрощуючи оновлення залежностей.

Процес складання Maven заснований на фазах і цілях. Фази є етапами життєвого циклу проекту, такі як компіляція, тестування та упаковка, а цілі визначають конкретні завдання, які мають бути виконані на кожному етапі. Це дозволяє легко налаштовувати та розширювати процес складання, додаючи нові цілі та плагіни для виконання специфічних завдань. Плагіни Maven відіграють ключову роль, забезпечуючи підтримку різних інструментів і технологій, таких як компілятори, тестові фреймворки і системи розгортання.

Ще однією значною перевагою Maven є його здатність працювати у багатомодульних проектах. Це особливо корисно для великих корпоративних систем, де один проект може складатися з безлічі модулів, кожен з яких є окремим компонентом. Maven дозволяє керувати такими проектами централізовано,

забезпечуючи узгодженість версій та залежностей між модулями. Це сприяє покращенню структури проекту та спрощує його супровід.

Незважаючи на безліч переваг, у Maven є свої недоліки. Одним з основних мінусів є складність конфігурації і висока крива навчання для розробників-початківців. Файл конфігурації POM (Project Object Model) може ставати досить складним і важко читати у великих проектах. Це вимагає від розробників глибокого розуміння структури та принципів роботи Maven для ефективного використання його можливостей. Також варто відзначити, що Maven може бути повільним при виконанні деяких завдань, особливо при початковому завантаженні залежностей віддалених репозиторіїв. Це може сповільнити процес складання, особливо в проектах з великою кількістю залежностей та складними конфігураціями.

Альтернативами Maven є такі інструменти, як Gradle та Apache Ant. Gradle надає більш сучасний та гнучкий підхід до управління проектами та складання. Він використовує мову програмування Groovy для опису конфігурації, що робить її більш виразною та потужною в порівнянні з XML, що використовується в Maven. Gradle також пропонує покращену продуктивність завдяки інкрементальному складання та кешування завдань. Однак, Gradle має свою криву навчання і потребує певних зусиль для освоєння.

Apache Ant, у свою чергу, є більш низькорівневим інструментом, який надає більшу гнучкість та контроль над процесом збирання. Ант використовує XML для опису завдань збирання, але не має вбудованої системи керування залежностями. Це означає, що розробники повинні вручну керувати залежностями та їх версіями, що може ускладнити процес складання та супроводу проекту. Ант підходить для проектів, що вимагають високого ступеня налаштування та нестандартних процесів складання, але можуть бути надлишковими для більшості стандартних завдань.

Вибір на користь Maven для вашого проекту на Java обґрунтований його здатністю спрощувати та стандартизувати процес розробки, надаючи потужні можливості для керування залежностями, складання та розгортання. Незважаючи на деякі недоліки, такі як складність конфігурації та крива навчання, Maven залишається одним з найбільш популярних інструментів, що широко

використовуються в екосистемі Java. Його переваги, такі як централізоване управління залежностями, підтримка багатомодульних проектів та гнучкість налаштування процесу складання роблять його оптимальним вибором для більшості проектів.

Використання Maven дозволяє розробникам зосередитись на написанні коду, а не на управлінні інфраструктурою проекту. Це особливо важливо у великих командах, де узгодженість та стандартизація процесів розробки є критичними факторами успіху. Вибір Maven для вашого проекту на Java дозволить створити надійний, масштабований і легко супроводжуваний додаток, який буде відповідати високим стандартам індустрії програмного забезпечення.

У подальшому також планується розробити частину додатку, що відповідає за зовнішній вигляд додатку. У процесі розробки візуальної частини потрібно також враховувати естетичну складову. "Основне завдання, що стоїть перед розробниками, дизайнерами та проектувальниками сьогодні – це розробка якісного програмного забезпечення, яке буде популярним та естетично привабливим для користувача."[8]

3.2 Опис архітектури системи

В процесі програмної реалізації проєкту було створено додаток з використанням REST API. Архітектура додатку наведена на рисунку (див. рис. 3.1)

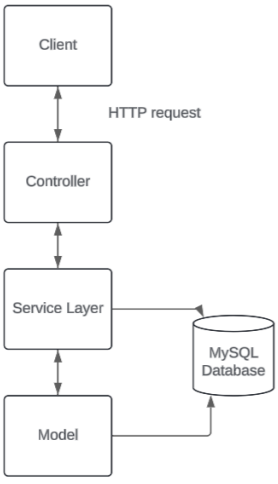

Рисунок 3.1 – Архітектура додатку (за даними [4])

Архітектура системи базується на моделі клієнт-сервер. Додаток розроблений за допомогою Spring Framework з використанням Spring Boot

Розглянемо клас Item.class що відображає кожен продукт що може бути доданий у додаток (див. рис. 3.2).

```java
@Entity
@Getter
@Setter
@Table(name = "items")
public class Item {

    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Min(1)
    private Long id;

    @Column(name = "name")
    @NotNull
    @NotBlank
    private String name;

    @Column(name = "satiety_index")
    @NotNull
    @NotBlank
    private int satietyIndex;

    @Column(name = "calories")
    @NotNull
    @Min(0)
    private int calories;

    @Column(name = "protein")
    @NotNull
    @Min(0)
    private int protein;

    @Column(name = "carbohydrate")
    @NotNull
    @Min(0)
    private int carbohydrate;

    @Column(name = "fat")
    @NotNull
    @Min(0)
    private int fat;

    public Item(Long id, String name,
                int satietyIndex, int calories, int protein,
                int carbohydrate, int fat) {
        this.id = id;
        this.name = name;
        this.satietyIndex = satietyIndex;
        this.calories = calories;
        this.protein = protein;
        this.carbohydrate = carbohydrate;
        this.fat = fat;
    }

    public Item(String name, int satietyIndex, int calories,
                int protein, int carbohydrate, int fat) {
        this.name = name;
        this.satietyIndex = satietyIndex;
        this.calories = calories;
        this.protein = protein;
        this.carbohydrate = carbohydrate;
        this.fat = fat;
    }
```

Рисунок 3.2 – Код Item.class (рисунок створено самостійно)

Клас `Item` у пакеті `com.ievlev.calorie_calculator.model` являє собою сутність для збереження даних про продукти в системі управління калоріями. Нижче наведено детальний опис кожної аннотації та атрибутів класу.

Entity аннотація вказує, що клас `Item` є сутністю JPA (Java Persistence API). Сутність представляє собою об'єкт, який буде відображатися на таблицю бази даних. Завдяки цій аннотації, JPA розуміє, що цей клас потрібно зберігати в базі даних і керувати ним.

Аннотація `@Table` задає ім'я таблиці в базі даних, до якої буде прив'язана сутність. У даному випадку, таблиця називається "items". Ця аннотація допомагає явно вказати ім'я таблиці, яке може відрізнятися від імені класу.

Аннотація `@Id` позначає поле `id` як первинний ключ для цієї сутності. Первинний ключ унікально ідентифікує кожен запис у таблиці.

Аннотація `@Column` вказує, що поле `id` буде відображатися на колонку з ім'ям "id" у таблиці. Подібні аннотації використовуються для прив'язки полів класу до відповідних колонок у таблиці бази даних.

Аннотація `@GeneratedValue` вказує, що значення для поля `id` буде генеруватися автоматично. Стратегія `GenerationType.IDENTITY` означає, що база даних буде автоматично збільшувати значення цього поля при вставці нового рядка (використовується для автогенерації ідентифікаторів).

Аннотація `@Min` задає мінімальне допустиме значення для поля `id`. У даному випадку, `id` повинно бути не менше 1. Це обмеження допомагає гарантувати, що значення ідентифікатора завжди буде позитивним.

Аннотація `@NotNull` вказує, що поле `name` не повинно бути `null`. Це умова застосовується також до інших полів (satietyIndex, calories, protein, carbohydrate, fat). Гарантує, що відповідні дані завжди будуть присутні.

Аннотація `@NotBlank` застосовується до рядкових полів і вказує, що значення не повинно бути порожнім або складатися лише з пробілів. Ця аннотація використовується для поля `name`, щоб гарантувати, що ім'я продукту завжди містить значущі символи.

Аннотація Lombok, яка автоматично генерує методи `getter` для всіх полів класу.

Аннотація Lombok, яка автоматично генерує методи `setter` для всіх полів класу.

Використання Lombok значно скорочує кількість коду, необхідного для написання геттерів і сеттерів, що покращує читабельність і спрощує підтримку коду.

Також в процесі створення проекту було створено клас User.class (див. рис. 3.3).

```java
@Entity
@Getter
@Setter
@Table(name = "users")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    @Min(1)
    private Long id;

    @Column(name = "username")
    @NotNull
    @Size(min = 1, max = 30)
    private String username;

    @Column(name = "password")
    @NotNull
    @Size(min = 1, max = 30)
    private String password;

    @ManyToMany
    @JoinTable(
            name = "users_roles",
            joinColumns = @JoinColumn(name = "user_id"),
            inverseJoinColumns = @JoinColumn(name = "role_id")
    )
    @NotNull
    private List<Role> roles;

    @OneToMany(mappedBy = "user")
    private List<Order> order;


    public User(String username, String password, List<Role> roles) {
        this.username = username;
        this.password = password;
        this.roles = roles;
    }

    public User(String username, String password, List<Role> roles, List<Order> order) {
        this.username = username;
        this.password = password;
        this.roles = roles;
        this.order = order;
    }

    public User() {
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        User user = (User) o;
        return Objects.equals(id, user.id);
    }

    @Override
    public int hashCode() { return Objects.hash(id); }
```

Рисунок 3.3 – Клас User.class (рисунок створено самостійно)

Клас `User` у пакеті `com.ievlev.calorie_calculator.model` представляє сутність для збереження даних про користувачів у системі управління калоріями. Нижче наведено детальний опис аннотацій та атрибутів класу.

Аннотація `@ManyToMany` вказує на зв'язок багато-до-багатьох між сутністю `User` та іншою сутністю. У даному випадку, цей зв'язок встановлено між `User` та `Role`.

Аннотація `@JoinTable` визначає проміжну таблицю для зв'язку багато-до-багатьох. Вона містить атрибути `joinColumns` та `inverseJoinColumns`, які вказують на колонки, що використовуються для зв'язку між таблицями `users` та `roles`.

Аннотація `@OneToMany` вказує на зв'язок один-до-багатьох між сутністю `User` та іншою сутністю. У даному випадку, цей зв'язок встановлено між `User` та `Order`, де один користувач може мати багато замовлень.

Також в процесі створення проекту було створено клас Order.class (див. рис. 3.4).

```
@Entity
@Getter
@Setter
@Table(name = "orders")
public class Order {

    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Min(1)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "user_id")
    @NotNull
    private User user;

    @ManyToMany
    @JoinTable(
            name = "orders_items",
            joinColumns = @JoinColumn(name = "order_id"),
            inverseJoinColumns = @JoinColumn(name = "item_id")
    )
    private List<Item> itemList;

    public Order(Long id, User user, List<Item> itemList) {
        this.id = id;
        this.user = user;
        this.itemList = itemList;
    }

    public Order(User user, List<Item> itemList) {
        this.user = user;
        this.itemList = itemList;
    }

    public Order() {
    }
}
```

Рисунок 3.4 – Клас Order.class (рисунок створено самостійно)

Клас Order у пакеті com.ievlev.calorie_calculator.model представляє сутність для збереження даних про раціон у системі управління калоріями. Нижче наведено детальний опис аннотацій та атрибутів класу.

Аннотація @ManyToOne вказує на зв'язок багато-до-одного між сутністю Order та іншою сутністю. У даному випадку, цей зв'язок встановлено між Order та User.

Аннотація @JoinColumn використовується для вказівки колонки, яка буде використовуватися для зв'язку між таблицями. У даному випадку, вона вказує на колонку user_id у таблиці orders.

Аннотація @ManyToMany вказує на зв'язок багато-до-багатьох між сутністю Order та іншою сутністю. У даному випадку, цей зв'язок встановлено між Order та Item.

Аннотація @JoinTable визначає проміжну таблицю для зв'язку багато-до-багатьох. Вона містить атрибути joinColumns та inverseJoinColumns, які вказують на колонки, що використовуються для зв'язку між таблицями orders та items.

Також в процесі розробки було створено класи сервісів та контролери.

Струкрура файлів у проекті наведена на рисунку 3.5.

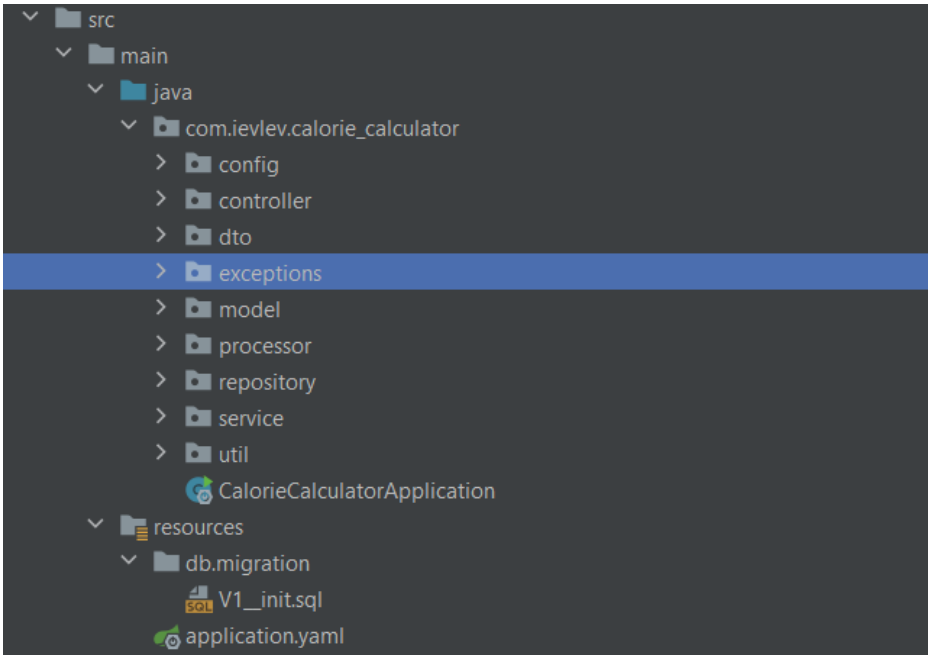

Рисунок 3.5 – Структура файлів (рисунок створено самостійно)

3.3 Опис роботи застосунку

Створений застосунок – це калькулятор БЖВ та калорійності продуктів. Цей додаток створений для допомоги користувачам досягати своїх цілей

При створенні замовлення користувач має змогу побачити кількість набраного індексу індивідуального насичення та скоригувати свій раціон під свої потреби.

Користувач має можливість готувати прийоми їжі заздалегіть та розраховувати індекс насиченості ще до прийому їжі що дозволить користувачу розуміти що потрібно додати чи відняти з раціону.

ВИСНОВКИ

В ході виконання роботи було досліджено існуючі методи гейміфікації у додатках для підрахунку БЖВ та калорійності продуктів. Аналіз показав, що гейміфікація значно підвищує залученість користувачів, сприяє їх мотивації та утриманню інтересу до використання додатків, що орієнтовані на контроль харчування та здоров'я.

В результаті виконання роботи створено додаток для підрахунку БЖВ та калорійності продуктів що реалізує новий метод гейміфікації, а саме врахування індивідуального індекса насиченості. Цей підхід дозволяє більш точно оцінювати стан користувача та відповідно адаптувати рекомендації щодо харчування.

Результати дослідження у вигляді додатку для підрахунку БЖВ та калорійності продуктів можливо використовувати для інформування людей про здорові харчові звички, тим самим потенційно зменшуючи кількість захворювань викликаних зайвою вагою чи іншими розладами харчової поведінки. Завдяки інтеграції нових методів гейміфікації, додаток здатний забезпечити користувачів мотивацією та підтримкою в процесі досягнення здорових звичок.

В подальшому додаток можливо доповнювати додатковими методами гейміфікації та досліджувати вплив конкретних видів гейміфікації на залученість користувачів. Розширення функціональності додатку передбачає включення персоналізованих порад та рекомендацій, соціальних функцій для взаємодії між користувачами, а також інтеграцію з іншими платформами та сервісами для підвищення зручності використання.

Загалом, подальші дослідження та розвиток додатку сприятимуть більш ефективному використанню гейміфікації у сфері здоров'я та харчування, що в свою чергу допоможе користувачам досягати кращих результатів у підтримці здорового способу життя.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Burke B. Gamify: How Gamification Motivates People to Do Extraordinary Things. Burke Publishing Group, 2014. 256 p.

2. Energy balance and its components: implications for body weight regulation. American Journal of Clinical Nutrition. 2012. Vol. 4, no. 989-994.

3. Jeukendrup, A. E., & Gleeson, M. Sport Nutrition: An Introduction to Energy Production and Performance. Human Kinetics, 2010. 420 p.

4. Walls C. Spring in Action. Manning Publications, 2023. 720 p.

5. Schildt H. _The Complete Reference. McGraw-Hill Education, 2023.

6. Johnson R. Professional Java Development with the Spring Framework. Wiley Publishing, 2005.

7. Beighley L. Head First SQL. O'Reilly Media, 2007.

8. Gruzdo, I., Kyrychenko, I., Tereshchenko, G., Shanidze, O. (2023). Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization. CEUR Workshop Proceedings, 3403, 387–409.

9. Gruzdo, I., Kyrychenko, I., Tereshchenko, G., Shanidze, N. (2021). Metrics applicable for evaluating software at the design stage. 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), Харків, Україна, 22-23 квітня 2021. CEUR Workshop Proceedings, 2870, Volume I, 916-936.