МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук
Кафедра Програмної інженерії

**КВАЛІФІКАЦІЙНА РОБОТА**
**Пояснювальна записка**

другий (магістерський)
(рівень вищої освіти)

# Дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога

Виконала:
студентка 2 курсу групи ІПЗздм-20-1
Непомняща І. В.
(прізвище, ініціали)

Спеціальність 121 - Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц. Вечур О. В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З. В. Дудар

2022 р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет: Комп'ютерних наук

Кафедра: Програмної інженерії

Рівень вищої освіти: другий (магістерський)

Спеціальність: 121 – Інженерія програмного забезпечення

(код і повна назва)

Тип програми: освітньо-наукова програма

Освітня програма: Інженерія програмного забезпечення

Затверджую:

Зав. кафедри _____

(підпис)

«___» _____2022 р.

**ЗАВДАННЯ**

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентки Непомнящої Інни Вікторівни

(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога»

затверджена наказом університету від «_24_»_03__ 2022 р. No 31 Стз

2. Термін здачі студентом закінченої роботи «_19_»_05___ 2022 р.

3. Вихідні дані до проекту: *електронні ресурси, методи вирішення оптимізаційних задач, методичні вказівки до виконання кваліфікаційної роботи магістра.*

4. Зміст розрахунково-пояснювальної записки: *вступ, аналіз проблемної області та постановка задач, опис прийнятих проєктних рішень, висновки, перелік посилань.*

5. Перелік графічного матеріалу: 44 рисунки, 8 таблиць.

# КАЛЕНДАРНИЙ ПЛАН

| Номер | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз предметної області | 17.01.2022-31.01.2022 | виконано |
| 2 | Постановка задачі | 31.01.2022-07.02.2022 | виконано |
| 3 | Аналіз існуючих методів | 07.02.2022-21.02.2022 | виконано |
| 4 | Розробка математичних моделей і алгоритмів | 21.02.2022-07.03.2022 | виконано |
| 5 | Розробка бази даних і архітектури | 07.03.2022-21.04.2022 | виконано |
| 6 | Програмна реалізація | 21.04.2022-30.04.2022 | виконано |
| 7 | Експериментальне дослідження методів | 30.04.2022-07.05.2022 | виконано |
| 8 | Підготовка пояснювальної записки | 07.05.2022-18.05.22 | виконано |
| 9 | Підготовка презентації та доповіді | 13.05.22-18.05.22 | виконано |
| 10 | Попередній захист | 19.05.22 | виконано |
| 11 | Нормоконтроль, рецензування | 20.05.22 | виконано |
| 12 | Занесення диплома в електронний архів | 22.05.22 | виконано |
| 13 | Допуск до захисту у зав. кафедри | 23.05.22 | виконано |

Дата видачі завдання 17.01.2022 р.

Керівник доц. _____(Вечур О. В.)

Завдання прийняла до виконання_____ (Непомняща І. В)

# РЕФЕРАТ/ABSTRACT

Пояснювальна записка містить: 73 с., 44 рис., 8 табл., 22 джерела.

JAVA, JAVASCRIPT, MYSQL, SPRING BOOT, БАЗА ДАНИХ, ДІЄТА, ЗДОРОВ'Я, КАЛОРІЯ, ОПТИМІЗАЦІЙНА ЗАДАЧА, УГОРСЬКИЙ МЕТОД.

Об'єктом дослідження є методи вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога.

Метою роботи є дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога і підвищення ефективності рішення задач про призначення на базі результатів експериментального дослідження методів їх вирішення.

У результаті роботи проведено аналіз та вибір методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога. Проведено експериментальне дослідження методів. Програмно реалізовано систему віртуального дієтолога.

The explanatory note contains: 73 pages, 44 pictures, 8 tables, 22 sources.

CALORIES, DATABASE, DIET, HEALTH, HUNGARIAN METHOD, JAVA, JAVASCRIPT, MYSQL, OPTIMIZATION TASK, SPRING BOOT.

The object of research is the methods of solving optimization problems about purpose of creating a system of virtual nutritionist.

The aim of the work is to study the methods of solving optimization problems about the purpose of creating a system of virtual nutritionist and to increase the efficiency of solving assignment problems based on the results of experimental research of Hungarian and greedy algorithms.

As a result, the analysis and selection of methods for solving optimization problems on the purpose of creating a system of virtual nutritionist. The experimental research of methods is carried out. The system of the virtual nutritionist is implemented.

Умови публікації пояснювальної записки

Я, Непомняща Інна Вікторівна, студентка групи ІПЗздм-20-1, здобувач вищої освіти на другому (магістерському) рівні, кафедра Програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога», що буде представлена до ЕК 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 ознайомлена 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

# ВСТУП

Правильне харчування та здоровий спосіб життя є нероздільними. Споживана нами їжа забезпечує постійне оновлення, розвиток клітин та тканин організму, є джерелом енергії. Продукти харчування – це джерела речовин, із яких синтезуються гормони, ферменти та інші регулятори обмінних процесів. Обмін речовин повністю залежить від характеру харчування. Склад їжі, її кількість та властивості визначають фізичний розвиток та ріст, захворюваність, працездатність, тривалість життя та нервово-психічний стан. Із їжею до нашого організму повинна надходити достатня, але не надлишкова, кількість білків, жирів, вуглеводів, мікроелементів, вітамінів та мінеральних речовин у правильних пропорціях.

Сучасна медицина зіткнулася з хворобами, виникнення яких в основному залежить від способу життя і особливо від якості та характеру харчування. Незважаючи на явне благополуччя й збільшення тривалості життя семимильними кроками наступають ожиріння та цукровий діабет у всьому світі, не виняток й Україна. Поруч із захворюваннями, пов'язаними з надмірним харчуванням, багато людей страждає від дефіциту необхідних поживних речовин, адже кількість не визначає якість, а дисбаланс у необхідних організму речовин, мікроелементів та вітамінів призводить до ослаблення імунної системи та інших негативних наслідків.

Поряд з тим стає проблема, що людям усе важче отримати якісне обслуговування в сфері дієтології. Вони постійно зіштовхуються з великими чергами в лікарнях та некваліфікованими лікарями. Але саме головне, що вони витрачають левову частку свого часу на очікування дієтолога, щоб отримати в нього поради стосовно лікування або почути швидку відповідь на своє запитання. Адже для цього вам потрібно прийти в години, коли цей лікар доступний, і відстояти чергу з інших людей, в яких такі ж самі проблеми.

Також однією з ключових проблем є недостовірність інформації в Інтернет-ресурсах. Адже ви можете потрапити на сайт, інформація на якому зовсім не відповідає правильному лікуванню та схудненню. Тож якщо притримуватися порад цього ресурсу, можна ще більше нашкодити своєму здоров'ю.

Під час виконання магістерської кваліфікаційної роботи був проведений аналіз проблемної області реалізації методів вирішення задач про призначення на основі публікацій світових та вітчизняних фахівців в проблемній області (див. додаток А), розглянуті основні існуючі методи. Метою даної роботи є дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога.

В ході роботи магістра було:

– проведено аналіз оптимізаційних моделей і методів;

– розроблено математичну модель задачі про призначення;

– розроблено угорський і жадібний алгоритми вирішення задачі про призначення;

– виконано програмну реалізацію системи;

– сплановані та проведені експериментальні дослідження розроблених методів.

У додатках Б, В представлено звіт результатів перевірки кваліфікаційної роботи на унікальність тексту та експертний висновок нормоконтролю відповідно. За результатами роботи магістра було розроблено презентацію (див. додаток Г), створено тези доповіді на дванадцятій міжнародній науково-технічній конференції (див. додаток Д) і на конференції «Молодіжний форум» (див. додаток Е).

# 1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

## 1.1 Аналіз проблемної області дослідження

У наші дні сучасна людина користується програмними системами кожен день, тому що це стало невід'ємною частиною життя [1]. Це можуть бути навчальні, медичні, а також націлені на дієтологію програмні системи [2]. Оскільки у світі існує велика кількість дієт, страв і інгредієнтів, необхідно знайти спосіб зберігання та швидкого отримання даних про ці дієти, страви та інгредієнти. Тому з'являється необхідність використовування комп'ютерної техніки, за допомогою якої можливо зберігати велику кількість даних, а також робити швидкі обчислення.

Ще однією причиною використання програмних систем є те, що кожна людина хоче зекономити свій час. Люди звикли жити в швидкому ритмі, а часто черги та дорога забирають дуже багато часу. Тому записатися онлайн до лікаря або отримати віддалену консультацію є дуже зручним. Але це не є повсякденними справами, тому не так багато часу може бути зекономлено, як, наприклад, замовлення їжі через сервіс доставки. Так людина не має потреби витрачати свій час на дорогу до магазину, вибирання продуктів і дорогу назад додому.

На жаль, замовлення їжі через сервіс доставки також має недоліки. Людина все одно має витратити свій час на те, щоб вибрати продукти, які необхідно доставити, гроші, тому що доставка не буває безкоштовною, та завжди бути на зв'язку, оскільки кур'єру може знадобитися допомога у виборі продуктів. Та найгірше, що може статися, - людина не отримає бажаний продукт, бо кур'єр може щось переплутати або забути.

У зв'язку з усім вище описаним з'являється актуальність створення програмної системи [3], яка б допомагала у швидкому підборі продуктів, мала б інформацію, в якому магазині можна придбати ці продукти найдешевше понад з усіх інших магазинів, і враховувала, до якого магазина дорога займе найменше часу.

Але обробка та вирахування такої великої кількості даних про продукти та дієти також буде займати багато часу. Тому процес розподілення, в якому магазині придбати який продукт, необхідно оптимізувати. Окрім оптимізації часу очікування результатів програмної системи, необхідно розподілити продукти по магазинах так, щоб кількість витрачених коштів була найменшою, кількість придбаних продуктів найближчою до кількості продуктів, указаної в дієти, та найменшої кількості часу та коштів, витрачених на дорогу до магазину. Також існує безліч інших задач у цій і інших областях, які можуть потребувати оптимізації.

Це вирішується за допомогою оптимізаційних задач [4]. Розв'язати оптимізаційну задачу означає знайти її оптимальний розв'язок чи встановити, що його немає. Методи розв’язку оптимізаційних задач – це методи математичного програмування. Оптимізаційна модель [5] є двох видів: задачі максимізації та мінімізації.

Задача максимізації полягає у знаходженні найбільшого значення цільової функції. Зазвичай відшукується глобальний максимум, і додатковою вимогою при аналізі рішення є перевірка того, чи є знайдений максимум глобальним чи локальним, тобто чи дійсно знайдено шукане рішення задачі. Множення цільової функції на від’ємний множник перетворює цю задачу максимізації в задачу мінімізації.

Для того, щоб розподілити продукти по магазинах, необхідно призначити, в якому магазині слід придбати який продукт. Для цього існує клас оптимізаційних задач про призначення.

Оптимізаційна задача про призначення – це окремий випадок транспортної задачі лінійного програмування. Лінійне програмування є гілкою математичного програмування – це розділ математики, що розробляє різні методи розв'язання екстремальних завдань.

Мета задачі про призначення – досягти максимальної корисності або мінімальних витрат при призначенні виконавців на роботи, отже, цільова функція матиме вигляд (1.1):

$$max(min)Z = c_{11}x_{11} + c_{12}x_{12} + \cdots + c_{1n}x_{1n} + c_{21}x_{21} + \\ + c_{22}x_{22} + \cdots + c_{2n}x_{2n} + \cdots + c_{n1}x_{n1} + c_{n2}x_{n2} + \cdots + c_{nn}x_{nn}, \quad (1.1)$$

де $x_{ij}$ – факт призначення і-го виконавця на j-ту роботу.

Для того, щоб користуватися такою програмною системою, необхідно її реалізувати. Для цього треба спланувати та розробити архітектуру системи. Проектування архітектури є найбільш важливим етапом, адже на ній буде основуватися вся система. Для створення такої системи можна обрати клієнт-серверну архітектуру [6]. Особливості такої моделі полягають у тому, що користувач надсилає певний запит на сервер, де той системно обробляється, і кінцевий результат надсилається клієнту. У можливості сервера входить одночасне обслуговування кількох клієнтів.

Наступним кроком є створення бази даних [7]. Як було описано вище, існує дуже велика кількість страв і продуктів, які складають ці страви, тобто є необхідність у зберіганні великої кількості даних. Тому потрібно вибрати базу даних, яка задовольняла б нашим вимогам. Існують такі види баз даних, як ієрархічні, мережеві, реляційні, об’єктні.

Ієрархічні бази даних можуть бути представлені як дерево, що складається з об'єктів різних рівнів. Верхній рівень займає один об'єкт, другий - об'єкти другого рівня і т.д. Мережеві бази даних подібні до ієрархічних, за винятком того, що в них є покажчики в обох напрямках, які з'єднують споріднену інформацію. Об'єктна СУБД ідеально підходить для інтерпретації складних даних, на відміну від реляційних СУБД, де додавання нового типу даних досягається ціною втрати продуктивності або за рахунок різкого збільшення термінів і вартості розробки додатків. Об'єктна база, на відміну від реляційної, не вимагає модифікації ядра при

додаванні нового типу даних. Новий клас і його екземпляри просто надходять у зовнішні структури бази даних. Система управління ними залишається без змін.

Реляційні бази даних [8] є базами даних, які використовуються для зберігання та надання доступу до взаємопов'язаних елементів інформації. Реляційні бази даних засновані на реляційній моделі – інтуїтивно зрозумілому, наочному табличному способі представлення даних. Кожен рядок, що містить у таблиці такої бази даних, є записом з унікальним ідентифікатором, який називають ключем. Стовпці таблиці мають атрибути даних, а кожен запис зазвичай містить значення для кожного атрибута, що дозволяє легко встановлювати взаємозв'язок між елементами даних.

Для створення програмної системи віртуального дієтолога найкращим варіантом є вибір реляційної бази даних MySQL [9].

Проектування бази даних складається з декількох етапів: концептуальне, логічне та фізичне проектування.

Концептуальна модель представляє загальний погляд на дані. Розрізняють два головних підходи до моделювання даних при концептуальному проектуванні:

− семантичні моделі;

− об'єктні моделі.

Зближення цих моделей реалізується в розширеному ER-моделюванні.

Логічне проектування полягає в створенні логічної моделі на основі вибраної моделі даних.

Наступним кроком є вибір засобів реалізації програмної системи. Для цього нам знадобиться мова програмування бекенд-частини Java 14, фреймворк Spring Boot [10], мова розмітки HTML 5, таблиця стилів CSS 3, мова програмування фронтенд-частини JavaScript, фреймворки jQuery, Bootstrap, Spring Thymeleaf [11].

Розроблена програмна система допоможе зменшити час на вибір та покупку продуктів завдяки алгоритму оптимального розподілення, в якому магазині слід придбати який продукт, зекономити гроші та спростити повсякденні справи та обов’язки.

## 1.2 Аналіз існуючих аналогів

Існує декілька систем, які є аналогами розроблюваної системи, але вони мають ряд недоліків, які будуть усунені у програмній системи віртуального дієтолога. Розглянемо основні з них.

«Love Sales» (див. рис. 1.1) – програмна система, в якій можна побачити знижки у популярних магазинах України. Знижки оновлюються кожен день, а програмна система є повністю безкоштовною. Можна додавати улюблені магазини в «Обране», отримувати сповіщення про нові акції, доступне на декількох операційних системах.

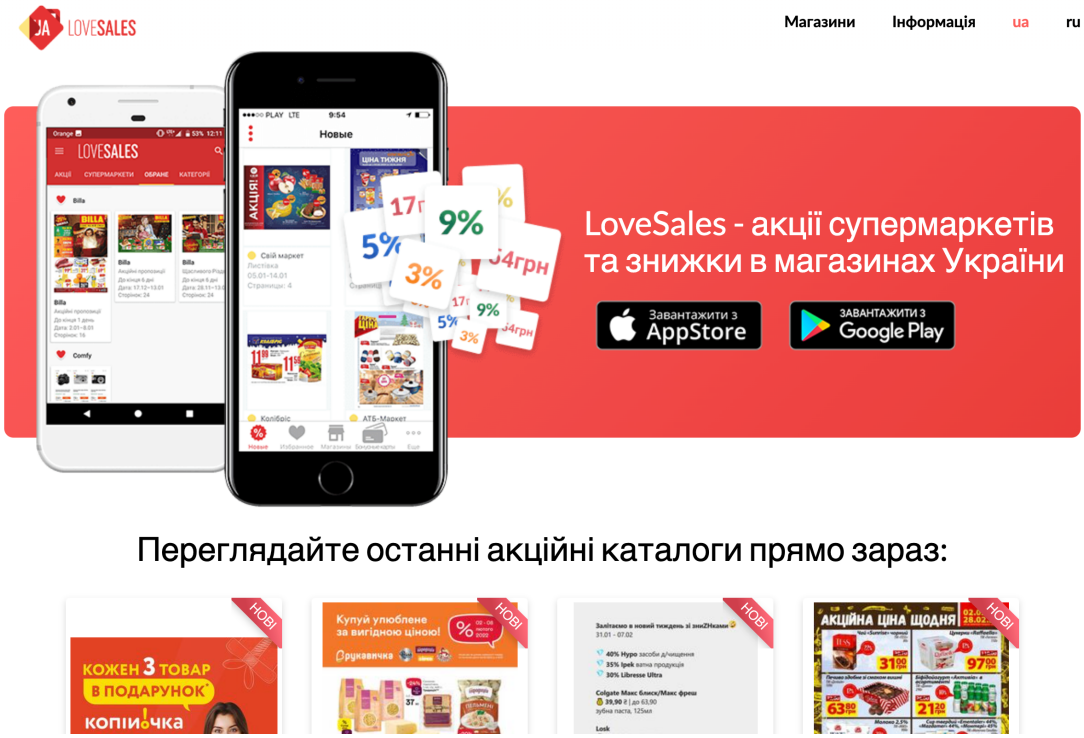


Рисунок 1.1 – Інтерфейс програмної системи «Love Sales»

Перевагою системи «Love Sales» є те, що можна побачити акційні товари, що допоможе зберегти бюджет. Дана система зручна в користуванні, але, на жаль, не може розподілити, в якому магазині буде найбільш вигідно купити той чи інший

продукт.

«GoToShop» (див. рис. 1.2) – система, яка допомагає шукати найкращі ціни на товари та послуги, включаючи продукти харчування, побутову хімію, будівельні матеріали, меблі, дитячі товари та багато іншого.

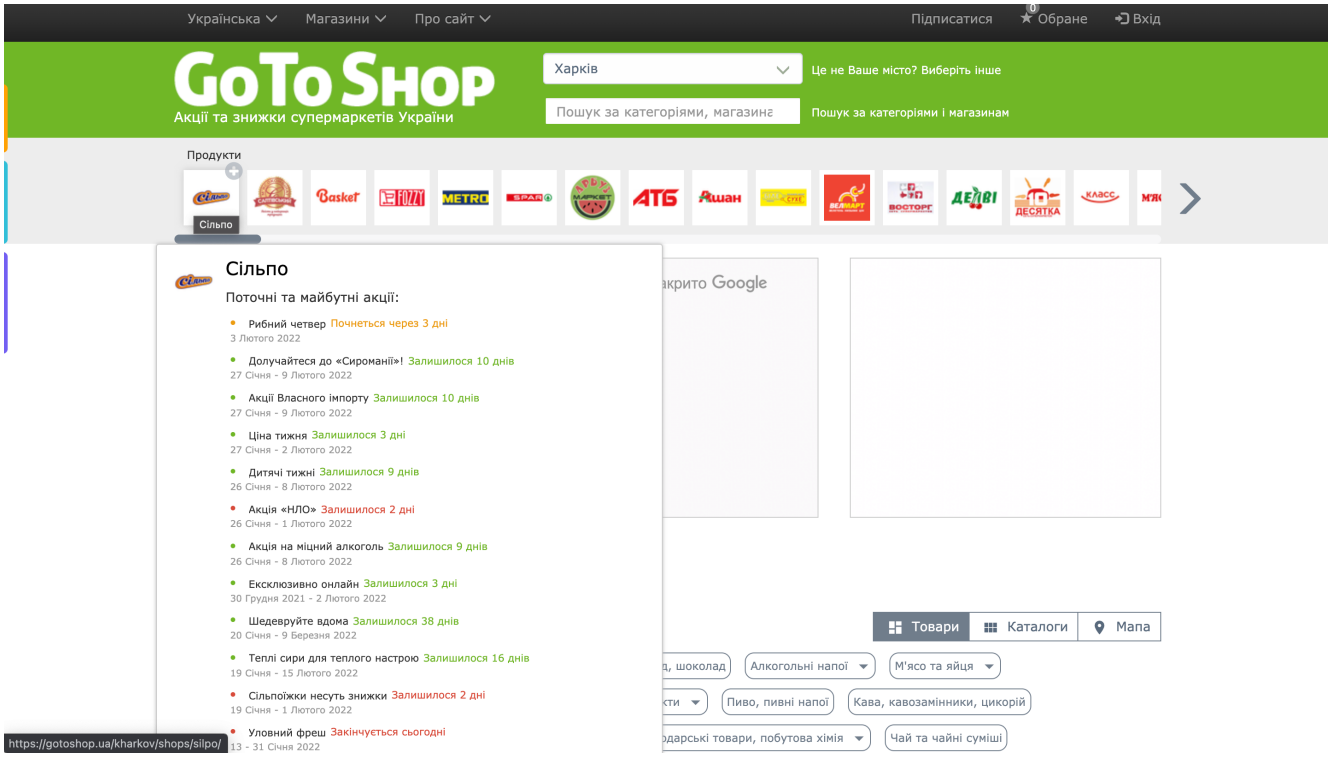


Рисунок 1.2 – Інтерфейс програмної системи «GoToShop»

У «GoToShop» розміщена інформація про акції та знижки магазинів та торговельних мереж більш ніж у 400 містах України. Для зручності пошуку всі позиції розділені на категорії, знижки, що сподобалися, можна додавати в обране або відправляти на електронну пошту. Крім того, користувачі можуть залишати коментарі та відгуки про той чи інший товар. Всі адреси магазинів вашого міста нанесені на зручну карту, що дає змогу бачити акції поблизу. Також дана система має доволі зручний інтерфейс.

Із недоліків можна зазначити те, що пошук товарів можна здійснювати тільки по категоріям та наявність великої кількості реклами. Також немає розподілення, в якому магазині буде найбільш вигідно купити продукти.

## 1.3 Постановка задачі

Метою роботи є дослідження методів вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога. Тобто необхідно проаналізувати методи вирішення оптимізаційних задач і обрати найкращі з них для проведення експериментів. Для цього потрібно розробити систему, де можна буде практично застосувати результати дослідження. Система повинна буде призначати, в якому магазині слід придбати який продукт, щоб користувач витратив на покупки якнайменше коштів та якнайменше часу на дорогу до магазину та назад. Також система надаватиме можливість вираховувати калорії та прогнозувати результати дієти.

Бекенд-частина має бути розроблена за допомогою мови програмування Java 14 і фреймворку Spring Boot.

Фронтенд-частина програмної системи має бути розроблена за допомогою мови розмітки HTML 5, таблиці стилів CSS 3, мови програмування JavaScript, фреймворків jQuery, Bootstrap, Spring Thymeleaf.

Програмна система віртуального дієтолога матиме наступні основні функції, такі як:

– занесення та зміна особистої інформації користувача;
– пошук продуктів, дієт, дієтичних страв, магазинів;
– калькулятор калорій;
– щоденне фіксування дотримання дієти;
– прогнозування результатів дієти;
– вирахування вартості дієти у конкретному магазині;
– вирахування вартості та часу проїзду до конкретного магазина;
– призначення, в якому магазині найдешевше буде придбати продукти для конкретної дієти.

Підсумовуючи, можна виділити наступні задачі, які необхідно виконати під час магістерського дослідження:

– провести аналіз існуючих методів вирішення оптимізаційної задачі про призначення та обрати найкращі для подальшого дослідження;

– провести аналіз та моделювання предметної області дієтології;

– побудувати математичну модель оптимізаційної задачі про призначення в даній предметній області;

– спроектувати базу даних;

– розробити алгоритми вирішення оптимізаційної задачі про призначення на базі розробленої моделі та обраних методів;

– розробити архітектуру та реалізувати програмну систему;

– провести експериментальне дослідження розроблених методів.

# 2 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

## 2.1 Аналіз та моделювання предметної області дієтології

Здоров’я – це перша і найважливіша потреба людини, що визначає працездатність і забезпечує гармонійний розвиток особистості. Це найважливіша умова пізнання навколишнього світу, самоствердження і людського щастя. Здоровий спосіб життя – це спосіб життя, заснований на етиці, розумно організований, активний, працьовитий, уникаючий шкідливого впливу навколишнього середовища, що допомагає зберегти моральне, психічне та фізичне здоров’я у старості.

Здоровий спосіб життя можна охарактеризувати як активну діяльність людини, основною метою якої є збереження і зміцнення здоров’я. Слід зазначити, що здоровий спосіб життя особистості та сім’ї не формується самостійно обставинами, а формується цілеспрямовано і безперервно протягом усього життя людини.

Здоровий спосіб життя – це вид активної фізичної активності, що включає такі основні елементи: режим дня (правильне поєднання), розумний підхід до праці та відпочинку, гігієна харчування, добре дихання, добрий сон, відсутність шкідливих звичок, аеробіка, масаж, фітнес.

Для описаної предметної області можна виділити такі об’єкти, як користувач, персональна дієта, дієта, страва, продукт, категорія, магазин.

Об’єкт «користувач» містить: ідентифікатор, ім’я, пароль, вік, електронну адресу, стать, вагу, зріст, дані про роль користувача.

Об’єкт «персональна дієта» містить: ідентифікатор, дані про користувача, дані про дієту, дати початку та кінця дієти, ціль персональної дієти.

Об’єкт «дієта» містить: ідентифікатор, назву та тип.

Об’єкт «страва» містить: ідентифікатор, тип страви, назву та рецепт страви.

Об’єкт «продукт» містить: ідентифікатор, назву продукту, кількість калорій, жирів, білків і вуглеводів, категорію продукту.

Об’єкт «категорія» містить: ідентифікатор, назву та опис категорії.

Об’єкт «магазин» містить: ідентифікатор, назву, координати, час початку та кінця роботи.

На рисунку 2.1 можна побачити схему взаємодії головних об’єктів системи.

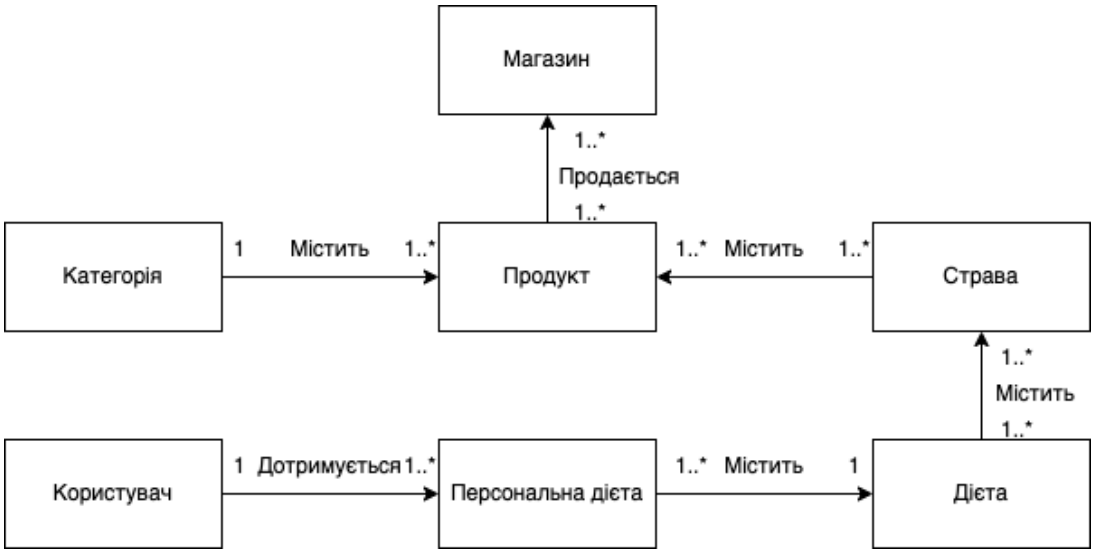


Рисунок 2.1 – Загальна діаграма класів

Проектування системи було проведено за допомогою UML-діаграм. Уніфікована мова моделювання (UML) – це стандартизована мова моделювання, яка дозволяє розробникам визначати, візуалізувати, конструювати та документувати артефакти програмної системи. Таким чином, UML робить ці артефакти масштабованими, надійними та надійними у виконанні. UML – важливий аспект, що бере участь у розробці об'єктно-орієнтованого програмного забезпечення. Він використовує графічні позначення для створення візуальних моделей програмних систем.

Архітектура UML базується на метаоб'єкті, що визначає основу для створення мови моделювання. Вони досить точні, щоб генерувати всю програму. Повністю виконуваний UML може бути розгорнутий на декількох платформах за

допомогою різних технологій і може використовуватися з усіма процесами протягом циклу розробки програмного забезпечення. UML призначений для того, щоб дати можливість користувачам розробити виразну, готову до використання мову візуального моделювання.

За допомогою UML-діаграм було відображено набір функцій системи. Першою було створено діаграму варіантів використання (див. рис. 2.2).

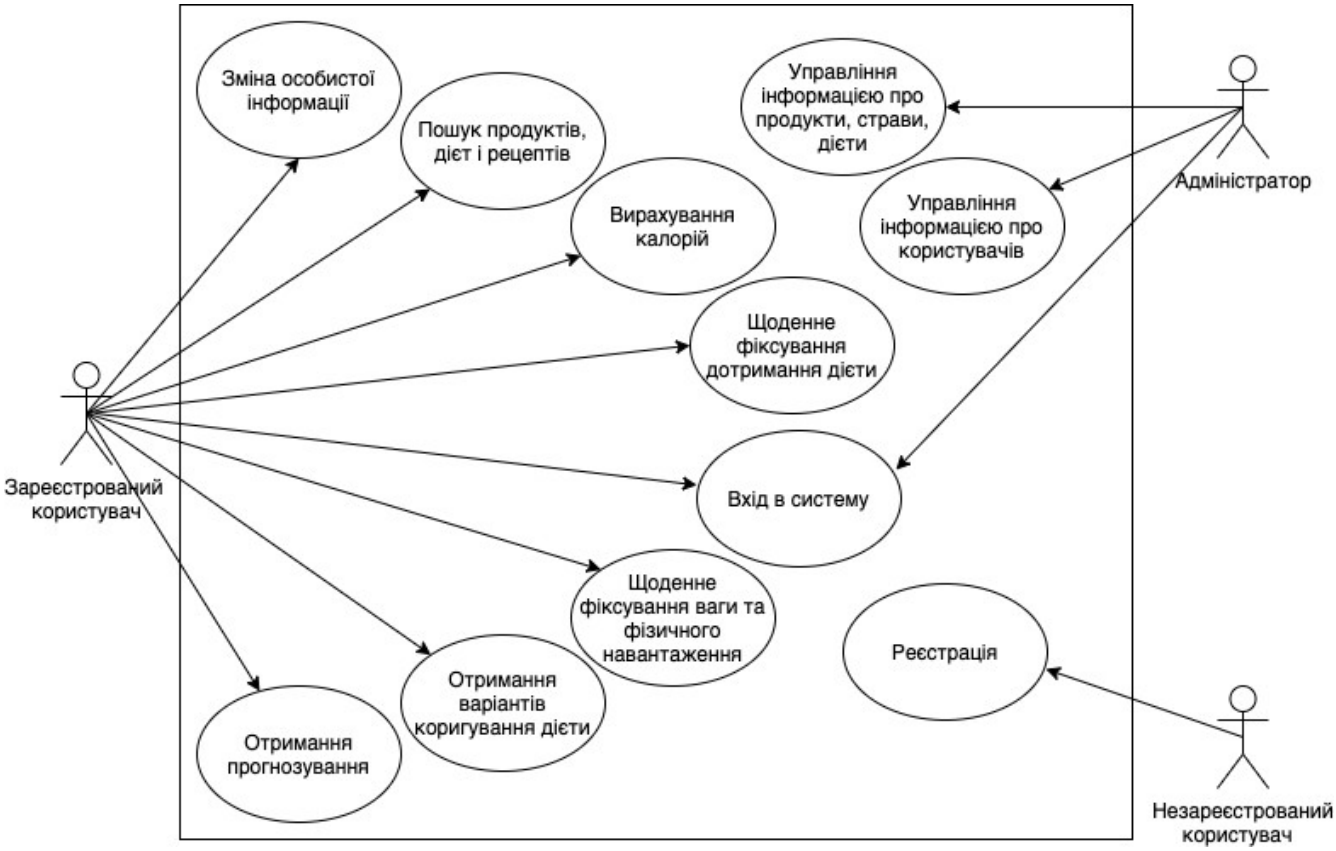


Рисунок 2.2 – Діаграма варіантів використання

Діаграма варіантів використання – це найпростіша з поведінкових діаграм UML [12]. Ця діаграма використовується для опису цілей, які переслідує користувач програми, будь це або людина, або інша програма. Діаграма варіантів використання застосовується для опису функціональних вимог до програми, або її підсистем, або іншого об’єкта. Вона представляє загальну картину того, як програма буде використовуватись.

У системі передбачена наявність трьох користувачів: незареєстрований і зареєстрований користувачі та адміністратор.

Незареєстрований користувач може:

– реєструватися в системі.

Зареєстрований користувач може:

– входити до системи;

– змінювати інформацію про себе;

– здійснювати пошук продуктів, дієт і рецептів;

– вираховувати спожити та витрачені калорії;

– фіксувати дотримання дієти та фізичне навантаження;

– отримувати варіанти коригування дієти;

– отримувати прогнозування результатів дієти.

Адміністратор системи може:

– входити до системи;

– управляти інформацією про продукти, страви, дієти, користувачів.

Проведений аналіз і концептуальне моделювання дозволять спроектувати базу даних і розробити функціонал системи.

## 2.2 Розробка математичної моделі оптимізаційної задачі в області дієтології

Результат діяльності будь-якої системи визначається ефективністю використання ресурсів. Це тягне за собою необхідність пошуку найкращого варіанту використання ресурсів, що забезпечує знаходження максимальних значень, наприклад для прибутку, або ж мінімальних значень – для збитків. Такі завдання отримали назву задач оптимізації [13].

Під рішенням оптимізаційних задач розуміється процес вибору таких значень змінних x, які забезпечують оптимальне значення деякої функції f (x) у певному

діапазоні її існування. Під оптимальним значенням функції розуміється її екстремум, тобто її максимальне або мінімальне значення.

Найбільш поширеною оптимізаційною задачею є транспортна задача [14] – завдання про пошук економічного плану перевезень однорідного продукту з пунктів виробництва в пункти споживання.

Одним із видів транспортної задачі є класична. Це задача оптимального плану перевезень однорідного продукту в однорідні пункти споживання на однорідних транспортних засобах.

Транспортна задача є особливою формою пошуку оптимального плану перевезень вантажу з мінімальними витратами. За допомогою транспортної задачі можна знайти оптимальний план перевезення вантажів, витрачаючи при цьому мінімальну кількість витрат.

Мета транспортної задачі – забезпечити доставку продукції споживачеві у потрібний час і місце за мінімальної вартості трудових, матеріальних та фінансових ресурсів. Мета вважається досягнута, коли виконуються такі умови:

– необхідний товар;
– достойна якість;
– необхідна кількість;
– потрібний час;
– потрібне місце;
– мінімальні витрати.

Окремим випадком транспортної задачі є задача про призначення [15], у якій число пунктів виробництва дорівнює кількості пунктів призначення, тобто транспортна таблиця має форму квадрата. Крім того, у кожному пункті призначення обсяг потреби дорівнює 1, і величина пропозиції кожного пункту виробництва дорівнює 1.

Для розробки математичної моделі задачі про призначення була виділена проблема визначення, в якому магазині слід придбати які продукти для дієти, щоб сума витрат на продукти була найменшою. Нехай маємо n магазинів $S_i$ $(i = \overline{1,n})$,

де треба купити m продуктів $P_j$ $(j = \overline{1,m})$ для дієти. Також маємо відстань до і-го $(i = \overline{1,n})$ магазина $L_i$, час початку роботи $Tmin_i$, час кінця роботи $Tmax_i$, час спланованої покупки $Tpl$ та кількість j-го $(j = \overline{1,m})$ продукту, яка необхідна для дієти, $Q_j$. Відомі вартість $C_{ij}$ за одиницю товара j-го $(j = \overline{1,m})$ продукту для дієти в і-ому $(i = \overline{1,n})$ магазині (де одиницею товара можуть бути кілограми, літри тощо) та мінімальний для купівлі обсяг j-го $(j = \overline{1,m})$ продукту в і-му $(i = \overline{1,n})$ магазині, $V_{ij}$. Також маємо мінімальну кількість продуктів $K$, заради якої користувач згоден піти в j-ий $(j = \overline{1,m})$ магазин.

Умову задачі можна представити у вигляді таблиці 2.1.

Таблиця 2.1 – Табличне представлення оптимізаційної задачі

| | Продукти | | | | | Характеристика магазина | | |
|---|---|---|---|---|---|---|---|---|
| Магазини | $P_1$ | … | $P_j$ | … | $P_m$ | Відстань | Час початку роботи | Час кінця роботи |
| $S_1$ | $C_{11}$, $V_{11}$ | … | … | … | … | $L_1$ | $Tmin_1$ | $Tmax_1$ |
| … | … | … | … | … | … | … | … | … |
| $S_i$ | … | … | $C_{ij}$, $V_{ij}$ | … | … | $L_i$ | $Tmin_i$ | $Tmax_i$ |
| … | … | … | … | … | … | … | … | … |
| $S_n$ | … | … | … | … | $C_{nm}$, $V_{nm}$ | $L_n$ | $Tmin_n$ | $Tmax_n$ |
| Кількість продуктів | $Q_1$ | … | $Q_j$ | … | $Q_m$ | | | |

Розробимо математичну модель задачі про призначення. Введемо змінну $X_{ij}$ ($i = \overline{1,n}$, $j = \overline{1,m}$) – це факт планування покупки j-го продукту в і-ому магазині ($X_{ij} = 1$, якщо покупка планується, і $X_{ij} = 0$, якщо не покупка не планується).

Ціль задачі про призначення – досягти мінімальної вартості продуктів при призначенні, в якому магазині слід купити які продукти для дієти, відповідно цільова функція матиме такий вигляд (2.1).

$$minZ = \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij}X_{ij}[Q_j] + \sum_{i=1}^{n} L_i|X_{ij}|, \tag{2.1}$$

де $|X_{ij}| = 1$, якщо $\exists X_{ij} > 0$.

Складемо систему обмежень. Якщо в одному магазині кількість запланованих до покупки продуктів більша за мінімальну кількість продуктів, заради якої користувач може піти в магазин, то можна спланувати покупку в цьому магазині (2.2):

$$\sum_{j=1}^{m} X_{ij}Q_j \geq K, \forall i = \overline{1,n} \tag{2.2}$$

Також складемо обмеження, що необхідно купити всі продукти, що входять до дієти (2.3):

$$\sum_{j=1}^{m} X_{ij} \geq 1, \forall i = \overline{1,n} \tag{2.3}$$

Оскільки магазин має час початку та кінця роботи, необхідно спланувати покупку продуктів в залежності від цього, тобто складемо обмеження, що покупка має бути спланована після початку роботи магазину (2.4):

$$\sum_{j=1}^{m} |X_{ij}|Tmin_i \leq Tpl \quad \forall i = \overline{1,n} \tag{2.4}$$

Також складемо обмеження, що покупка має бути спланована до закриття магазину (2.5):

$$\sum_{j=1}^{m} |X_{ij}| Tmax_i \geq Tpl \quad \forall i = \overline{1,n} \tag{2.5}$$

Ще одне обмеження, яке потрібно скласти – обмеження стосовно доцільності покупки продуктів в магазині з огляду на розмір мінімальної упаковки (2.6):

$$\sum_{i=1}^{n} X_{ij} V_{ij} \leq Q_j, \forall j = \overline{1,m} \tag{2.6}$$

Таким чином, цільова функція та система обмежень складають математичну модель задачі про призначення.

## 2.3 Аналіз методів вирішення задачі про призначення

Будь-яка задача про призначення може бути вирішена з використанням методів лінійного програмування або алгоритму розв'язання транспортної задачі. Однак з огляду на особливу структуру цієї задачі було розроблено спеціальний алгоритм, який отримав назву угорського методу [16]. Також задачу можна розв’язати за допомогою спеціального метода Мака [17] чи класичними методами – симплекс-методом [18] і жадібним алгоритмом [19].

Симплекс-метод – це ітеративний процес спрямованого розв'язання системи рівнянь по кроках, який починається з опорного рішення і в пошуках кращого варіанта рухається по кутових точках області допустимого рішення, що покращують значення цільової функції, доки цільова функція не досягне оптимального значення. Алгоритм симплекс-методу включає наступні етапи. Упорядкування першого опорного плану – перехід до канонічної форми задачі

лінійного програмування шляхом запровадження невід'ємних додаткових балансових змінних. Перевірка плану оптимальність – кщо знайдеться хоча б один коефіцієнт індексного рядка менший за нуль, то план не оптимальний, і його необхідно покращити. Визначення провідних стовпця та рядки – з негативних коефіцієнтів індексного рядка вибирається найбільший за абсолютною величиною. Побудова нового опорного плану – перехід до нового плану здійснюється внаслідок перерахунку симплексної таблиці методом Жордана-Гаусса.

Ще одним алгоритмом, яким можливо вирішити задачу про призначення є жадібний алгоритм. Жадібний алгоритм це алгоритмічна стратегія, що використовується для того, щоб зробити найкращий необов'язковий вибір на дуже маленькій стадії і зрештою вивести глобально оптимальне рішення. Цей алгоритм вибирає найкраще рішення, яке можливе в даний момент, без урахування будь-яких наслідків. Жадібний метод говорить, що проблема повинна вирішуватися поетапно, при цьому кожен вхід розглядається з урахуванням того, що можливий цей внесок. Оскільки цей підхід фокусується лише на негайному результаті без урахування загальної картини, він вважається жадібним.

Жадібний алгоритм в основному будує рішення частинами і вибирає наступну частину таким чином, щоб відразу ж знайти найкраще рішення для розглянутої проблеми. Ніколи не переглядаючи зроблений раніше вибір, жадібний алгоритм не дає оптимального рішення, хоча і дає майже оптимальне рішення.

Обрання методів для подальшого дослідження можна звести до задачі ПР. Існують різні методи ПР, найбільш популярними є методи, засновані на згортках з теорії корисності [21].

Порівняємо ці методи за допомогою лінійної адитивної згортки з вагомими коефіцієнтами.

Множина альтернатив:

– симплекс метод;
– жадібний алгоритм;
– угорський метод;

– метод Мака.

Критерії вибору:

– складність алгоритму;

– наявна реалізація;

– використання пам’яті.

Шкали оцінок за критеріями:

– складність алгоритму:

1) лінійна складність – 3 балів;

2) поліноміальна складність – 2 бали;

3) експоненційна складність – 1 бал;

– наявна реалізація:

1) можливість повного використання наявних реалізацій – 3 бали;

2) можливість часткового використання наявних реалізацій – 2 бали;

3) немає можливісті використання наявних реалізацій – 1 бал;

– використання пам’яті:

1) лінійна складність – 3 бали;

2) квадратична складність – 2 бали;

3) поліноміальна складність – 1 бал.

У таблиці 2.2 наведено моделювання задачі ПР у загальному вигляді.

Таблиця 2.2 – Моделювання задачі ПР

| | Час роботи алгоритму | Наявна реалізація | Використання пам’яті |
|---|---|---|---|
| Симплекс метод | $O(2^n)$ | https://math.semestr.ru/simplex/simplex.php, MS Excel | $O(n^2)$ |
| Жадібний алгоритм | $O(n)$ | - | $O(n)$ |
| Угорський метод | $O(n^3)$ | - | $O(n^2)$ |
| Метод Мака | $O(n^4)$ | - | $O(n^3)$ |

У таблиці 2.3 наведено моделювання задачі ПР з оцінками та вагомими коефіцієнтами.

Таблиця 2.3 – Моделювання задачі ПР з оцінками та ваговими коефіцієнтами

| | Час роботи алгоритму | Наявна реалізація | Використання оперативної пам’яті | Z* |
|---|---|---|---|---|
| Симплекс метод | 1 | 3 | 2 | 1.06 |
| Жадібний алгоритм | 3 | 1 | 3 | 1.87 |
| Угорський метод | 2 | 1 | 2 | 1.27 |
| Метод Мака | 2 | 1 | 1 | 1.01 |
| Вагові коефіцієнти | 5/15 | 1/15 | 4/15 | |

Так як усі альтернативи в таблиці не є гіршими одна від одної хоча б за одним критерієм, то усі вони входять до множини Парето.

Метод рішення – за допомогою лінійної адитивної згортки з ваговими коефіцієнтами

Для симплекс методу результат згортки буде: $1 * \frac{5}{15} + 3 * \frac{1}{15} + 2 * \frac{4}{15} = 0.33 + 0.2 + 0.53 = 1.06$.

Для жадібного алгоритму результат згортки буде: $3 * \frac{5}{15} + 1 * \frac{1}{15} + 3 * \frac{4}{15} = 1 + 0.07 + 0.8 = 1.87$.

Для угорського методу результат згортки буде: $2 * \frac{5}{15} + 1 * \frac{1}{15} + 2 * \frac{4}{15} = 0.67 + 0.07 + 0.53 = 1.27$.

Для методу Мака результат згортки буде: $2 * \frac{5}{15} + 1 * \frac{1}{15} + 1 * \frac{4}{15} = 0.67 + 0.07 + 0.27 = 1.01$.

Згідно з розрахунками найкращими методами для вирішення оптимізаційної задачі є жадібний алгоритм і угорській метод.

## 2.4 Проектування бази даних

Для того, щоб реалізувати математичну модель задачі про призначення, необхідно почати з проектування бази даних. У якості бази даних було обрано реляційну базу даних MySQL.

Перша база даних буде складатися з таблиць «Магазин», «Продукт», «Партія продукту (див. рис. 2.3).


Рисунок 2.3 – Схема бази даних

Ця база даних буде використовуватися в сервісі призначення, в якому магазині слід придбати який продукт для дієти.

Друга база даних буде складатися з таблиць «Роль», «Користувач», «Категорія», «Продукт», «Тип страви», «Страва», «Інгредієнт», «Дієта»,

«Персональна дієта», «Розклад», «Персональний розклад», «Замір» (див. рис. 2.4). Ця база даних буде використовуватися в сервісі для підбору дієт, калькулятора калорій і прогнозування результатів дієти.

Обидві схеми баз даних було нормалізовано до третьої нормальної форми, тобто всі атрибути є взаємно незалежними та повністю залежать від первинних ключів.

## 2.5 Розробка алгоритмів для вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога

Для вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога було обрано угорський алгоритм і жадібний алгоритм.

Основна ідея угорського методу полягає в переході від вихідної квадратної матриці вартості C до еквівалентної їй матриці Ce з невід'ємними елементами і системою n незалежних нулів, з яких жодні два не належать одному і тому ж рядку або тому самому стовпцю. Для заданого n існує n! допустимих рішень. Якщо в матриці призначення X розташувати n одиниць так, що в кожному рядку і стовпці знаходиться лише по одній одиниці, розставлених відповідно до розташованих n незалежних нулів еквівалентної матриці вартості Ce, то отримаємо допустимі рішення задачі про призначення. Слід пам'ятати, що з будь-якого неприпустимого призначення відповідна йому вартість умовно належить рівної досить великому числу M в завданнях на мінімум. Якщо вихідна матриця не є квадратною, слід ввести додатково необхідну кількість рядків або стовпців, а їх елементам привласнити значення, що визначаються умовами завдання, можливо після редукції, а домінуючі альтернативи дорогі або дешеві виключити.

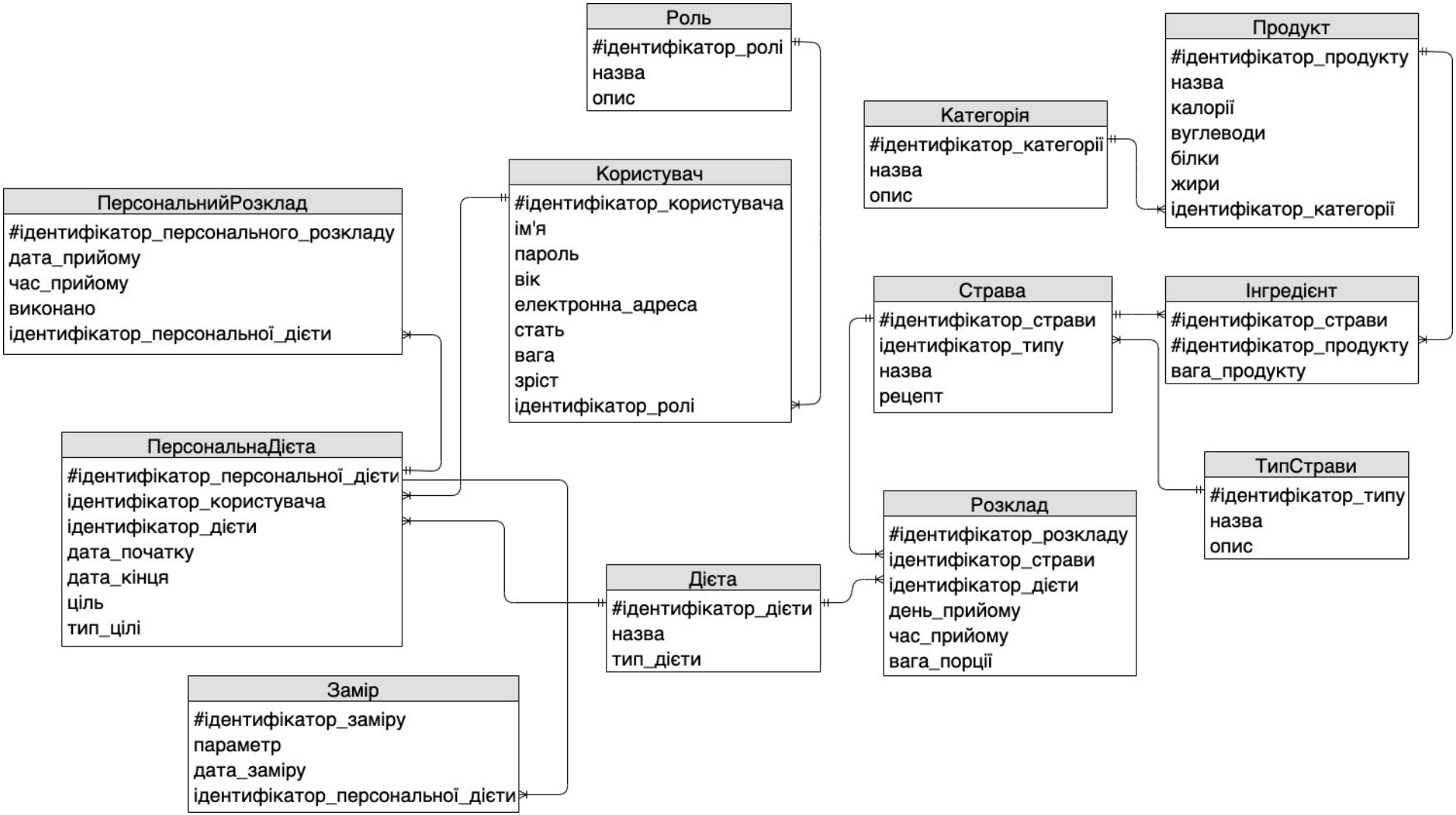


Рисунок 2.4 – Схема другої бази даних

На рисунку 2.5 зображена схема етапів виконання угорського алгоритму для призначення, в якому магазині слід придбати який продукт.

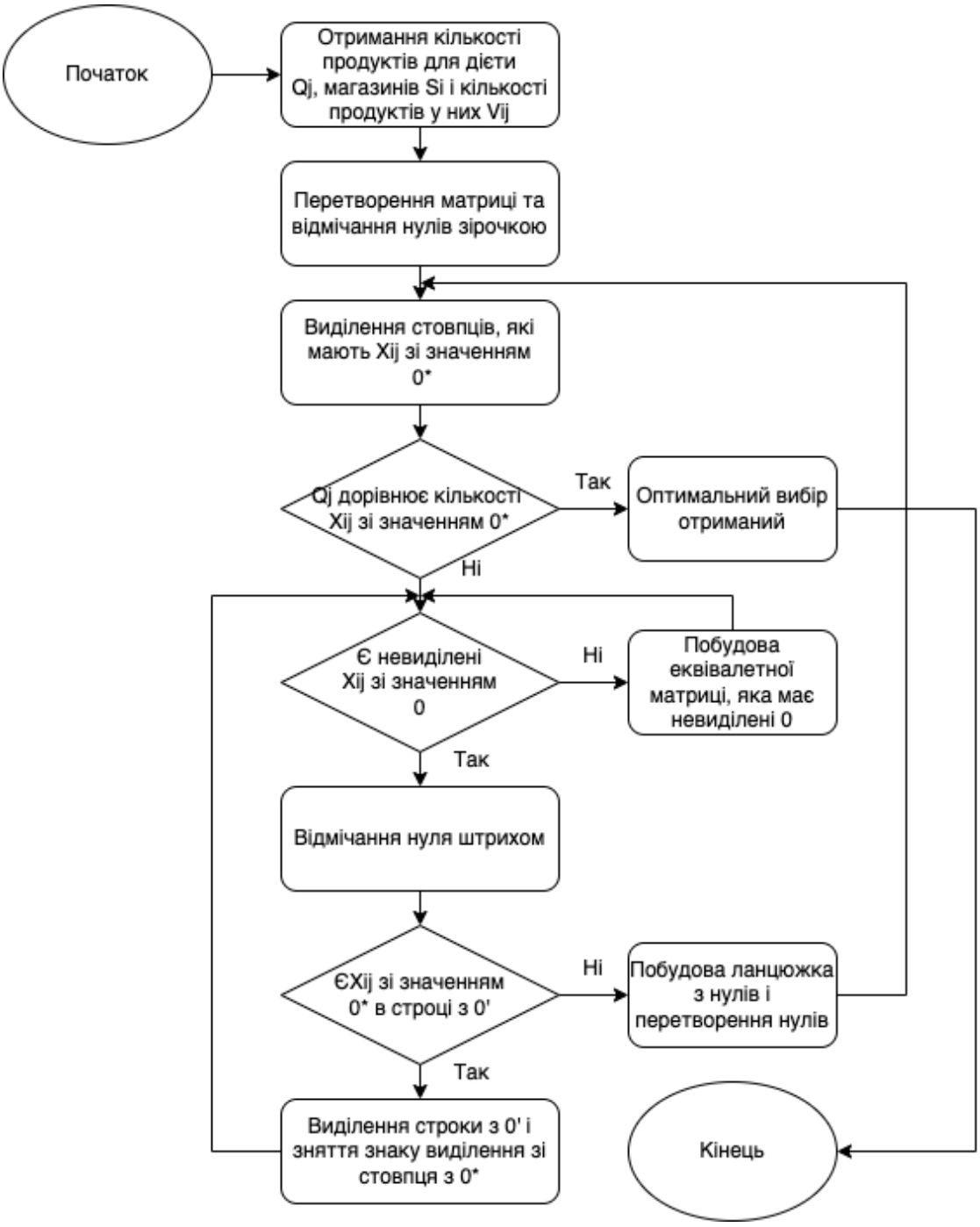


Рисунок 2.5 – Схема етапів виконання угорського алгоритму

За цією схемою згодом буде програмно реалізовано алгоритм призначення, в якому магазині слід придбати який продукт для дієти.

На рисунку 2.6 зображена схема етапів виконання жадібного алгоритму для призначення.

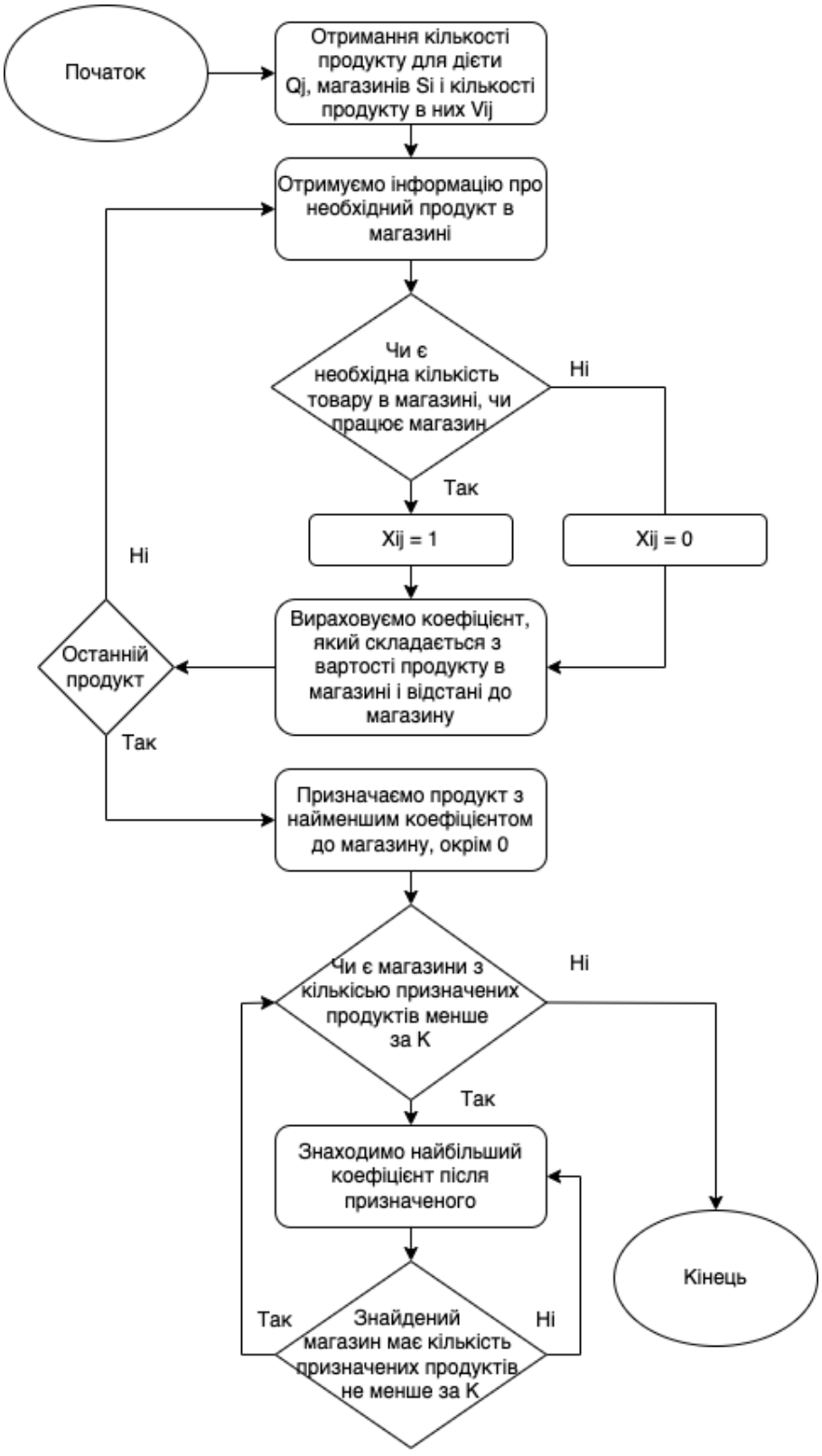


Рисунок 2.6 – Схема етапів виконання жадібного алгоритму

## 2.6 Розробка архітектури системи

Для проектів будь-якого розміру необхідно продумати архітектуру, оскільки непродуманість архітектури може мати погані наслідки, особливо для великих проектів.

Для системи віртуального дієтолога було обрано клієнт-серверну архітектуру (див. рис. 2.7).

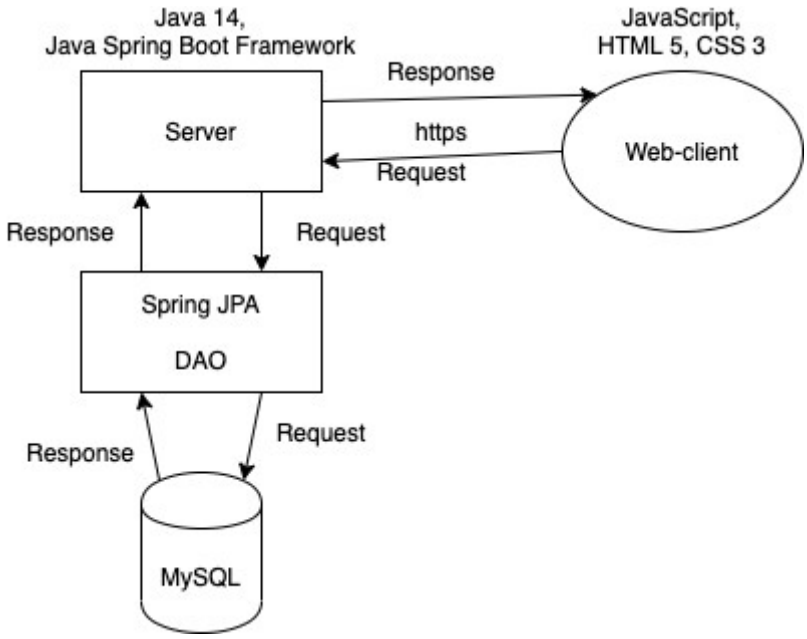


Рисунок 2.7 – Діаграма розгортання системи

Клієнт – локальний комп'ютер на стороні віртуального користувача, який виконує надсилання запиту на сервер для можливості надання даних або виконання певної групи системних дій.

Сервер – дуже потужний комп'ютер або спеціальне системне обладнання, яке призначається для вирішення певного кола завдань процесу виконання програмних кодів. Він виконує роботи сервісного обслуговування на клієнтські запити, надає користувачам доступ до певних системних ресурсів, зберігає дані або БД.

Особливості такої моделі полягають у тому, що користувач надсилає певний запит на сервер, де той системно обробляється та кінцевий результат надсилається клієнту. У можливості сервера входить і одночасне обслуговування кількох клієнтів.

В якості архітектурного шаблону було обрано MVC [19] (див. рис. 2.8). MVC (Model-view-controller) — архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення.

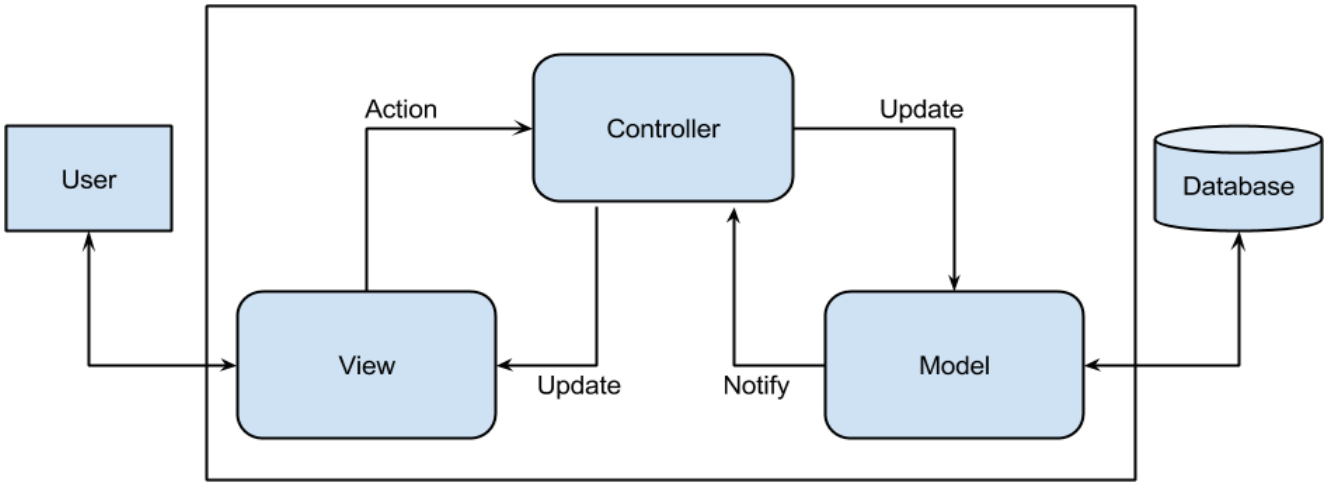


Рисунок 2.8 – Детальна схема обробки запиту користувача в шаблоні MVC

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета цього шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми [22].

# 3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

## 3.1 Вибір засобів програмної реалізації

При реалізації серверної частини системи було використано фреймворк для розробки enterprise-проектів на Java – Spring Boot Framework. Spring Boot має великий функціонал, але його найбільш значущими особливостями є: управління залежностями, автоматична конфігурація та вбудовані контейнери сервлетів.

Щоб прискорити процес управління залежностями, Spring Boot неявно пакує необхідні сторонні залежності для кожного типу програми на основі Spring і надає їх розробнику за допомогою так званих starter-пакетів.

Starter-пакети є набором зручних дескрипторів залежностей, які можна включити у свою програму. Це дозволить отримати універсальне рішення для всіх, пов'язаних зі Spring технологією, позбавляючи програміста від зайвого пошуку прикладів коду та завантаження з них необхідних дескрипторів залежностей.

Другою чудовою можливістю Spring Boot є автоматична конфігурація програми. Після вибору відповідного starter-пакета, Spring Boot спробує автоматично налаштувати Spring-додаток на основі доданих вами jar-залежностей.

Кожний Spring Boot web-додаток включає вбудований web-сервер. Розробникам не треба турбуватися про налаштування контейнера сервлетів та розгортання програми на ньому. Тепер програма може запускатися сама, як виконуваний jar-файл з використанням вбудованого сервера.

Якщо потрібно використовувати окремий HTTP-сервер, для цього достатньо виключити залежності за замовчуванням. Spring Boot надає окремі starter-пакети для різних HTTP-серверів.

Оскільки Java enterprise-проекти складаються з великої кількості файлів і додаткових бібліотек, було використано систему збірки Gradle для контролювання процесів збірки проекту.

Gradle було розроблено для побудови мультипроєктів, які можуть розростатися, і підтримує інкрементальне збирання. Вона визначає, які частини було змінено, і виконує лише ті завдання, які залежать від цих частин.

Основні плагіни призначені для розробки та розгортання Java, Groovy та Scala додатків. На відміну від Apache Maven, заснованого на концепції життєвого циклу проекту, і Apache Ant, у якому порядок виконання завдань (targets) визначається відносинами залежності (depends-on), Gradle використовує направлений ациклічний граф для визначення порядку виконання завдань.

До проекту було підключено такі бібліотеки:

– data-jpa – ORM-бібліотека для роботи з базою даних;
– security – для налаштування захисту системи;
– web – для реалізації REST-точок;
– thymeleaf – для використання патерну MVC для побудови графічного інтерфейсу.

## 3.2 Опис фізичної моделі бази даних

У якості бази даних було обрано реляційну СУБД MySQL. Для роботи з базою даних було обрано Spring JPA, який використовує драйвер JDBC. Для простішої обробки даних було обрано ORM-бібліотеку data-jpa, оскільки система зберігає велику кількість інформації в базі даних. ORM — цетехнологія програмування, яка зв'язує бази даних із концепціями об'єктно- орієнтованих мов програмування, таким чином створюючи «віртуальну об'єктнубазу даних».

У першій базі даних зберігається інформація про продукт, магазин і товар. У другій базі даних зберігається інформація про ролі, користувачів, категорії, продукти, типи страви, страви, інгредієнти, дієти, персональні дієти, розклад, персональний розклад і заміри.

У фізичних моделях баз даних є такі типи відносин між таблицями: «один до багатьох», «багато до багатьох». У разі відносин «багато до багатьох» створюється проміжна таблиця. Також у фізичних моделях баз даних указані первинні та зовнішні ключи для зв'язку з іншими таблицями. База даних нормалізована до третьої нормальної форми.

Фізичну модель першої бази даних можна побачити на рисунку 3.1.


Рисунок 3.1 – Фізична модель першої БД

З приведеної вище фізичної моделі баз даних можна побачити, що усі атрибути в таблицях взаємонезалежні та повністю залежать від первинного ключа.

Фізичну модель другої бази даних можна побачити на рисунку 3.2, що приведений нижче.

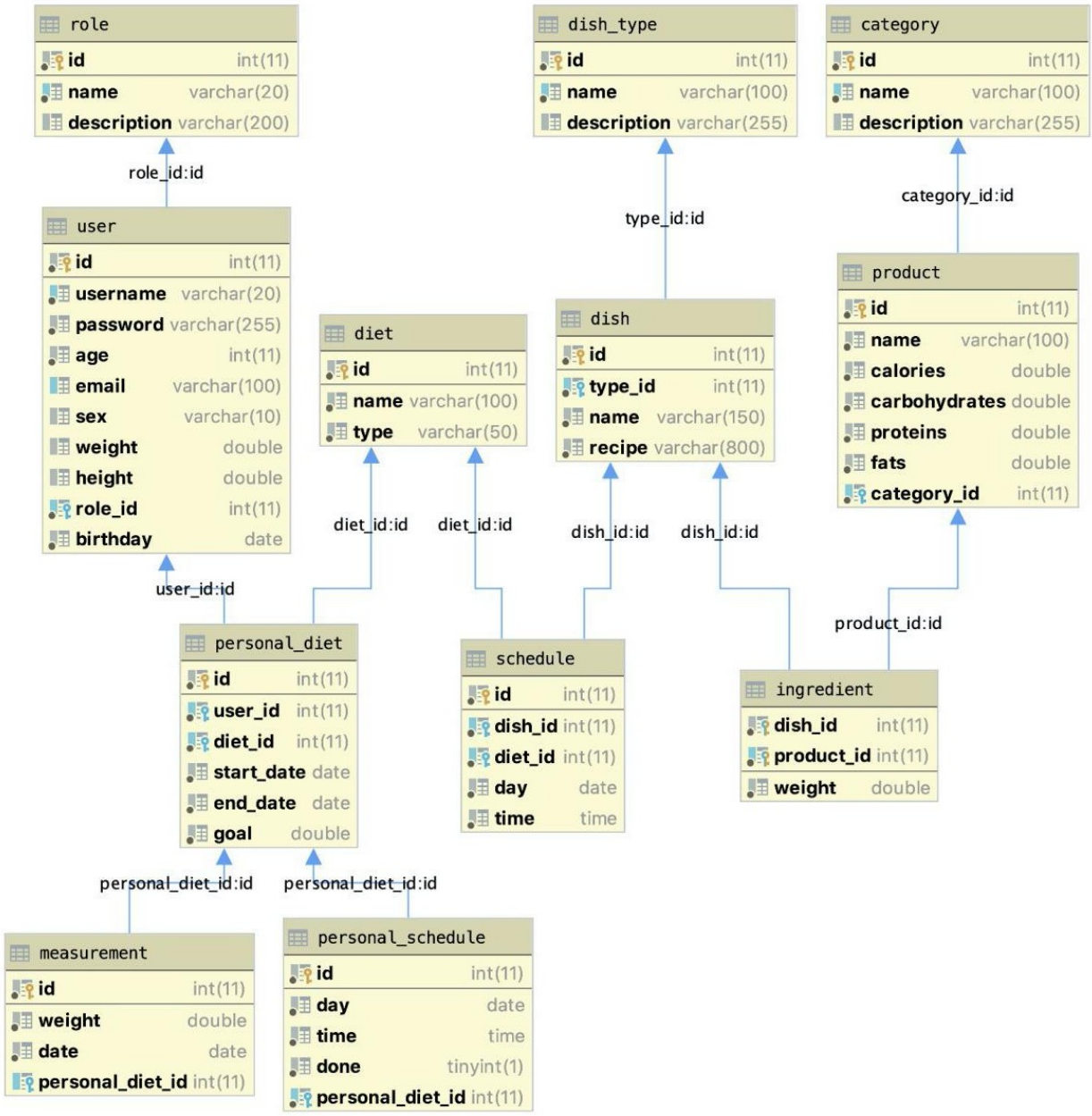


Рисунок 3.2 – Фізична модель другої БД

З приведеної вище фізичної моделі баз даних можна побачити, що усі атрибути в таблицях взаємонезалежні та повністю залежать від первинного ключа.

## 3.3 Опис серверної частини

Було створено POJO-модель StoreProduct, яка зберігає інформацію про магазин і продукт (див. рис. 3.3).

```
@Getter
@AllArgsConstructor
public class StoreProduct {
    private int storeId;
    private long storeDistance;
    private int productId;
    private String productName;
    private double price;
    private double amount;
    private double requiredAmount;
    private LocalTime storeStartTime;
    private LocalTime storeEndTime;
}
```

Рисунок 3.3 – POJO-модель StoreProduct

Було реалізовано інтерфейс сервісу StoreAssignService, який взаємодіє з моделлю StoreProduct (див. рис. 3.4).

```
public interface StoreAssignService {

    /**
     * Призначає продукт до магазину, де це оптимально.
     *
     * @param matrix - вхідна матриця, де рядочки є магазинами, а колонки — продуктами.
     * @param k - мінімальна кількість продуктів, за якою клієнт готовий піти в магазин.
     * @return повертає мапу з призначеннями.
     */
    Map<Integer, List<Integer>> calculateProductStore(StoreProduct[][] matrix, int k);
}
```

Рисунок 3.4 – Інтерфейс сервісу StoreAssignService

Інтерфейс використовує інформацію з цієї моделі для призначення, в якому магазину слід придбати які продукти.

На основі інтерфейсу сервісу StoreAssignService було розроблено реалізацію алгоритму про призначення за допомогою угорського метода. На рисунку 3.5 зображено фрагмент реалізації інтерфейсу StoreAssignService.

```
@Override
public Map<Integer, List<Integer>> calculateProductStore(StoreProduct[][] matrix, int k) {
    //вказують позицію відміченого нуля зі штрихом в рядочках
    var strokeRows = new double[matrix.length];
    //вказують позицію відміченого нуля зі штрихом в колонках
    var strokeCols = new double[matrix[0].length];
    //вказує рядочки, які є призначеними
    var coveredRows = new double[matrix.length];
    //вказує колонки, які є призначеними
    var coveredCols = new double[matrix[0].length];
    //зберігає значення, які відмічені як 0*
    var zeroStartRows = new double[matrix.length];

    Arrays.fill(zeroStartRows, UNMARKED_VALUE);
    Arrays.fill(strokeRows, UNMARKED_VALUE);
    Arrays.fill(strokeCols, UNMARKED_VALUE);

    //вираховуємо коефіцієнти, за якими буде робитися призначення
    var coefficientsMatrix = calculateCoefficients(matrix);

    //крок 1
    reduceMatrix(coefficientsMatrix);
    //крок 2
    markIndependentZeros(coefficientsMatrix, strokeRows, strokeCols);
    //крок 3
    coverColumnsWithMarkedZero(strokeCols, coveredCols);

    //перевіряє, чи ми знайшли оптимальне рішення
    while (!allColumnsAreCovered(coveredCols, k)) {
        //крок 4
        double[] mainZero = findZeroPosition(coefficientsMatrix, coveredRows, coveredCols, zeroStartRows);
        while (mainZero == null) {…
        if (strokeRows[(int) mainZero[0]] == UNMARKED_VALUE) {…
        } else {…
    }

    //призначаємо продукт до магазину
    var result = new HashMap<Integer, List<Integer>>(strokeCols.length);
    for (int i = 0; i < strokeCols.length; i++) {…

    return result;
}
```

Рисунок 3.5 – Фрагмент реалізації класу HungarianStoreAssignService угорським методом

На основі інтерфейсу сервісу StoreAssignService було розроблено реалізацію алгоритму про призначення за допомогою жадібного алгоритму.

На рисунку 3.6 зображено фрагмент реалізації інтерфейсу calculateProductStore жадібним алгоритмом.

```
@Override
public Map<Integer, List<Integer>> calculateProductStore(StoreProduct[][] matrix, int k) {
    //коефіцієнт між магазином та продуктом
    double[][] coefficients = new double[matrix.length][matrix[0].length];
    //індекс є продуктом, а значенням – пара індекс магазину і коефіцієнт
    var assignments = makeFirstAssignment(matrix, coefficients);

    //пошук магазинів, яким призначенно кількість продуктів менше k
    correctAssignments(assignments, coefficients, k);

    //підготовлюємо результуючу мапу
    var result = new HashMap<Integer, List<Integer>>(matrix.length);
    for (int i = 0; i < assignments.size(); i++) {
        var storeInd = assignments.get(i).getFirst();
        var productId = matrix[storeInd][i].getProductId();
        var storeId = matrix[storeInd][i].getStoreId();
        if (result.containsKey(storeId)) {
            result.get(storeId).add(productId);
        } else {
            result.put(storeId, new ArrayList<>() {{
                add(productId);
            }});
        }
    }

    return result;
}
```

Рисунок 3.6 – Фрагмент реалізації класу GreedyStoreAssignService жадібним алгоритмом

Таким чином було реалізовано два підходи для рішення задачі про призначення, в якому магазині слід придбати які продукти, які було інтегровано в систему віртуального дієтолога.

## 3.4 Опис клієнтської частини

Перший інтерфейс, який побачить користувач, – інтерфейс сторінки реєстрації користувачів (див. рис. 3.7).

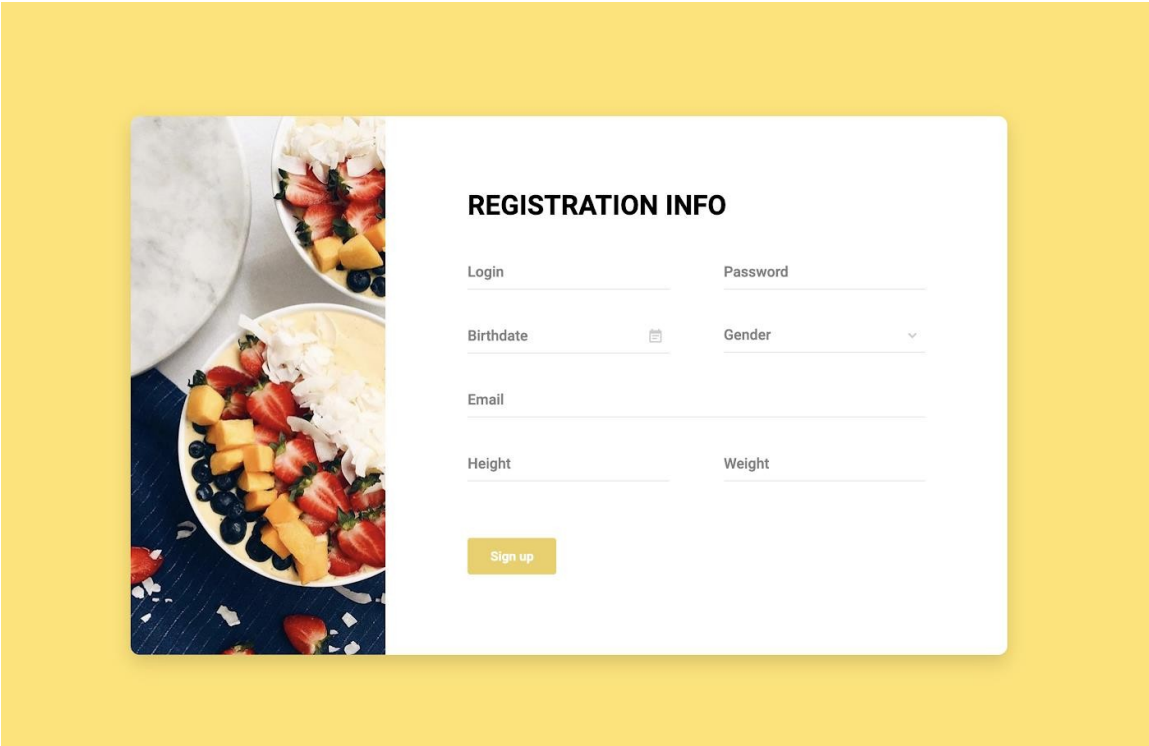


Рисунок 3.7 – Інтерфейс сторінки реєстрації

На сторінці присутні такі поля: логін, пароль, дата народження, стать, електронна пошта, зріст і вага.

Також на сторінці присутня кнопка «Sign up» для реєстрації користувача в системі.

Усі поля мають підказку, яку саме інформацію потрібно ввести. При неправильному заповненню даних поля підсвітяться червоним і з’явиться повідомлення про необхідність змінити формат введених даних або перевірити, чи всі поля заповнені.

Наступний інтерфейс, який побачить користувач після реєстрації – інтерфейс головної сторінки (див. рис. 3.8).

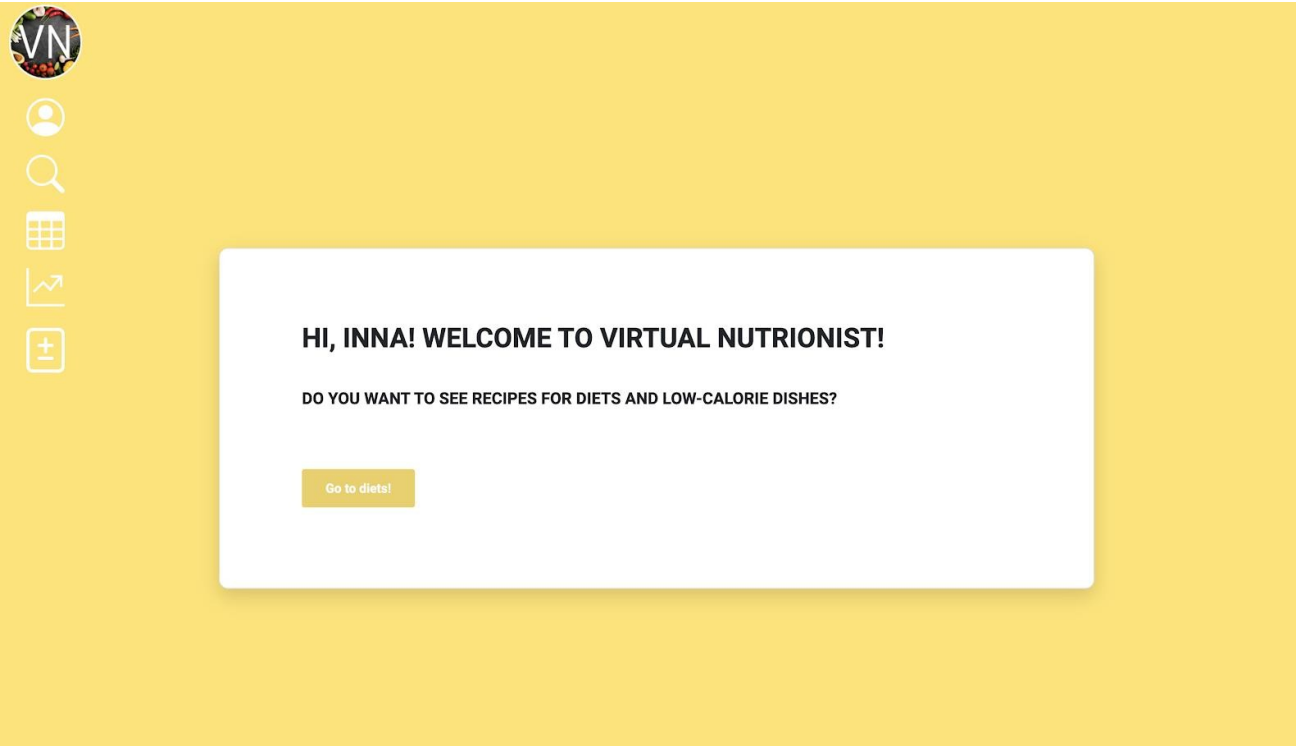


Рисунок 3.8 – Інтерфейс головної сторінки

На сторінці присутнє меню, за яким користувач може перейти до сторінки профілю користувача, сторінки пошуку, сторінки розкладу користувача та калькулятору калорій. Також присутня кнопка для переходу на сторінку з рецептами та дієтами.

Наступний інтерфейс – сторінка розкладу користувача, на яку він може перейти з головного меню (див. рис. 3.9). На цій сторінці користувач зможе побачити інформацію, як він щоденно фіксує дотримання дієти.

Сторінка розкладу користувача містить щоденні прийоми їжі користувача та перекуси, поряд з якими розташовані чексбокси для відмічання, чи був дотриманий прийом їжі. На сторінці можна зазначити час прийому їжі та обрати фізичні справи, які були виконані протягом дня.

Також є поле для фіксування ваги. За цими даними є можливість розрахувати прогнозовану вагу на наступний день і побачити її. Разом із цією інформацією зображується кількість днів до досягнення мети.

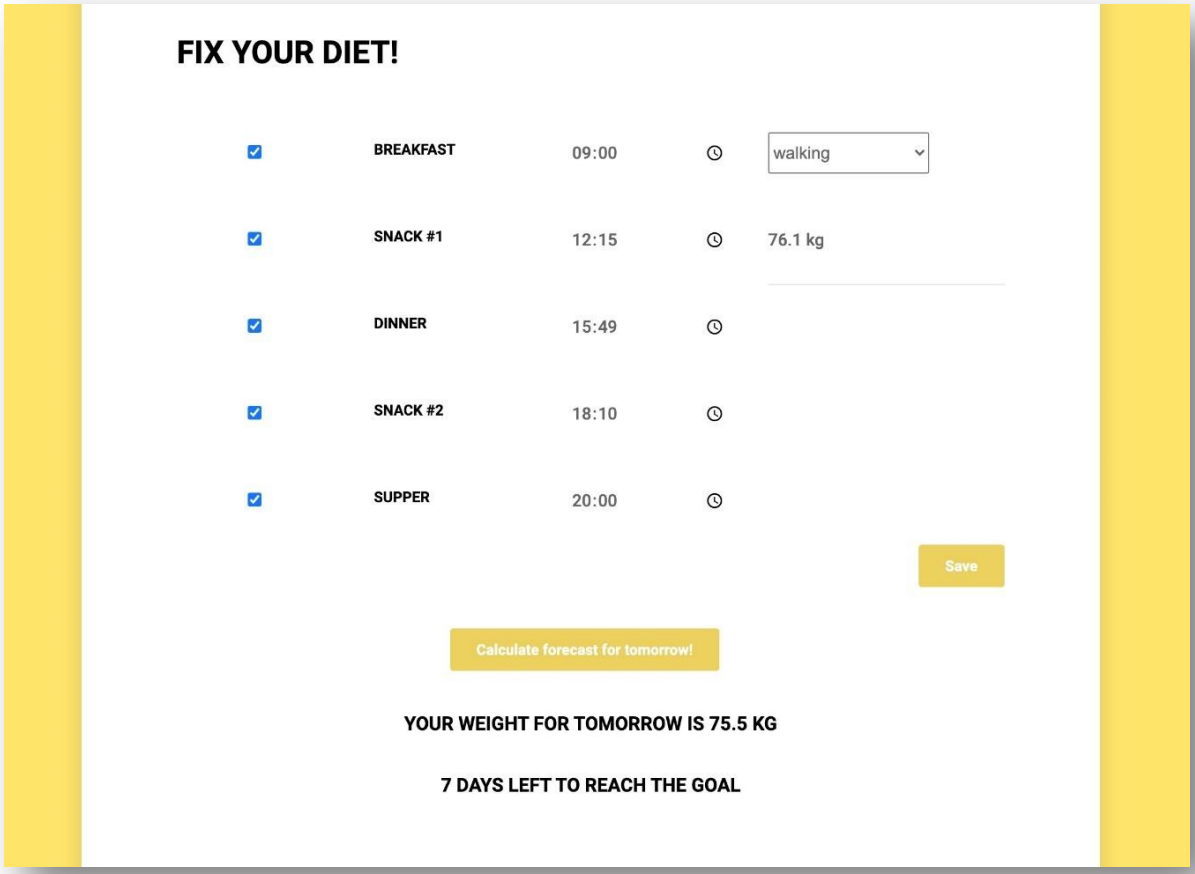


Рисунок 3.9 – Інтерфейс розкладу користувача

Наступний інтерфейс – сторінка з зображенням інформації про дієту (див. рис. 3.10). На сторінці можна побачити інформацію про дієту та про продукти, з яких вона складається. Інформація про продукти для дієти на сторінці зображена для тижня дотримання дієти.

На сторінці присутнє поле, в яке користувач може ввести мінімальну кількість товару, за якою він згоден піти в той чи інший магазин, аби не обходити всі магазини, в яких є найдешевші товари.

Також на сторінці присутня можливість обрати алгоритм, за допомогою

якого буде розраховано призначення, в якому магазині слід купити які продукти. Можна порахувати призначення за допомогою угорського метода та жадібного алгоритму. Також присутня кнопка розрахувати вартість товарів для обраної дієти у призначених магазинах.

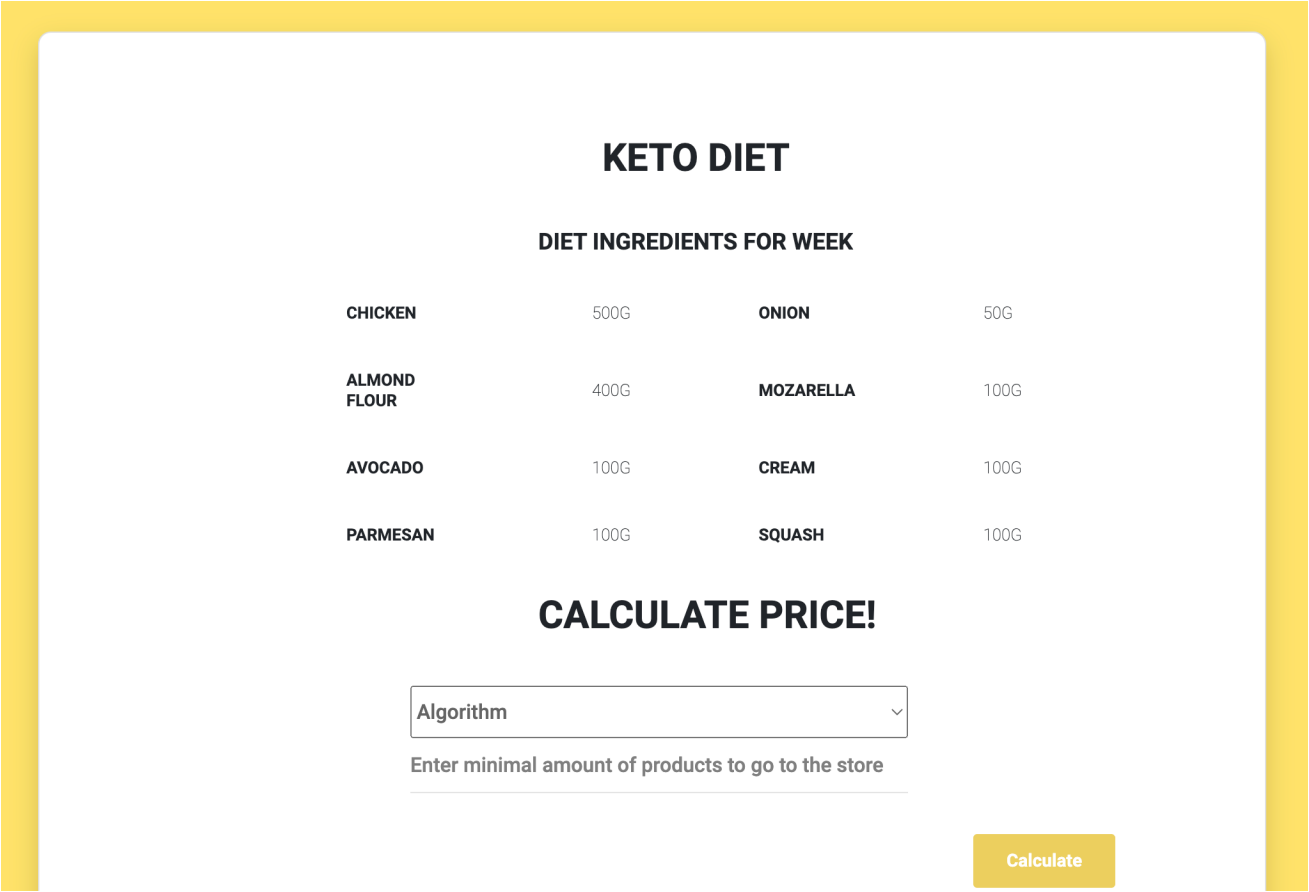


Рисунок 3.10 – Інтерфейс дієти

Наступний інтерфейс – сторінка з зображенням інформації про дієту з результатами роботи алгоритму призначення, в якому магазині слід купити продукти для цієї дієти (див. рис. 3.11).

На сторінці є можливість побачити, в яких магазинах буде дешевше придбати продукти для цієї дієти, тобто можна побачити результат роботи методів вирішення задачі про призначення, в якому магазині слід придбати який продукт.

Разом із цим на сторінці можна побачити сумарну вартість продуктів для

дієти та відстань до найближчого магазину, в якому вартість продуктів для цієї дієти буде найменшою.

Також можна побачити час роботи магазинів і інгредієнти, які можна придбати в цьому магазині.

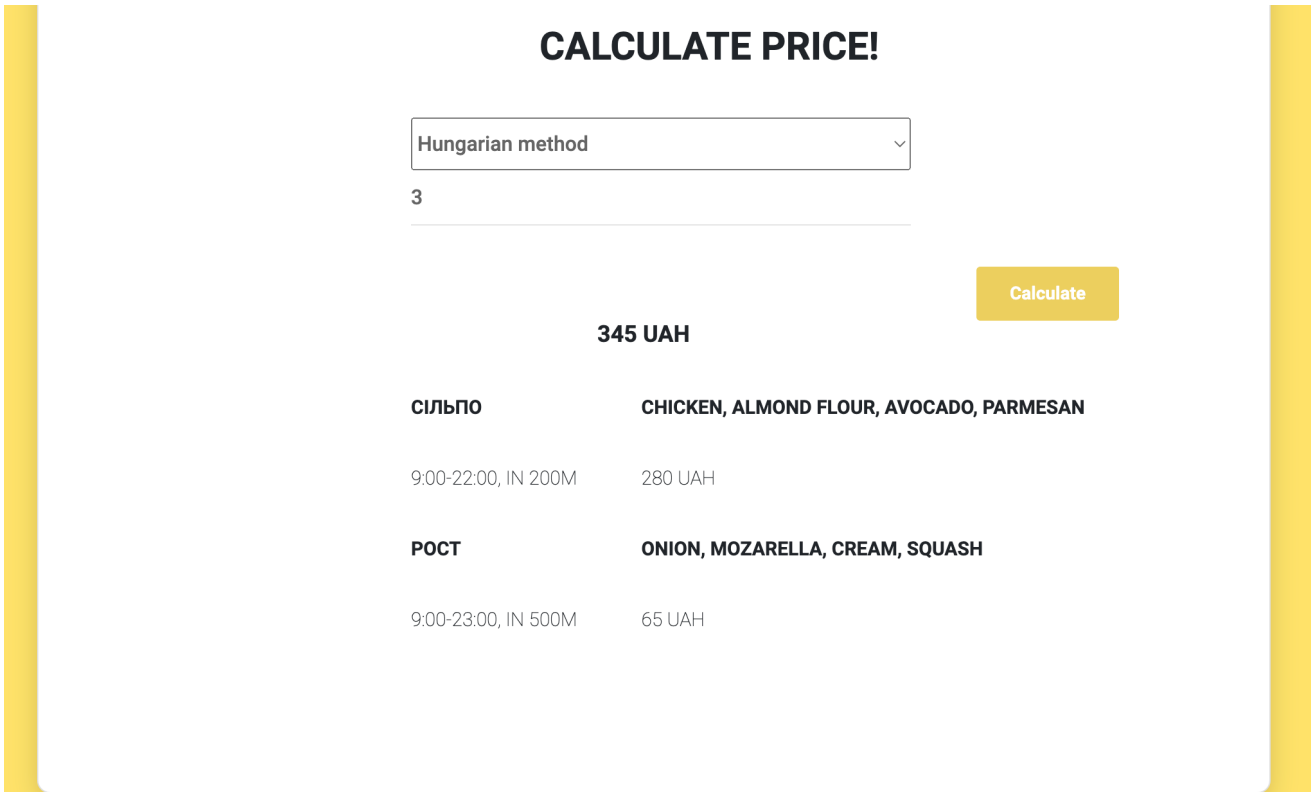


Рисунок 3.11 – Інтерфейс дієти з виконанням алгоритму призначення

На сторінці також присутня інформація про вартість призначених продуктів у магазинах.

# 4 ОПИС ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

## 4.1 Планування експерименту

Для того, щоб перевірити, наскільки ефективними є реалізовані методи, було сплановано серію експериментів.

Перша серія складається з:

– вирішення задачі про призначення, в якому магазині слід придбати який продукт для дієти, жадібним алгоритмом (G1, G2, G3, G4, G5);

– вирішення задачі про призначення, в якому магазині слід придбати який продукт для дієти, угорським методом (H1, H2, H3, H4, H5).

Критерії перевірки:

– час роботи алгоритму;

– використання пам’яті;

– значення цільової функції.

Експерименти необхідно проводити з різною кількістю продуктів і магазинів, наприклад:

– 5 магазинів, 5 продуктів;

– 5 магазинів, 15 продуктів;

– 5 магазинів, 25 продуктів;

– 7 магазинів, 35 продуктів;

– 10 магазинів, 50 продуктів.

Експерименти будуть проведені на одному сервері з наступними характеристиками:

– процесор Dual-Core Intel Core i5;

– частота процесору 2.3 GHz;

– оперативна пам’ять 8 ГБ.

## 4.2 Результати проведення експерименту

Було проведено серію експериментів для порівняння угорського методу та жадібного алгоритму, засновуючись на середньому часі виконання алгоритмів і на середнім використанні пам’яті. Результати експериментів занесені в таблиці 4.1, 4.2, 4.3, 4.4, 4.5.

Таблиця 4.1 – Результати першого експерименту

| Алгоритм/Показники | Середній час роботи алгоритму, мс | Середнє використання пам’яті, Кб | Значення цільової функції, грн |
|---|---|---|---|
| Жадібний алгоритм, G1 | 20 | 281 | 109 |
| Угорський алгоритм, H1 | 54 | 506 | 107 |

Таблиця 4.2 – Результати другого експерименту

| Алгоритм/Показники | Середній час роботи алгоритму, мс | Середнє використання пам’яті, Кб | Значення цільової функції, грн |
|---|---|---|---|
| Жадібний алгоритм, G2 | 27 | 313 | 334 |
| Угорський алгоритм, H2 | 71 | 594 | 322 |

Таблиця 4.3 – Результати третього експерименту

| Алгоритм/Показники | Середній час роботи алгоритму, мс | Середнє використання пам’яті, Кб | Значення цільової функції, грн |
|---|---|---|---|
| Жадібний алгоритм, G3 | 32 | 342 | 585 |
| Угорський алгоритм, H3 | 101 | 616 | 551 |

Таблиця 4.4 – Результати четвертого експерименту

| Алгоритм/Показники | Середній час роботи алгоритму, мс | Середнє використання пам’яті, Кб | Значення цільової функції, грн |
|---|---|---|---|
| Жадібний алгоритм, G4 | 40 | 370 | 837 |
| Угорський алгоритм, H4 | 139 | 703 | 725 |

Таблиця 4.5 – Результати п’ятого експерименту

| Алгоритм/Показники | Середній час роботи алгоритму, мс | Середнє використання пам’яті, Кб | Значення цільової функції, грн |
|---|---|---|---|
| Жадібний алгоритм, G5 | 56 | 391 | 1176 |
| Угорський алгоритм, H5 | 182 | 724 | 970 |

Із результатів експериментів можна побачити, що середній час роботи алгоритму та середнє використання пам’яті залежать від кількості магазинів і продуктів, тобто від кількості параметрів в моделі.

Найбільш ефективним по часові та використанні пам’яті виявився жадібний алгоритм, але мінімальна вартість товару була не найменшою.

Угорський алгоритм працював повільніше та використовував більше пам’яті, але мінімальна вартість товару завжди виходила менше, ніж у жадібному алгоритмі.

## 4.3 Висновки та рекомендації з експериментального дослідження

За результатами проведення експериментів можна побачити, що жадібний алгоритм вираховує призначення доволі швидко у порівнянні з угорським методом. Також помітно, що жадібний алгоритм використовує менше пам’яті. Але мінімальна вартість продуктів є меншою при використанні угорського алгоритму.

Можна порекомендувати використовувати жадібний алгоритм для вирішення задачі про призначення, в якому магазині слід придбати які продукти, якщо користувачу є критичним обсяг використання пам’яті алгоритмом, оскільки час виконання та використання пам’яті є меншим за угорський.

Можна порекомендувати використовувати угорський алгоритм, якщо життєво важливою є значення цільової функції, наприклад, користувачу важливіше зекономити гроші на покупці продуктів, ніж мати менший час виконання алгоритму та менші витрати пам’яті.

# ВИСНОВКИ

У результаті кваліфікаційної роботи магістра було проаналізовано та обрано методи для вирішення оптимізаційних задач про призначення для створення системи віртуального дієтолога.

Для реалізації та проведення експериментів було обрано два методи: угорський метод і жадібний алгоритм. Реалізація усіх методів знаходиться на серверній частині системи.

Було проведено планування експериментального дослідження методів: визначено передумови для експериментів, критерії для порівняння, кількість експериментів.

У ході роботи були виконані наступні етапи:

– проведено аналіз та моделювання предметної області роботи дієтології;

– побудовано математичну модель оптимізаційної задачі в області дієтології;

– проведено аналіз існуючих методів вирішення оптимізаційних задач про призначення та обрано кращі для подальшого дослідження;

– розроблено схему бази даних для зберігання даних моделей;

– розроблено угорський алгоритм вирішення оптимізаційної задачі про призначення та його реалізація;

– було проведено експериментальне дослідження розроблених методів.

Результати роботи були опубліковані на дванадцятій міжнародній науково-технічній конференції та на конференції «Міжнародний молодіжний форум».

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ярка У. Б., Білущак Т. М. Інформатика та комп'ютерна техніка: навч. посіб. – Львів, 2015. – 199 с.

2. Зубар Н. Основи фізіології та гігієни харчування, 2017. – 265 с.

3. Бернерс-Лі Т. Сплітаючи павутиння: витоки та майбутнє Всесвітньої мережі, 2007. – 176 с.

4. Акулич И. Л. Математическое программирование в примерах и задачах – Москва, 1996.

5. O. Mazurova, O. Samantsov, O. Topchii, M. Shirokopetleva A Study of Optimization Models for Creation of Artificial Intelligence for the Computer Game in the Tower Defense Genre, 2020 IEEE International Conference on Problems of Infocommunications.Science and Technology (PIC S&T), 2020, pp. 491-496.

6. Васкевич Д. Стратегии клиент/сервер – Киев, 1997. – 360 с.

7. Кузнецов С. Д. Основы баз данных – Москва: БИНОМ. Лаборатория знаний, 2007. – 484 с.

8. Дейт С. Д. Введення до баз даних, 2010. – 164 с.

9. Васвани В. MySQL: использование и администрирование = MySQL Database Usage & Administration – Москва: Питер, 2011. – 368 с.

10. Введение в Spring Boot: создание простого REST API на Java / Хабр. URL: https://habr.com/ru/post/435144/ (дата звернення: 03.03.2022).

11. Thymeleaf. URL: https://www.thymeleaf.org/index.html (дата звернення: 05.03.2022).

12. Хассан Г. UML-проектирование систем реального времени параллельных и распределенных приложений – Москва, 2011. – 704 с.

13. G. Shcherbakova, V. Krylov, R. Pisarenko, N. Bilous Signal restoration by means of blind deconvolution based on optimization with wavelet transformation, 2016

3rd International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T), 2017, pp. 21-23.

14. Юдин Д. Б. Задачи и методы линейного программирования: Задачи транспортного типа – Москва, 2010. – 184 с.

15. Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность, 1985.

16. Bruff D. The assignment problem and the Hungarian method, 2010.

17. Банди Б. Основы линейного программирования – Москва, 2011. – 240 с.

18. Ronald R. Optimization in operations research, 1997. – 919 p.

19. Кормен Т., Лейзерсон Ч. Вступ в алгоритми, 2001.

20. Колмагоров А. Н., Фомин С. В. Элементы теорий функций и функционального анализа – Москва, 1976. – 544 с.

21. Фримен А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов – Москва, 2013. – 688 с.

22. Рогачев С. Обобщенный Model-View-Controller, 2007.