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 цілочисленого програмування для системи керування пасікою

Виконала :
студентка 2 курсу групи ІПЗм-20-3
Шаповал А. А.
(прізвище, ініціали)

Спеціальність 121 - Інженерія програмного забезпечення
Тип програми Освітньо-наукова
Керівник доц. Мазурова О. О.
(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 Шаповал Альони Андріївни

(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою

затверджена наказом університету від «_24_»__03__ 2022 р. No 412 Ст

2. Термін здачі студентом закінченої роботи «_18_»__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, опис програмної реалізації, опис експериментальних досліджень, висновки, перелік посилань.*

5. Перелік графічного матеріалу: 33 рисунків, 6 таблиць

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблемної області дослідження	17.01.22 -31.0122	виконано
2	Постановка задачі	31.01.22-07.02.22	виконано
3	Дослідження існуючих оптимізаційних моделей та методів	07.02.22-21.02.22	виконано
4	Розробка оптимізаційних моделей та методів	21.02.22-07.03.22	виконано
5	Розробка бази даних та архітектури	07.03.22-21.04.22	виконано
6	Програмна реалізація	21.04.22-30.04.22	виконано
7	Експериментальне дослідження методів	30.04.22-06.05.22	виконано
8	Підготовка пояснювальної записки	60.05.22-18.05.22	виконано
9	Підготовка презентації та доповіді	13.05.22-18.05.22	виконано
10	Нормоконтроль	18.05.22	виконано
11	Рецензування	20.05.22	виконано
12	Занесення диплома в електронний архів	21.05.22	виконано
13	Попередній захист	21.05.22	виконано
14	Допуск до захисту у зав. кафедри	22.05.22	виконано

Дата видачі завдання 17.01.2022 р.

Керівник доц. _____(Мазурова О. О.)

Завдання прийняла до виконання_____ (Шаповал А. А.)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка містить: 71 с., 33 рис., 6 табл., 26 джерел.

ЖАДІБНИЙ АЛГОРИТМ, МАТЕМАТНИЧНА МОДЕЛЬ, МЕТОД ВИРІШЕННЯ ОПТИМІЗАЦІЙНОЇ ЗАДАЧІ, ПАСІКА, СИМПЛЕКС МЕТОД, МЕТОД ГІЛОК ТА КОРДОНІВ

Об’єктом дослідження є методи вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою.

Мета роботи полягає в теоретичному дослідженні, розробці та практичній реалізації методів вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою та проєктування програмної системи.

Методи розробки базуються на таких технологіях як Java, Angular 11, Spring, RxJS.

У результаті роботи проведено аналіз та вибір методів вирішення оптимізаційних задач цілочисленого програмування для системи догляду та керування пасікою, проведено аналіз та моделювання предметної області, побудовано математичну модель оптимізаційної задачі в області пасіки, розроблено схему бази даних. Програмно реалізовано алгоритми та систему. Проведено експериментальне дослідження обраних методів.

Explanatory note contains: 71 pages, 33 figures, 6 tables, 26 sources.

APIARY, GREEDING ALGORITHM, MATHEMATICAL MODEL, METHODS OF SOLVING OPTIMIZING PROBLEMS, SIMPLEX METHOD, METHOD OF BRANCHES AND BORDERS

The object of research is the methods of solving optimization problems of integer programming for the apiary management system.

The aim of the work is theoretical research, development and practical implementation of methods for solving optimization problems of integer programming for the apiary management system and software system design.

Development methods are based on technologies such as Java, Angular 11, Spring, RxJS.

As a result of the work, the analysis and selection of methods for solving optimization problems of integer programming for the system of care and management of the apiary were done, analysis and modeling of the subject area were done, a mathematical model of the optimization problem in the apiary was built, the database scheme was developed. Algorithms and system were implemented. An experimental study of selected methods was conducted

Умови публікації пояснювальної записки

Я, Шаповал Альона Андріївна, студент групи ІПЗм-20-3, здобувач вищої освіти на другому (магістерському) рівні, кафедра Програмної інженерії, заявляю: моя кваліфікаційна робота на тему «Дослідження методів вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою», 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 8

1 Аналіз проблемної області та постановка задач 10

1.1 Аналіз проблемної області дослідження 10

1.2 Аналіз існуючих аналогів 14

1.3 Постановка задач 15

2 Опис прийнятих проєктних рішень 17

2.1 Аналіз та моделювання предметної області роботи пасіки 17

2.2 Розробка математичної моделі оптимізаційної задачі в області роботи пасіки 19

2.3 Аналіз методів цілочисленого програмування 23

2.4 Проєктування бази даних 30

2.5 Розробка алгоритмів для вирішення оптимізаційних задач цілочисленого програмування для системи догляду за пасікою 31

2.6 Розробка архітектури системи 35

3 Опис програмної реалізації 38

3.1 Вибір засобів програмної реалізації 38

3.2 Опис фізичної моделі бази даних 39

3.3 Реалізація серверної частини 40

3.4 Опис інтерфейсу користувача 43

4 Опис експериментальних досліджень 46

4.1 Планування експерименту 46

4.2 Результати проведення експерименту 47

Висновки 51

Перелік джерел посиланнь 53

Додаток А Перелік джерел посилання за науковими напрямами керівника та науковців кафедри програмної інженерії 56

Додаток Б Звіт результатів перевірки кваліфікаційної роботи на унікальність тексту 57

Додаток В Експертний висновок результатів перевірки кваліфікаційної роботи на відповідність оформлення вимогам ДСТУ 3008:2015 57

Додаток Г Слайди презентації 60

Додаток Д Тези доповіді 70

Додаток Е Тези доповіді на молодіжному форумі «Радіоелекторніка та молодь у XXI ст» 71

ВСТУП

В Україні існує велика кількість сфер сільського господарства, серед яких бджільництво є однією із найприбутковіших. Проте, воно потребує великої кількості як матеріальних, так і нематеріальних ресурсів. До недавнього часу Україна входила в трійку експортерів меду в світі. На жаль, в останні роки прибуток від бджільництва систематично зменшується. Так, за останні два роки показник експортованого меду знизився до 49,4 тис. тон. Такий різкий спад був зумовлений рядом об'єктивних чинників, а саме масовим отруєнням бджіл, спричиненим безвідповідальним розпилюванням отруйних гербіцидів, появою ряду нових та мутуючих існуючих інфекційних хвороб бджолосімей, несприятливими погодними умовами тощо.

Зазначені проблеми доповнюються викликами логістичного та науково обґрунтованого характеру, а саме необхідністю організації збору сортового меду, плануванням та забезпеченням правильного розміщення необхідної кількості бджолосімей на полях з різним агрокультурами чи лісосмугами, організацією викачки меду, його доставки до пунктів збору та продажу тощо.

Не менш важливим є правильний розподіл ресурсів для отримання максимального прибутку з пасіки, що передбачає отримання різних товарних позицій з кожного вулика, а саме мед, віск, стільники з медом тощо. Усі описані вище задачі можна оптимізувати з метою полегшення роботи фермера та збільшення прогнозованого прибутку.

Метою даної роботи є дослідження методів вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою, які забезпечать можливість надання пасічникам необхідних рекомендацій щодо вирішення проблем логістичного характеру.

В ході кваліфікаційної роботи магістра :

– проведено аналіз існуючих методів вирішення оптимізаційних задач цілочисленого програмування та обрано кращі для подальшого дослідження;

– проведено аналіз та моделювання предметної області роботи пасіки;

– побудовано математичну модель оптимізаційної задачі в області роботи пасіки;

– розроблено схему бази даних для зберігання даних моделей;

– розроблено алгоритми обраних методів вирішення оптимізаційних задач та їх реалізація;

– сплановано та проведено експериментальне порівняння методів.

Під час виконання магістерської кваліфікаційної роботи був проведений аналіз проблемної області дослідження та реалізації методів цілочисленого програмування на основі публікацій світових та вітчизняних фахівців в проблемній області (див. додаток А), розглянуті основні існуючі методи.

Результатом роботи стала система, яка складається з трьох частин : серверу бази даних, серверу за стосунку та вебсайту. На сервері реалізовано методи вирішення оптимізаційних задач.

Робота пройшла перевірку на академічну доброчесність (див. додаток Б) та перевірку норм оконтролю (див. додаток В).

За результатами кваліфікаційної роботи магістра створено презентацію (див. додаток Г). За результатами дослідження опубліковано тези доповідей на дванадцятій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (див. додаток Д) та на Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті» (див. додаток Е).

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Аналіз проблемної області дослідження

На сучасному етапі свого розвитку Україна позиціонує себе як аграрна країна. Велика увага уряду та профільного міністерства приділяється розвитку різноманітних сфер сільського господарства, проте бджільництво є однією з найвагоміших його частин, оскільки забезпечує не лише безпосереднє отримання продуктів бджільництва, а й опилення злакових культур, вирощування яких становить вагому частку дохідної частину бюджету держави Україна. Крім того, бджільництво є однією з найбільш прибуткових сфер фермерства, тож постійно викликає зацікавленість все більшої кількості бажаючих опанувати цей вид підприємництва [1]. Кожний бізнес вимагає певних базових знань, технологізації та капіталовкладень. Прибуткове бджільництво не є винятком та вимагає запровадження певного ряду процесів, які б сприяли зростанню видобутку кінцевої продукції та збільшенню прибутків. На даний час бджільництво може стати прибутковим лише за умови постійного його розвитку та запровадження сучасних новітніх технологій у роботу пасіки. Так, підтримування необхідної температури, вологості та певного рівня кисню у вулику, слідкування за роїнням бджіл, догляд за станом бджіл, збір продуктів бджільництва тощо є тими виробничими процесами, які можуть та мають бути оптимізованими та технологізованими [2]. Технологізації вимагають процеси перевезення вуликів на різні локації відповідно до термінів цвітіння різноманітних сортів медоносних рослин, забір меду та інших продуктів бджільництва, як то віск, стільники з медом, пилок, бджолині матки та ін., завантаження та розвантаження транспортних засобів, розрахунок максимально вигідної траєкторії пересування транспортних засобів, оптимізації витрат пального тощо [3]. Оскільки така продукція має різну цінову категорію в залежності від періоду та місця збору, процес їх сортування та пакування вимагає попереднього планування та подальшого контролю.

Для вирішення перелічених завдань існують різні задачі оптимізації. Оптимізаційна задача – це задача знаходження екстремуму цільової функції, тобто максимуму чи мінімуму, які мають лінійні або нелінійні обмеження [4]. Функція F(x) називається цільовою функцією. Цільова функція – це функція, що зв'язує мету з керованими змінними в задачі оптимізації. Серед оптимізаційних задач виділяють задачі мінімізації та задачі максимізації, як показано у формулах (1.1) та (1.2). Вони записуються у наступному вигляді:

$$F(x) \to min, x \in X, \quad (1.1)$$
$$F(x) \to max, x \in X, \quad (1.2)$$

де F(x) це цільова функція, а X – допустима множина, будь-який елемент $x \in X$ – допустима точка цієї множини. Елементи множини X за своїм характером поділяють на безперервні, дискретні та цілочисельні. У випадку, коли елемент (змінна) приймає нові значення, він буде безперервним, приймаючи тільки значення цілих чисел, буде цілочисельним елементом (змінною), а у випадку, коли він може приймати тільки певні значення, буде називатися дискретним. Залежності між змінними в цільовій функції можуть буди лінійними або нелелійними. Прикладом лінійної залежності про n невідомих може бути формула (1.3):

$$F(x) = a_1x_1 + a_2x_2 + \cdots + a_nx_n, \quad (1.3)$$

У даному випадку залежності знаходяться в першому ступені, тому така залежність і називається лінійною. У інших випадках залежність буде нелінійною.

Для вирішення оптимізаційних задач існують різні методи. Математичне програмування вивчає методи вирішення оптимізаційних задач [5]. У загальному вигляді математичне програмування являє собою повторювану обчислювальну процедуру, яка допомагає знайти екстремуми цільової функції.

Існують наступні методи математичного програмування :

– методи лінійного програмування (використовуються, якщо між змінними у математичній моделі існують лінійні залежності);

– методи нелінійного програмування (використовуються, якщо між змінними у математичній моделі існують нелінійні залежності);

– методи цілочисельного або дискретного програмування (використовуються, якщо деякі, а може й усі змінні у математичній моделі є цілочисельними або дискретними);

– методи стохастичного програмування (використовуються у разі, якщо вихідні дані чи його частина є випадковими величинами);

– математичний апарат теорії ігор (використовується, якщо вхідна інформація є невизначеною).

Для вирішення задачі оптимізації продажу та виготовлення товару ми обрали методи оптимізації цілочисельного програмування [6].

Забезпечення можливості використання пасічниками у своїй роботі оптимізаційних методів вимагає їх програмної реалізації.

Першим етапом проєктування системи є обрання архітектури. Архітектура системи – це сукупність рішень щодо організації програмної системи. Критеріями хорошої архітектури є ефективність та гнучкість системи, масштабованість процесу розробки та можливість її розширення тощо.

Існують такі види архітектурних шаблонів :

– шаблон «Модель-Представлення-Контролер» – ця система поділяється на модель, яка зберігає дані; представлення, яке представляє дані та взаємодіє з користувачем; контролера, який є посередником між представленням та моделлю. Використання такого розподілу має досить суттєвий недолік, оскільки система втрачає свою продуктивність;

– шаблон посередника – у даній системі наявна велика кількість модулів, для зв’язування та забезпечення роботи яких має бути посередник. Фактор необхідності наявності посередника також зумовлює певні недоліки, а саме зниження продуктивності та, за відсутності посередника, зупинку роботи;

– багатошаровий шаблон – система поділяється на декілька рівнів, кожен з яких знаходиться один під одним. Поточний рівень може визивати лише

попередній рівень. Ця архітектура також має ряд недоліків : зниження продуктивності системи та складність її реалізації;

– шаблон «клієнт-сервер» дозволяє розподілити навантаження між клієнтською та серверною частинами. Перевагами цього шаблону є те, що всі обчислення знаходяться на сервері, що дозволяє полегшити роботу клієнтським частинам. Використання цієї архітектури дозволяє зробити зберігання даних більш захищеним, адже усі вони знаходяться на сервері. Ця система здатна до масштабування [7].

Враховуючи усі вищезазначені характеристики архітектурних шаблонів, було визначено, що наші вимоги задовольняє «клієнт-серверна» архітектура.

Даний програмний продукт потребує зберігання великої кількості даних, тому для задоволення цих потреб доцільним є використання бази даних.

Існують такі види бази даних :

– ієрархічна – представляє базу даних у вигляді дерева (ієрархічної) структури, що складається з об’єктів (даних) різних рівнів. Структура запис-нащадок повинна мати в точності одного предка;

– мережева – є розширенням ієрархічного підходу. В мережевій структурі даних у нащадка може бути будь-яке число предків;

– реляційна – дані в базі даних є набором відносин. Відносини (таблиці) відповідають певним умовам цілісності. Дані в реляційній моделі є незалежними;

– об’єктна й об’єктно-орієнтована – дані в таких базах являють собою об’єкти з певними наборами властивостей, методів і поведінки. Відносини даних об’єктів будуються на основі узагальнення властивостей, методів і поведінки різних об’єктів по відношенню один до одного та інші.

Оскільки саме у реляційній моделі дані зберігаються у простому для використання та розуміння форматі із забезпеченням їх незалежності, ми обрали її для нашої системи.

Проєктування реляційних баз даних поділяється на три етапи : концептуальне, логічне, та фізичне. Під час концептуального проєктування

вивчається та описується предметна область. На етапі логічного проєктування відбувається розроблення структури зберігання даних, методів доступу й логічної структури системи баз даних без прив’язки до конкретної СУБД. На етапі фізичного моделювання обирається СУБД. Для цього необхідно провести аналіз усіх наявних СУБД та обрати ту, яка буде відповідати всім вимогам системи. Крім того, логічна модель розширюється характеристиками щодо типу пристрою для зберігання даних, об’єму пам’яті, способів фізичного зберігання та використання бази даних тощо.

1.2 Аналіз існуючих аналогів

На даний час існує ряд застосунків, які дозволяють управляти пасікою, отримувати дані з вуликів, контролювати процеси роїння та догляду за пасікою, оптимізувати роботу пасічника. На даний момент немає прямих аналогів нашої системи, які б дозволяли вирішувати оптимізаційні задачі планування виготовлення товару на пасіці, тому було проаналізовано окремі функції різних застосунків.

Так Hivemanager Beekeeping App [8] створений, щоб допомогти автоматизувати процеси, пов’язані з продуктивним та ефективним бджільництвом. Доступ до всіх ресурсів в одному місці дозволяє контролювати відповідність вимогам, керувати завданнями, швидше та простіше вводити дані, надаючи доглядачам більше часу для фізичної роботи з вуликами, уникаючи суттєвої частки організаційної роботи. Застосунок не має можливостей, які б дозволяли вирішувати оптимізаційні задачі планування заготовки товару на пасіці.

Ще одним аналогом нашої системи є застосунок Nectar [9]. Він дозволяє слідкувати за здоров’ям бджіл. Принцип роботи цієї системи полягає у наступному : користувач встановлює датчик, який сканує місцевість і передає схематичне зображення пасіки у систему. Застосунок змінює статус вуликів, якщо

там помічається велика смертність бджіл. Ця система акцентує свою роботу на здоров'ї бджіл, проте не менш важливим є керування процесами логістичного характеру, тому відсутність цією функції вважаємо недоліком.

Ще однією системою, яку можна вважати певним аналогом нашої, є InspectNext [10]. Ця програма для бджільництва відстежує погоду та пропонує ідеальний час для огляду колонії. Для роботи потрібно встановити оптимальні умови, вибрати місце розташування вулика, а застосунок покаже, чи є часовий проміжок оптимальним чи недоцільним для огляду бджіл на основі зазначених умов. Недоліком цього застосунку також є відсутність логістичної складової.

Оскільки проаналізовані системи не мають можливості оптимізувати роботу на пасіці для отримання максимального прибутку, то очевидною є необхідність створення такої системи, що доказує доцільність проведення даного дослідження.

1.3 Постановка задач

Метою роботи є дослідження методів вирішення оптимізаційних задач цілочисельного програмування для системи керування пасікою та розробка програмної системи, яка буде пропонувати оптимальний план збору та продажу продуктів бджільництва на основі даних про пасіку.

Архітектура системи буде складатися з трьох компонентів :

– сервер бази даних, який буде зберігати дані та співпрацювати з сервером додатку;

– сервер застосунку, який буде співпрацювати с базою даних, міститиме декілька методів вирішення оптимізаційної задачі та на основі даних з пасіки буде обчислювати оптимальний план для продажу товарів з пасіки;

– клієнт, який буде використовуватися для взаємодії з користувачем. Клієнт буде представлений вебсайтом.

Поставлена мета зумовила необхідність вирішення наступних задач :

– провести аналіз існуючих методів вирішення оптимізаційних задач цілочисленого програмування та обрати кращі для подальшого дослідження;

– провести аналіз та моделювання предметної області роботи пасіки;

– побудувати математичну модель оптимізаційної задачі в області роботи пасіки;

– розробити схему бази даних для зберігання даних моделей;

– розробити алгоритми обраних методів вирішення оптимізаційних задач та їх реалізація;

– розробити програмну систему, яка складатиметься з серверної та частини та веб-сайту;

– спланувати та провести експериментальне порівняння методів.

2 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

2.1 Аналіз та моделювання предметної області роботи пасіки

Створення системи автоматичного контролю за роботою певної предметної області вимагає детального аналізу цієї області та вивчення усіх її компонентів, що вимагають автоматизації. Вивчаючи бджільництво як сферу підприємництва, ми з'ясували, що основною формою бджільницького господарства є пасіка, що містить від 100 до 200 бджолиних сімей. Декілька пасік в одному господарстві, що включає в себе супутні будівлі : майстерні, зимівники, сховища стільників тощо називається бджолиною фермою. Її розмір залежить від багатьох факторів, а саме необхідності використання бджіл для запилення ряду сільськогосподарських культур, наявності достатньої кількості площ з медоносними рослинами, наближеність пасіки до джерел збору меду тощо. Визначивши потребу господарства в бджолиних сім'ях для запилення сільськогосподарських культур, з'ясовують, скільки меду можуть зібрати бджоли на цих культурах та прораховують доцільність та цінову ефективність збільшення кількості бджолиних сімей. З'ясовано, що пасіки поділяються на стаціонарні та пересувні (кочові), які перевозяться від одних сільгоспугідь до інших, згідно з періодами медозбору та запилення сільськогосподарських культур та наявністю квітучих медомістких масивів. Більшість пасік одноосібних господарств є переважно стаціонарними, оскільки перевезення невеликих пасік часто економічно себе не виправдовує, проте укрупнені фермерські господарства з розвиненою логістикою, що передбачає теоретично обґрунтовану розстановку вуликів з бджолами як на стаціонарне розміщення, так і при кочівлі, технологізацією та оптимізацією виробничих процесів, використовують переважно пересувні пасіки.

Наявність великої кількості підробок та медозамінників у сучасній торгівлі робить природний продукт бджільництва затребуваним, унікальним і високо прибутковим товаром. Постійно зростаючий купівельний попит на якісну природню продукцію цієї сфери надає хороші перспективи даному бізнесу. Так,

сфера бджільництва дозволяє підприємцям розвивати свої потужності, спрямовані на :

- збір пилку;
- вирощування воскової молі;
- виробництво прополісу та настоянок на його основі;
- збір маточного молочка;
- вирощування бджоломаток;
- створення племінного матеріалу.

Так, у перший же рік роботи пасіки можна отримати близько 35 кілограмів меду з однієї бджолосім'ї [11].

Отримані дані щодо технології організації роботи пасіки підтвердили попередню гіпотезу щодо необхідності створення системи автоматичного контролю за її основними виробничими процесами, з метою їх максимальної оптимізації.

Розробка будь-якої автоматизованої контролюючої системи вимагає попереднього моделювання заданої предметної області. Оскільки концептуальна модель – це модель предметної області, що складається з переліку взаємопов'язаних понять, що використовуються для опису цієї області, їх властивостей, характеристик та класифікації за типами, ознаками в даній області та законів протікання процесів у ній, та, зважаючи на основні її компоненти – опис об'єктів предметної області та зв'язків між ними, зазначимо, що для заданої предметної галузі основними об'єктами є : вулик, робітник, товар, пасіка.

Об'єкт «вулик» включає в себе наступну інформацію : код вулика, його координати, кількість рамок.

Об'єкт «робітник» включає в себе : код робітника, ім'я та прізвище працівника, його паспортні дані, які отримуються при працевлаштуванні робітника, його електронну пошту та пароль для входу в систему та його роль (адміністратор чи працівник).

Об'єкт «пасіка» містить дані про кількість вуликів.

Об'єкт «товар» містить назву товару, дані про максимальну кількість товару, яку можна отримати з однієї рамки, скільки максимально може обробити пасічник

цього товару за сезон та вартість цього товару на ринку.

Схема взаємодії основних об’єктів системи зображена на рисунку 2.1.

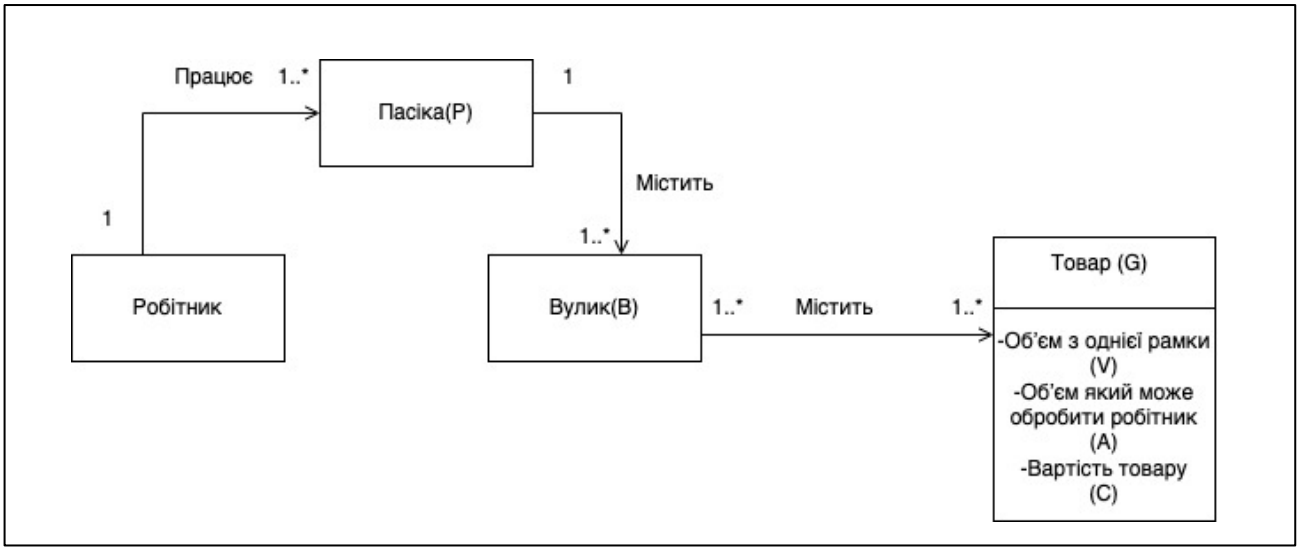

Рисунок 2.1 – Загальна діаграма класів

Користувачами програмної системи є робітники та адміністратори. Адміністратори мають можливість видаляти, змінювати та додавати інформацію про робітника, тож отримують доступ до захищених даних системи.

2.2 Розробка математичної моделі оптимізаційної задачі в області роботи пасіки

Математичне програмування вивчає методи вирішення оптимізаційних задач. У загальному вигляді математичне програмування являє собою повторювану обчислювальну процедуру, яка допомагає знайти екстремуми цільової функції [12].

Існують наступні етапи побудови оптимізаційної моделі :

– постановка проблеми та її аналіз;

– побудова математичної моделі;

- математичний аналіз моделі;
- підготовка вхідної інформації;
- розробка алгоритмів вирішення оптимізаційної задачі;
- порівняння результатів виконання алгоритмів.

На етапі побудови математичної моделі необхідно визначити величини, для яких буде побудована модель, обмеження та цільову функцію, рішення якої необхідно буде знайти.

Сформулюємо опис математичної моделі оптимізаційної задачі для роботи на пасіці. Для отримання прибутку пасічники можуть продавати після завершення кожного сезону певні товари G_i (i=1,n) (мед, віск, стільники з медом тощо). Крім того, пасічнику необхідно залишати певну кількість рамок на зиму, для того, щоб бджоли мали чим харчуватися під час зимівлі. Нехай у пасічника існують P_j (j=1,m) пасік, які розміщенні на різних ділянках. Кожна пасіка містить вулики B_{jk} (k=1,h_j), які у свою чергу містять R_{jk} рамок, в залежності від будови вулика.

Пасічник може виробити лише певну кількість товару A_i, через обмежену кількість робочої сили. Необхідно врахувати те, що з кожної рамки можна отримати лише обмежений об'єм певного товару V_i . Кожен товар можна продати за вартістю C_i за кілограм. Ціна за мед може бути різна в залежності від регіону знаходження пасіки та сезону отримання меду. Так мед, який викачується навесні коштує в 2 рази дорожче ніж мед, який викачується влітку. Нехай Q_{is} – це коефіцієнт, який змінює вартість меду та стільників з медом у залежності від місця перебування пасіки та від сезону викачування меду (де s=1,z), де z – це місце де стоїть пасіка. Дані про кількість сировини на кожній пасіці та прибуток від продажу товарів наведені у таблиці 2.1. Необхідно визначити оптимальний план отримання товару з пасіки для отримання максимального прибутку від реалізації. X_{ij} – це кількість рамок, які необхідно використати для отримання G_i товару з P_j пасіки.

Таблиця 2.1 – Табличне надання оптимізаційної задачі

	Пасіки			Обмеження та параметри за ресурсами		
Продукти	P_1	P_j	P_m	Обсяг видобування з 1 рамки (кг) V_i	Обмеження по обробці продукції (кг) A_i	Вартість на ринку (грн/кг) C_i
G_1(Мед)	X_{11}	X_{1j}	X_{1m}	V_1	A_1	C_1
G_2 (Віск)	X_{21}	X_{2j}	X_{2m}	V_2	A_2	C_2
…	…	…	…	…	…	…
G_i (Стільники с медом)	X_{i1}	X_{ij}	X_{im}	V_i	A_i	C_i
…	…	…	…	…	…	…
G_n (Залишити для зимівлі)	X_{n1}	X_{nj}	X_{nm}	V_n	A_n	
Загальна кількість вуликів на пасіці h_j	h_1	h_j	h_m			

Побудуємо математичну модель для заданої задачі. Мета оптимізації полягає в створенні чіткого оптимального плану вироблення необхідної кількості товару, з метою отримання максимального прибутку з бджолиної ферми, як показано у формулі (2.1):

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{s=1}^{z} k_i Q_{is} X_{ij} V_i C_i \rightarrow max, \quad (2.1)$$

де X_{ij} – кількість рамок, які необхідно використати для отримання G_i товару з P_j пасіки;

V_i – максимальний об’єм товару, який можна отримати з однієї рамки;

C_i – вартість товару на рикну за один кілограм;

Q_{is} – коефіцієнт, який змінює вартість меду та стільників з медом у залежності від місця перебування пасіки та від сезону викачування меду;

k_i – коефіцієнт, який показує відсоток використання рамки під час видобування товару.

Існують наступні обмеження для заданої задачі :

– кількість рамок, які необхідно використати для отримання певного товару з пасіки повинна буди невід’ємним числом, як показано у формулі (2.2):

$$X_{ij} \geq 0,\ \forall i = \overline{1,n}\ .\ \forall j = \overline{1,m}\ , \tag{2.2}$$

– обмежений обсяг можливого видобутку товару.

Сума добутків між кількістю рамок та об’ємом товару з однієї рамки $X_{ij} * Vi$ не повинна перевищувати максимально можливий об’єм обробки товару пасічником за сезон. Також необхідно взяти до уваги стільники з медом, які необхідно залишити для зимування, як показано у формулах (2.3) та (2.4) :

$$\sum_{j=1}^{m} X_{ij} * Vi \leq A_i\ ,\ \forall i = \overline{1,n-1}, \tag{2.3}$$

$$\sum_{j=1}^{m} X_{ij} * Vi \geq A_i\ ,\ \forall i = n, \tag{2.4}$$

– загальна кількість рамок, що необхідно використати для отримання певних товарів з пасіки, не повинна перевищувати загальну кількість рамок на пасіці, як показано у формулі (2.5) :

$$\sum_{i=1}^{n} X_{ij} \leq R_{jk}\ \ \forall j = \overline{1,m}\ \text{, k=}\overline{1,hJ}\ , \tag{2.5}$$

Оскільки кількість рамок не може бути дробовим числом, то введемо обмеження, що X_{ij} – це цілі числа.

2.3 Аналіз методів цілочисленого програмування

Математичне програмування спрямоване на пошук методів знаходження екстремальних значень цільової функції [13].

Відповідно до властивостей цільової функції, задачі математичного програмування поділяються на задачі лінійного або нелінійного програмування. До задач лінійного програмування відносяться задачі, в яких цільова функція та функції обмежені лінійні, якщо ж вони нелінійні, то задача відноситься до задач нелінійного програмування.

Лінійне програмування вирішує оптимізаційні проблеми при обмеженій кількості ресурсів між конкуруючими видами діяльності для того, щоб максимізувати або мінімізувати деякі величини [14].

Використання методів лінійного програмування передбачає дотримування наступних критеріїв :

- задача повинна мати обмежені ресурси;
- необхідно провести точне формулювання цільової функції;
- задача повинна характеризуватися лінійністю;
- ділимість (методи лінійного програмування будуються на припущенні, що результати і ресурси можна поділити на долі).

Зазвичай у задачах лінійного програмування не має обмеження щодо цілочисленості змінних. На практиці ж такі обмеження зустрічаються досить часто.

Задачі цілочисленого програмування є подібними по формулюванню до задач лінійного програмування, але вони відрізняються тим, що шукані змінні повинні бути цілими значеннями [15].

Для вирішення задач цілочисленого програмування існують наступні групи методів:

– методи відсікання;
– комбінаторні методи;
– наближені методи.

Методи відсікання в основі мають методи лінійного програмування для послідовності задач, в які поступово вводяться додаткові обмеження, наприклад симплекс метод.

Комбінаторні ж методи не використовують симплекс методи, вони на початку роботи проводять аналіз вхідних даних і після цього скорочують пошук оптимальних значень.

Для вирішення поставленої перед нами задачі існує велика кількість методів. Розглянемо та порівняємо методи, які найкраще задовольняють наші вимоги.

Симплекс метод – це метод послідовного переходу від одного базисного рішення до іншого до тих пір, поки цільова функція не прийме оптимальне рішення [16]. Для вирішення задачі симплекс методом використовуються симплекс таблиці, які є легкими у розумінні та наочно показують кроки знаходження оптимального рішення. Для початку у симплекс методі знаходиться перший опорний (базисний) план, який потім проходить перевірку на оптимальність, якщо після перевірки план буде не оптимальним, його необхідно буде удосконалити. Для цього знаходяться ведучій стовпчик та рядок, після чого проводяться модифікації з іншими стовпчиками та будується новий опорний план [17]. На даний час існує велика кількість реалізацій симплекс методу, які можуть бути використані для знаходження цільової функції. Симплекс метод не гарантує, що після його обчислення змінні цільової функції будуть цілі, тому необхідно або накладати умову цілочисленості, або використовувати інший метод.

Як було зазначено, необхідною умовою для застосування симплекс методу є знаходження опорного (базисного) плану. Для цього необхідно виконати наступні умови : система повинна мати канонічну структуру, наявні лише обмеження рівняння, перші частини обмежень не є негативними та змінні задачі позитивні. На

жаль, у реальних умовах не завжди можливо досягнути виконання усіх цих вимог. Для вирішення таких задач існує спеціальний метод, який називається методом штучного базису [18]. Цей метод дозволяє в будь-якій задачі лінійного програмування отримати перший опорний план. Для того щоб задовольняти усім вимогам базисного плану вводяться штучні змінні, за рахунок яких досягаються усі вимоги. Головна ідея цього методу полягає у перетворенні штучних змінних із базисних змінних у вільні. Для того, щоб мати змогу використовувати симплекс метод для цілочисельного програмування, необхідно додавати обмеження цілочисельності під час розрахунків.

Жадібний алгоритм заснований на знаходженні найкращого рішення шляхом перебору та знаходженні оптимального рішення [19]. Такий алгоритм характеризується швидкодією та здатністю обробляти велику кількість змінних. У свою чергу, на відміну від симплекс методу, він не має реалізацій методів для конкретних задач та необхідно розробляти його кожен раз під певну задачу, бо жадібний алгоритм жорстко зав’язується на мету задачі, її обмеження та особливості.

Метод Гоморі використовується для вирішення задач цілочисленого програмування.Цей метод відноситься до класу методів відтинання. В його основі покладено поступове звуження області допустимих розв’язків задачі. Пошук цілочисленого оптимуму починається з розв’язування задачі з так званими послабленими обмеженнями, тобто без урахування вимог цілочисленості змінних. Далі введенням у модель спеціальних додаткових обмежень, що враховують цілочисловість змінних, многокутник допустимих розв’язків послабленої задачі поступово зменшуємо доти, доки змінні оптимального розв’язку не набудуть цілочислених значень. Проте практична цінність цього методу є дуже малою, адже для його виконання необхідна велика кількість ітерацій [20].

Метод гілок та кордонів, як і метод Гоморі, використовується для вирішення задач цілочисленого програмування. Він відноситься до комбінаторних методів цілочисленої оптимізації [21]. Починаючи з розв’язування послабленої задачі, він передбачає розбиття початкової задачі на дві підзадачі виключенням областей, що

не мають цілочислових розв’язків, і дослідженням кожної окремої частини многокутника допустимих розв’язків. Цей метод, на відміну від методу Гоморі, у меншій мірі передбачає помилки округлення [22].

Ще одним методом, який відноситься до комбінаторних методів, є метод Фора та Мальгранжа. Цей метод передбачає 2 етапи : пошук початкового плану та його покращення. Спочатку шукається початковий план, а потім за допомогою інтерактивної перебірки планів, знаходиться найкращий варіант. Перед проведенням розрахунків невідомі слід розташувати у порядку спадання коефіцієнтів цільової функції.

Для визначення найкращого методу вирішення оптимізаційної задачі цілочисленого програмування для системи догляду та керування за пасікою порівняємо описані вище методи за допомогою лінійної адитивної згортки з ваговими коефіцієнтами [23].

Лінійна згортка полягає в призначенні коефіцієнтів у лінійному згортку (лінійної комбінації) вихідних критеріїв та подальшої її екстремізації на безлічі допустимих варіантів. Відповідно до цього методу знайдене таким способом рішення вважається «найкращим».

Визначимо критерії за якими будемо проводити порівняння :

– часова складність алгоритму для найгіршого випадку впорядкування даних;

– просторова складність;

– можливість використання наявної реалізації.

У таблиці 2.2 наведено моделювання задачі прийняття рішень у загальному вигляді.

Для простоти порівняння введемо шкали оцінок для деяких критеріїв :

а) можливість використання наявної реалізації :

1) можливість повного використання наявних реалізацій – 3 бали;

2) можливість часткового використання наявних реалізацій – 2 бали;

3) неможливість використання наявних реалізацій – 1 бал.

б) часова складність алгоритму для найгіршого випадку впорядкування даних :

1) константна складність – 7 балів;
2) лінійна складність – 6 балів;
3) логарифмічна складність – 5 балів;
4) логлінійна складність – 4 бали;
5) квадратична складність – 3 бали;
6) кубічна складність – 2 бали;
7) експоненціальна складність – 1 бал.

в) просторова складність :

1) константна складність – 7 балів;
2) лінійна складність – 6 балів;
3) логарифмічна складність – 5 балів;
4) логлінійна складність – 4 бали;
5) квадратична складність – 3 бали;
6) кубічна складність – 2 бали;
7) експоненціальна складність – 1 бал.

Після введення шкали оцінок промоделюємо задачу прийняття рішень з новими значеннями та ваговими коефіцієнтами та прорахуємо згорткову оптимізаційну модель за формулою (2.6) :

$$Z^* = max \sum_{j=1}^{n} \alpha_j \beta_j a_{ij}, \qquad (2.6)$$

де α_j – нормуючі множники,

β_j – вагові коефіцієнти, що відображають відносний внесок окремих критеріїв до загального критерію,

a_{ij} – значення оцінки певного критерію для даного варіанту.

Згортка підраховує скільки разів та чи інша стратегія була оптимальною.

Таблиця 2.2 – Моделювання задачі ПР

	Просторова складність	Часова складність алгоритму	Можливість використання наявної реалізації
Симплекс метод	$O(n^2)$	$O(2^n)$	https://math.semestr.ru/simplex/simplex.php, MS Excel
Метод штучного базису	$O(n^2)$	$O(2^n)$	Немає
Жадібний алгоритм	$O(1)$	$O(n \log(n))$	Немає
Метод Гоморі	$O(n^3)$	$O(n^2)$	https://math.semestr.ru/simplex/integer.php
Метод Фора та Мальгранжа	$O(n^3)$	$O(2^n)$	Немає
Метод гілок та кордонів	$O(\log(n))$	$O(n^2)$	https://math.semestr.ru/lp/branch.php

Нижче наведені розрахунки згорткової оптимізаційної моделі :

Для симплекс методу результат згортки буде: $3 * \frac{5}{15} + 1 * \frac{3}{15} + 3 * \frac{1}{15} = 1 + 0.2 + 0.2 = 2.04$

Для методу штучного базису результат згортки буде: $3 * \frac{5}{15} + 1 * \frac{3}{15} + 1 * \frac{1}{15} = 1 + 0.2 + 0.67 = 1.87$

Для жадібного алгоритму результат згортки буде: $7 * \frac{5}{15} + 4 * \frac{3}{15} + 1 * \frac{1}{15} = 2.33 + 0.8 + 0.067 = 3.197$

Для методу Гоморі результат згортки буде: $2 * \frac{5}{15} + 3 * \frac{3}{15} + 1 * \frac{1}{15} = 0.67 + 0.6 + 0.067 = 1.33$

Для методу Фора та Мальгранжа результат згортки буде: $2 * \frac{5}{15} + 1 * \frac{3}{15} + 1 * \frac{1}{15} = 0.67 + 0.2 + 0.07 = 0.94$

Для методу гілок та кордонів результат згортки буде: $5 * \frac{5}{15} + 3 * \frac{3}{15} + 2 * \frac{1}{15} = 1.67 + 0.6 + 0.13 = 2.67$

Таблиця 2.3 – Моделювання задачі ПР з оцінками та ваговими коефіцієнтами

	Просторова складність	Часова складність алгоритму	Можливість використання наявної реалізації	Z*
Симплекс метод	3	1	3	1.07
Метод штучного базису	3	1	1	0.94
Жадібний алгоритм	7	4	1	2.137
Метод Гоморі	2	3	1	1.66
Метод Фора та Мальгранжа	2	1	1	1.94
Метод гілок та кордонів	5	3	2	2
Вагові коефіцієнти	5/15	3/15	1/15	

Згідно з розрахунками найкращі значення отримали методи жадібного алгоритму, симплекс метод, метод гілок та кордонів, тож, для вирішення оптимізаційних задач цілочисленого програмування для системи догляду та керування за пасікою реалізації було обрано метод гілок та кордонів та жадібний алгоритм.

2.4 Проєктування бази даних

Для досліджуваної проблемної області необхідно зберігати дані про пасіку, товари, обмеження щодо товарів та дані про отримання товару з пасіки. Крім того, необхідно зберігати дані про користувачів системи та їхню роботу на пасіці. Все це зумовлює появу наступних сутностей : «Пасіка», «Товар», «Роль», «Робітник».

У ході концептуального моделювання було встановлено, що сутність «Пасіка» має наступні атрибути : ID, загальна кількість рамок, ID робітника. Сутність «Вулик» має наступні атрибути: ID, загальна кількість рамок у вулику, ID пасіки. Сутність «Товар» має наступні атрибути : ID, обсяг видобування з 1 рамки, обмеження по обробці, вартість, можливість використання товару для годування бджіл. Сутність «Видобування товару» необхідна для визначення взаємодії між сутностями «Товар» та «Вулик». Ці сутності мають між собою зв’язок один до багатьох. Сутність «Роль» необхідна для визначення наявних ролей робітників у системі. Вона має наступні атрибути : ID та назва посади. Сутність «Робітник» пов’язана з сутністю «Роль» зв’язком один до багатьох, оскільки у одного робітника може бути лише одна посада. Та вона має наступні атрибути : ID, ім’я, прізвище, паспортні дані, електронна пошта, пароль, ID ролі.

Після опису необхідних для системи сутностей, було створено ER-діаграму бази даних (див. рис. 2.2).

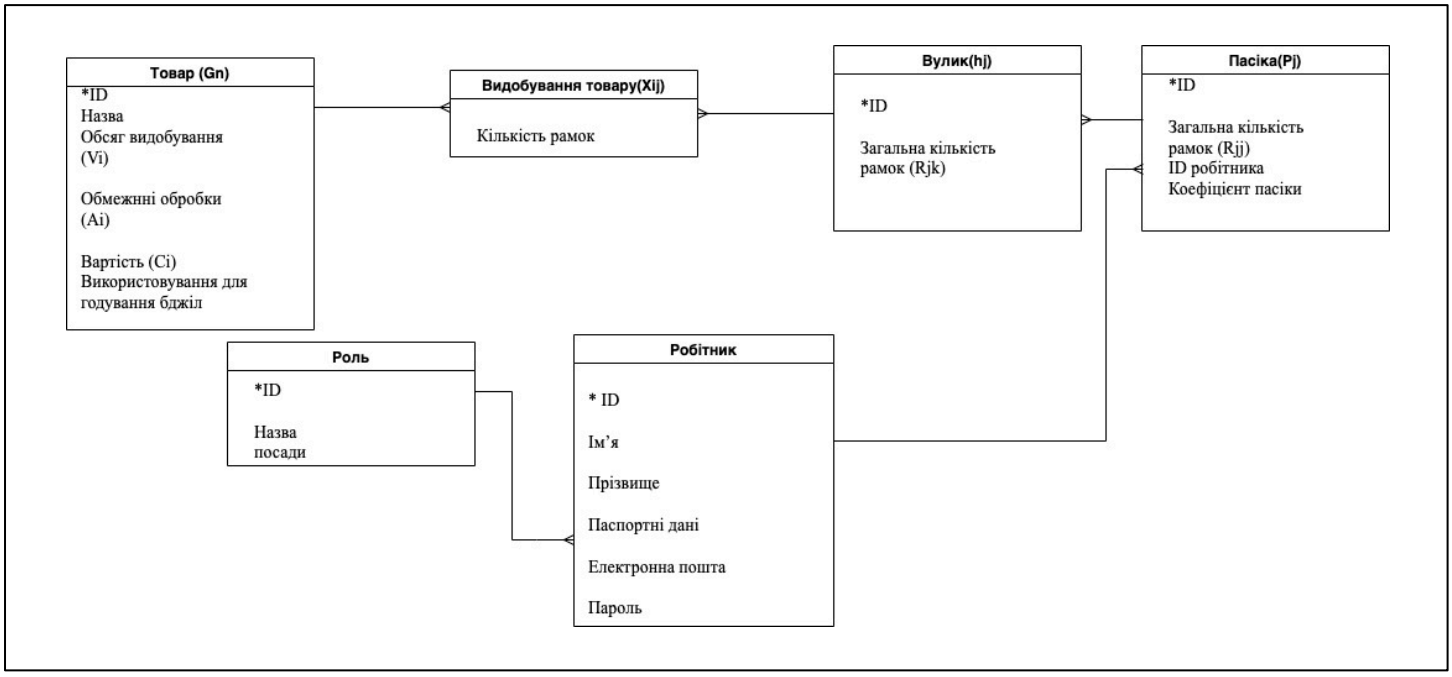

Рисунок 2.2 – ER діаграма бази даних

На діаграмі, окрім сутностей та їх атрибутів, також позначені зв'язки, які присутні між цими сутностями, та ті атрибути, що виступають первинними ключами.

Після створення ER-діаграми було створено схему бази даних. На рисунку 2.3 зображена схема бази даних :

Схема бази даних знаходиться у третій нормальній формі, а це відбувається тоді і тільки тоді, коли вона знаходиться в другій нормальній формі і жоден з неключових атрибутів не є транзитивно залежним від їх первинного ключа.

2.5 Розробка алгоритмів для вирішення оптимізаційних задач цілочисленого програмування для системи догляду за пасікою

Для вирішення задач цілочисельного програмування використовують симплекс метод, метод Гоморі, жадібний алгоритм, метод Фора та Мальгранжа та метод гілок та кордонів.

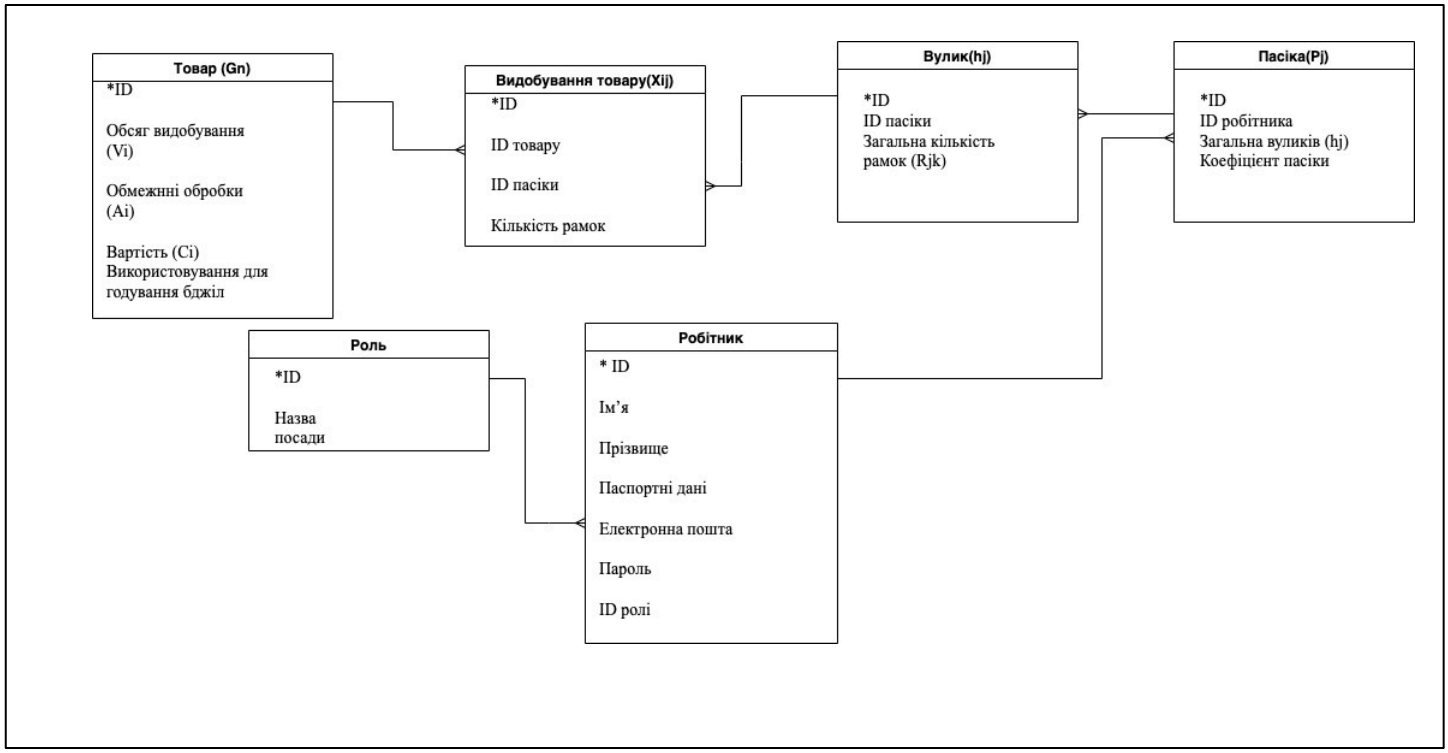

Рисунок 2.3 – Схема бази даних

Найбільш розповсюдженим методом для вирішення задач цілочисленого програмування є метод гілок та кордонів [24]. Його суть полягає у послідовному розбитті допустимої множини рішень на підмножини та розгляданні лише тих варіантів, які за деякими ознаками є перспективними, та відкиданні тих, що не є перспективними. Вирішення триває до знайдення рішення цілочисленої задачі.

Жадібний алгоритм завжди передбачає вибір, який є найкращим у даний момент, тобто виробляється локально оптимальний вибір, який гіпотетично призведе до оптимального вирішення глобального завдання. Легкий у реалізації та часто дуже ефективний за часом виконання.

Для вирішення задач цілочисленого програмування для системи догляду за пасікою було обрано, розроблено та реалізовано два різних алгоритми на основі методу гілок та кордонів та жадібного алгоритму. На рисунках 2.4 та 2.5 наведено блок схеми алгоритмів на основі жадібного алгоритму та методу гілок і кордонів відповідно.

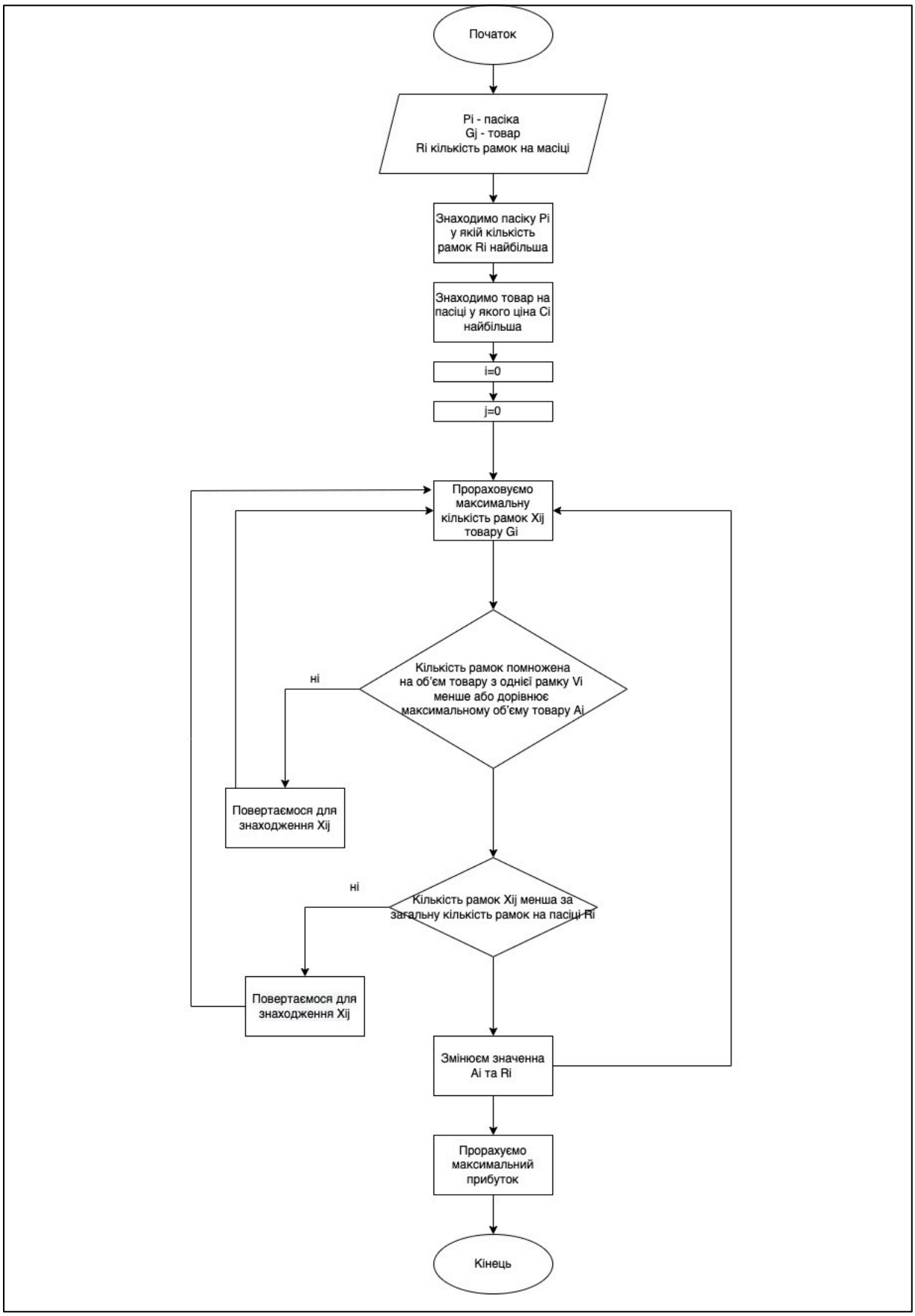

Рисунок 2.4 – Алгоритм на основі жадібного алгоритму

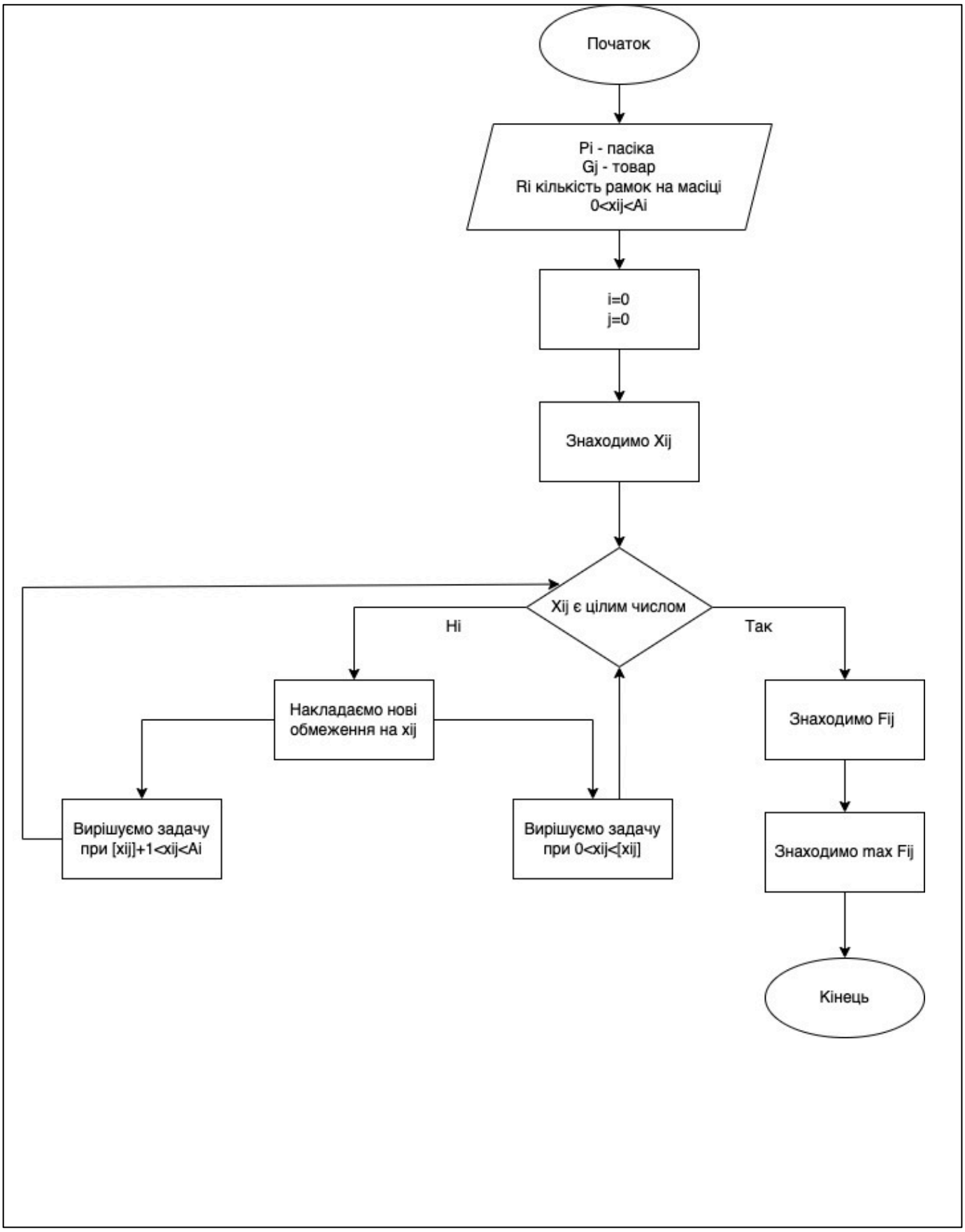

Рисунок 2.5 – Алгоритм на основі методу гілок та кордонів

Під час роботи жадібного алгоритму для початку обирається пасіка з найбільшою кількістю вуликів і рамок. На цій пасіці знаходимо товар з найбільшою вартістю та кількість рамок цього товару (x_{ij}), при якій прибуток від продажу буде максимальним. Також при розрахунках необхідно брати до уваги обмеження про цілочисельність змінної x_{ij}, об'єм товару, який може обробити пасічник за сезон та максимальну кількість рамок на пасіці. Після знаходження змінної x_{ij}виконується обчислення цільової функції, беручи до уваги коефіцієнт пасіки, який показує де знаходиться пасіка та пору року.

Під час роботи методу гілок та кордонів змінна x_{ij} обчислюється за допомогою симплекс методу, потім вона перевіряється на умову цілочисельності, якщо змінна не відповідає умові цілочисельності, то необхідно застосувати додаткові обмеження на змінну та вирішити кожну задачу. Після вирішення кожної задачі нові рішення знаходяться та знову перевіряються на умову цілочисельності, якщо умова не була задовільнена, то накладаються додаткові обмеження. Якщо ж x_{ij} є цілим числом, то далі знаходимо цільову функцію F_{ij}.

2.6 Розробка архітектури системи

Створення програмної системи вимагає розробки її архітектури. Клієнт-серверна архітектура – це обчислювальна або мережева архітектура, в якій завдання або мережеве навантаження розподілені між постачальниками послуг, так званими серверами, і замовниками послуг, так званими клієнтами. Така архітектура дозволяє розподілити навантаження та функціонал між серверною та клієнтською частинами. Ще однією перевагою такої системи є те, що все навантаження та всі розрахунки знаходяться на серверній частині, що дозволяє не висувати високі вимоги до машин користувачів системи. Крім того, така архітектура дозволяє зробити дані клієнтів більш захищеними, адже усі вони будуть знаходитися на

сервері. Клієнт-серверна архітектура дозволяє об’єднати багато різних клієнтів, у нашому випадку це вебсайт.

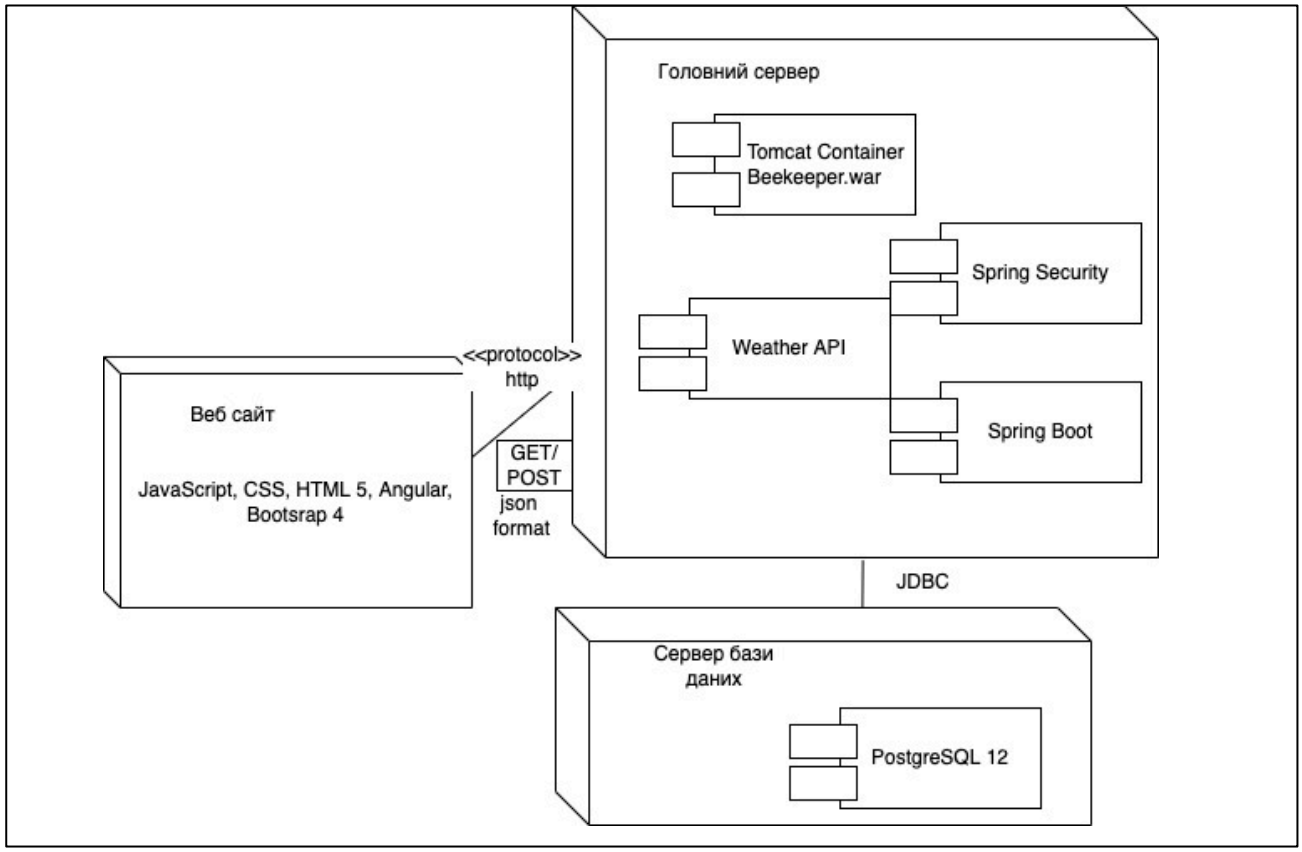

Рисунок 2.6 – Діаграма розгортання

Система «Beekeeper» складається з трьох основних частин :

– серверу (Java, Spring Boot, Spring Security);
– вебсайту (Angular);
– бази даних (Postgres).

Як видно з рисунку 2.6, всі частини системи працюють за протоколом HTTP, усі частини клієнт-серверної архітектури взаємодіють з головним сервером.

На сервері буде знаходитися основний функціонал, для розробки якого була обрана мова програмування Java 8 та Spring Boot. Превагами Spring Boot є наявність вбудованого сервера, автоматичне налаштування функції Spring та легкість у налаштуванні та написанні коду. Перевагами обраної мови програмування Java 8 є платформо незалежність, надійність, захищеність та висока продуктивність.

Оскільки сервер потребує усіх цих функцій, то обрана мова як найкраще підходить для розробки нашої системи.

Для вебзастосунку було обрано наступні технології – HTML 5, JavaScript, Angular, CSS 3, JQuery. JavaScript підтримується усіма браузерами, має гарну документацію, тож не спричинятиме проблем з відкриттям вебчастини у різних браузерах. Крім того, нами була обрана бібліотека JQuery, оскільки вона є розширюваною та добре працює з DOM.

Для зберігання даних у системі була обрана СУБД PostgreSQL. У ній знаходяться майже усі можливості всіх існуючих безкоштовних та платних СУБД, а також є додаткові можливості, властиві тільки їй. Основними перевагами PostgreSQL є те, що вона є об’єктно-орієнтованою СУБД, не має обмежень на розмір бази даних, відповідає вимогам ACID, а саме атомарність, узгодженість, ізольованість і надійність. Ця СУБД також безкоштовно розповсюджується. PostgreSQL дає можливість створювати власні користувацькі типи, транзакції, вкладені запити тощо. Обрана СУБД може використовуватися на різних платформах.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Вибір засобів програмної реалізації

Для забезпечення легкого масштабування системи та економії ресурсів користувача, необхідно обрати відповідні засоби розробки, що дозволять задовільнити поставлені задачі.

Розроблена система складається з двох частин : серверу та вебсайту. Сервер міститиме основний функціонал системи. Для його реалізації була обрана мова програмування Java 8 з використанням фреймворку Spring Boot. Java 8 – це революційна версія платформи розробки у світі. Вона включає в себе величезне оновлення моделі програмування Java та скоординовану еволюцію JVM, мови Java та бібліотек. Java 8 включає функції для підвищення продуктивності, простоти використання, покращеного поліглотного програмування, безпеки та підвищення продуктивності.

Для вебзастосунку було обрано Angular 13, Angular Material, RxJs. Angular – це платформа та фреймворк для створення односторінкових клієнтських програм за допомогою HTML і TypeScript. Angular написаний на TypeScript. Він реалізує основні та додаткові функції у вигляді набору бібліотек TypeScript, які імпортуються у програми. RxJS – це бібліотека для реактивного програмування з використанням спостережуваних, що полегшує створення асинхронного коду або коду на основі зворотного виклику [25].

Для зберігання даних у системі була обрана СУБД PostgreSQL. Перевагами обраної СУБД є об’єктно-орієнтованість, відповідність вимогам ACID, підтримка різноманітних типів даних для збереження. Також PostgreSQL контролює доступ до даних має декілька варіантів зашифрованого підключення до даних. Для роботи серверу з обраною СУБД буде використовуватися JPA (Java Persistent API).

Для кожної частини системи були обрані відповідні середовища розробки : Intellij IDEA для розробки серверу, WebStorm для розробки вебсайту.

3.2 Опис фізичної моделі бази даних

Для реалізації бази даних було обрано СУБД PostgreSQL. PostgreSQL – це потужна система об’єктно-реляційних баз даних з відкритим вихідним кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають і масштабують найскладніші робочі навантаження даних. PostgreSQL має міцну репутацію завдяки своїй перевіреній архітектурі, надійності, цілісності даних, надійному набору функцій, розширюваності та відданості спільноти відкритих вихідних кодів, які стоять за програмним забезпеченням, щоб постійно надавати продуктивні й інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, має ACID-сумісність [26].

Фізична модель бази даних зображає інформацію про всі об’єкти та типи даних у таблиці, показує відносини між цими об’єктами, містить зовнішні та первинні ключі.

База даних зберігає дані про пасіки, вулики, товари та працівників (див. рис. 3.1):

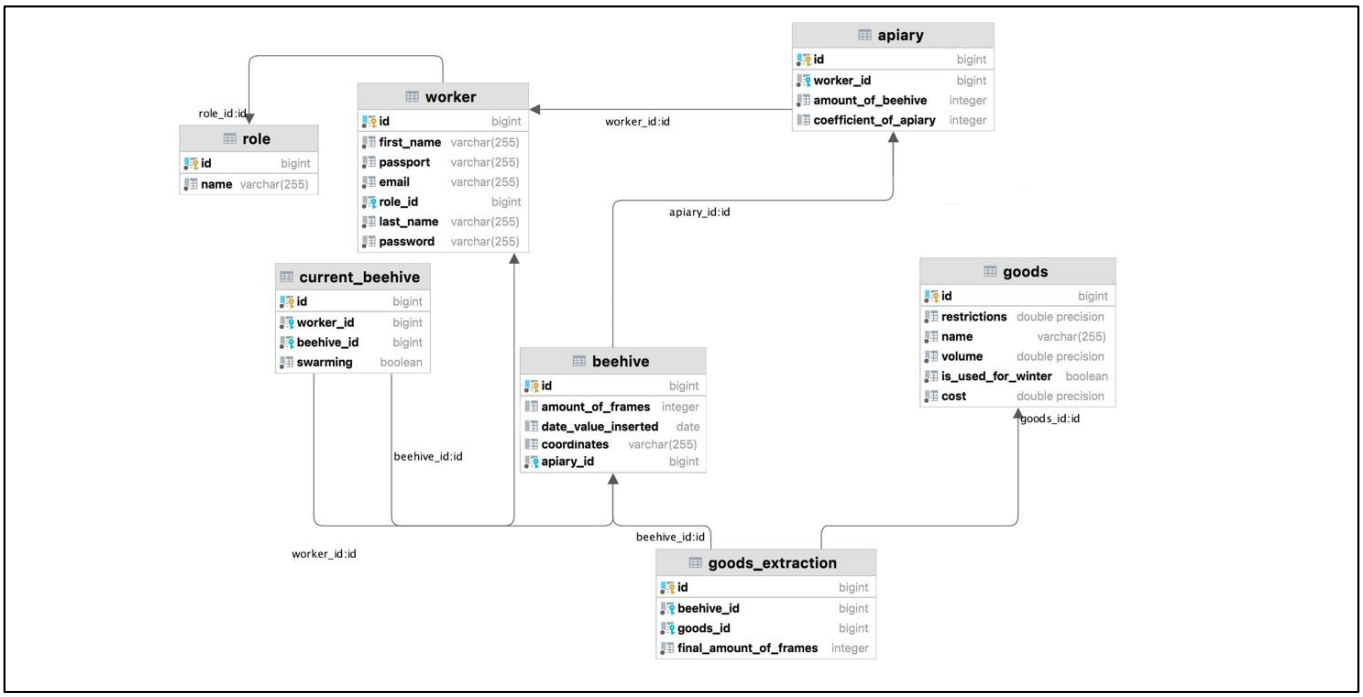

Рисунок 3.1 – Фізична модель бази даних

У базі даних є відношення один до багатьох (один робітник може працювати на багатьох пасіках) також таке відношення використовується для проміжних сутностей («Видобування товару»). База даних нормалізована до третьої нормальної форми.

3.3 Реалізація серверної частини

Для реалізації серверної частини було обрану мову програмування Java з використанням фреймворку Spring Boot. Spring Boot – це веб-фреймворк Java з відкритим вихідним кодом на основі мікросервісів. Фреймворк Spring Boot створює повністю готове до виробництва середовище, яке можна повністю налаштувати за допомогою попередньо створеного коду в його базі коду. Мікросервісна архітектура надає розробникам повністю закритий додаток, включаючи вбудовані сервери додатків.

Робота за базою даних здійснюється за допомогою JPA – Java Persistent API. Spring Boot JPA – це специфікація Java для керування реляційними даними в застосунках Java. Це дозволяє отримувати доступ і зберігати дані між об’єктом Java та реляційною базою даних. JPA слідує відображенню об’єктних відносин (ORM). Це набір інтерфейсів. Він також надає API EntityManager під час виконання для обробки запитів і транзакцій щодо об’єктів у базі даних.

Жадібний алгоритм та метод гілок і кордонів були реалізовані на серверній частині застосунку.

Фрагменти коду реалізації жадібного алгоритму та методу гілок і кордонів для отримання максимального прибутку з пасіки наведено на рисунках 3.2 та 3.3.

```
public List<OutputModel>
processGreedy(List<GoodsExtractionValue>
goodsExtractionValues, List<ApiaryValue> apiaries,
                                        List<BeehiveValue>
beehiveValues, List<GoodsValue> goodsValues) {
    List<OutputModel> outputModels = new ArrayList<>();
    Map<Long,List<BeehiveValue>> beehives =
beehiveValues.stream()
            .collect( Collectors.groupingBy(beehive->
beehive.getApiaryValue().getId()));
    Map<Long,List<GoodsExtractionValue>> goodsExtraction =
goodsExtractionValues.stream()
            .collect(
Collectors.groupingBy(goodsExtractionValue->
goodsExtractionValue
                    .getBeehive().getId()));
    Map<Long,List<GoodsValue>> beehivesToGoodsSorted =
convertToViewBeehiveGoods(goodsExtraction,
            goodsValues);
    Map<Long,Integer>apiaryToAmountOfFrames =
sumMapValues(beehives);
    int j = 0;
    for (Map.Entry<Long,Integer> apiary:
apiaryToAmountOfFrames.entrySet()) {
        OutputModel outputModel = new OutputModel();
        List<BeehiveValue> beehive  =
beehives.get(apiary.getKey());
        currentMaxAmountOfFrames = apiary.getValue();
        for (BeehiveValue bv:beehive) {
          long goodsId =
beehivesToGoodsSorted.get(bv.getId()).get(0).getId();
            double currentCost = 0.0;

            GoodsExtractionValue production =
getProductionByBeehiveAndGoods(bv.getId(),goodsId,
goodsExtractionValues);
            int numberOfBeehives =
findNumberOfBeehives(beehivesToGoodsSorted.get(bv.getId()).get
(0),
                    production);
           // production.setAmountOfBeehive(numberOfBeehives);
            int currentFrameCount = numberOfBeehives *
production.getFinalCountOfFrames();
            currentCost += currentFrameCount *
beehivesToGoodsSorted.get(bv.getId()).get(0).getCost()
                    *
beehivesToGoodsSorted.get(bv.getId()).get(0).getVolume();
            List<Integer> leftAmountOfProduction = new
ArrayList<>();
```

Рисунок 3.2 – Частина коду для жадібного алгоритму

```
public void createdTree(Node node, int index) {
    if(index >= goodsExtractionValue.getFinalCountOfFrames())
{
        return;
    }
    ItemInfo currentItem = itemInfoList.get(index);
    if(node.getLeft() == null) {
        Node leftNode = new Node();
        leftNode.setValue(currentItem.getValue() +
node.getValue());
        leftNode.setWeight(currentItem.getWeight() +
node.getWeight());
        leftNode.setLayer(index + 1);
        int leftAmountOfFrames =
goodsExtractionValue.getFinalCountOfFrames() -
leftNode.getWeight();
        if(leftAmountOfFrames >= 0) {
            if(index + 1 < itemInfoList.size()) {
                ItemInfo nextNode = itemInfoList.get(index +
1);
                int up = leftAmountOfFrames *
(nextNode.getValue() / nextNode.getWeight());
                if(index == 0) {
                    leftNode.setUp(up +
currentItem.getValue());
                } else {
                    leftNode.setUp(up + leftNode.getValue());
                }
            } else {
                leftNode.setUp(node.getValue());
            }
            leftNode.setCurrentResult(node.getCurrentResult()
+ 1);
            node.setLeft(leftNode);
            createdTree(node.getLeft(), index + 1);
        }
    }
    if(node.getRight() == null) {
        Node rightNode = new Node();
        rightNode.setValue(node.getValue());
        rightNode.setWeight(node.getWeight());
```

Рисунок 3.3 – Частина коду для методу гілок та кордонів

3.4 Опис інтерфейсу користувача

Для взаємодії користувача з системою було створено вебсайт. Він створений з використанням Angular 13, Angular Material, RxJs. Angular Material –це компонент бібліотеки інтерфейсу користувача, розроблений Google у 2014 році. Він спеціально розроблений для розробників AngularJS. Це допомагає структурувати програму. Його компоненти допомагають створювати привабливі, послідовні та функціональні вебсторінки та вебдодатки. Він використовується для створення адаптивного та швидшого вебсайту.

Актуальність наявності вебсайту зумовлена необхідністю мати можливість прораховувати оптимальний план щодо видобування та продажу товару. Для початку роботи користувачеві необхідно зареєструватися та ввійти у систему (див. рис. 3.4).

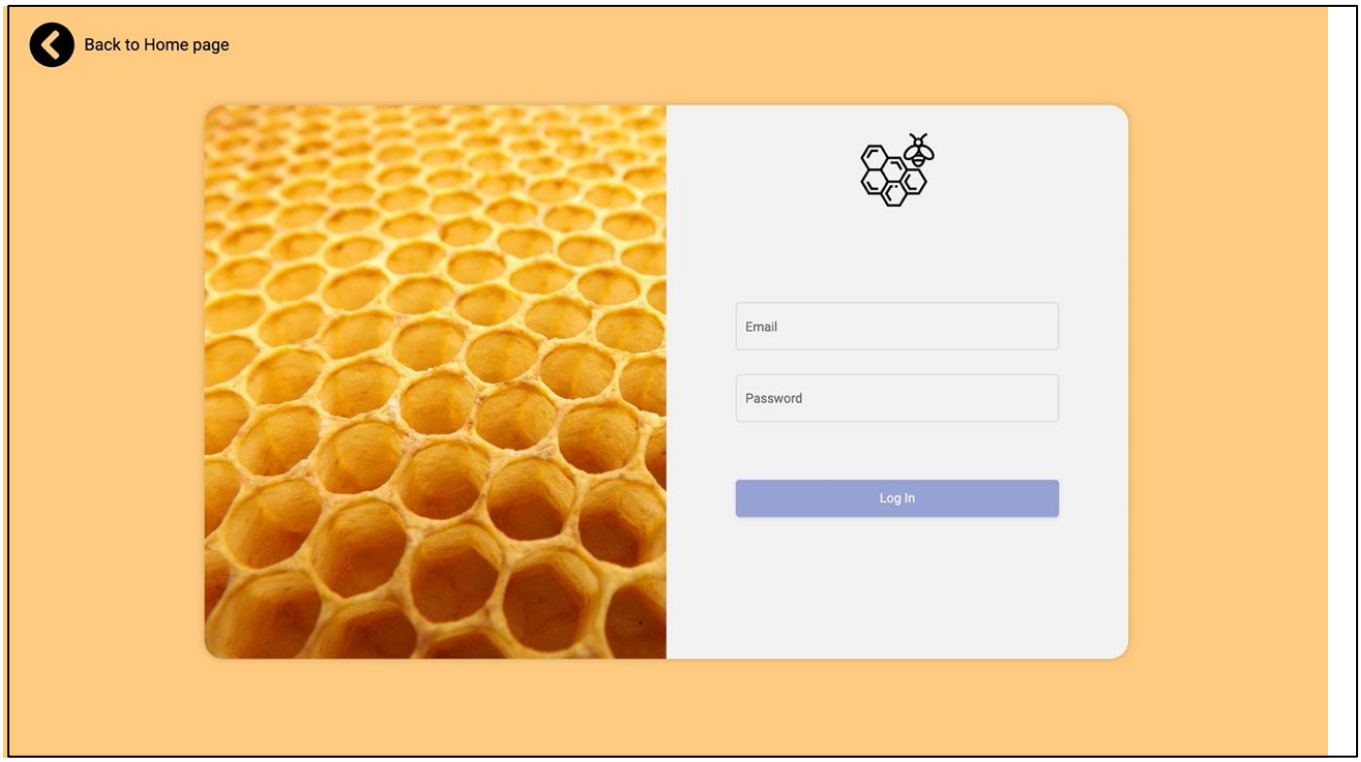

Рисунок 3.4 – Сторінка входу

На сайті присутні дані щодо вартості товару та максимального об’єму товару, який можна отримати з пасіки. Ці дані можна змінювати, щоб отримувати коректні результати розрахунку оптимального плану. На даній сторінці можна видаляти, редагувати або додавати товари. Ціна товарів може змінюватися в залежності від місця розміщення пасіки, тому на сайті було додано фільтр, який відображає до якої саме пасіки відноситься товар (див. рис. 3.5).

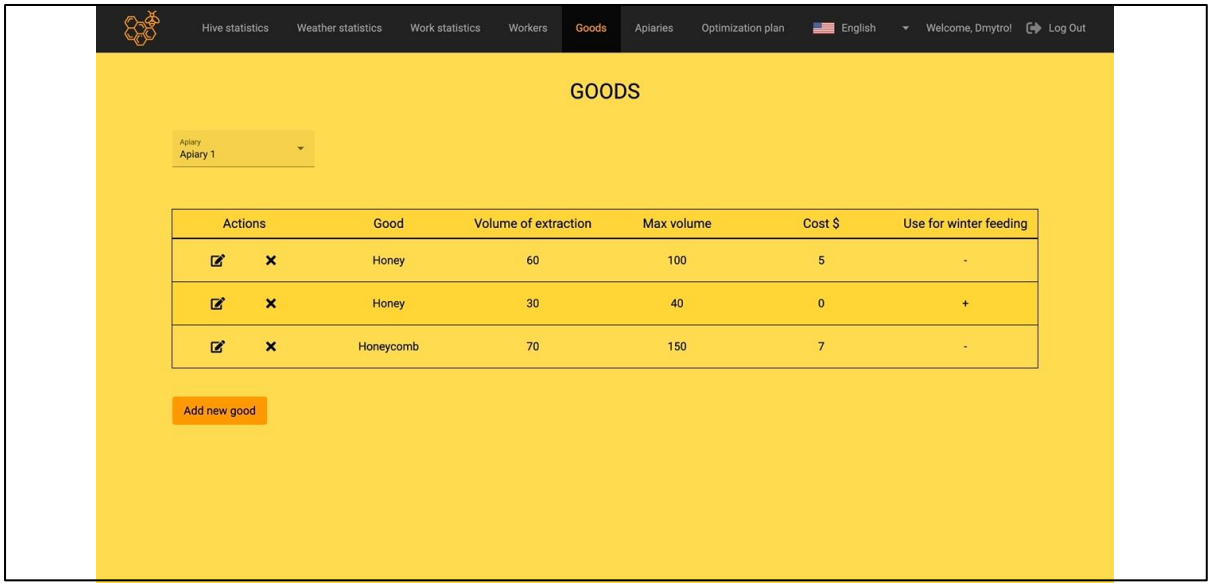

Рисунок 3.5 – Сторінка редагування даних про товар

Також сайт містить інформацію щодо пасіки : кількість вуликів, рамок. Ці дані також можна коригувати (див. рис. 3.6).

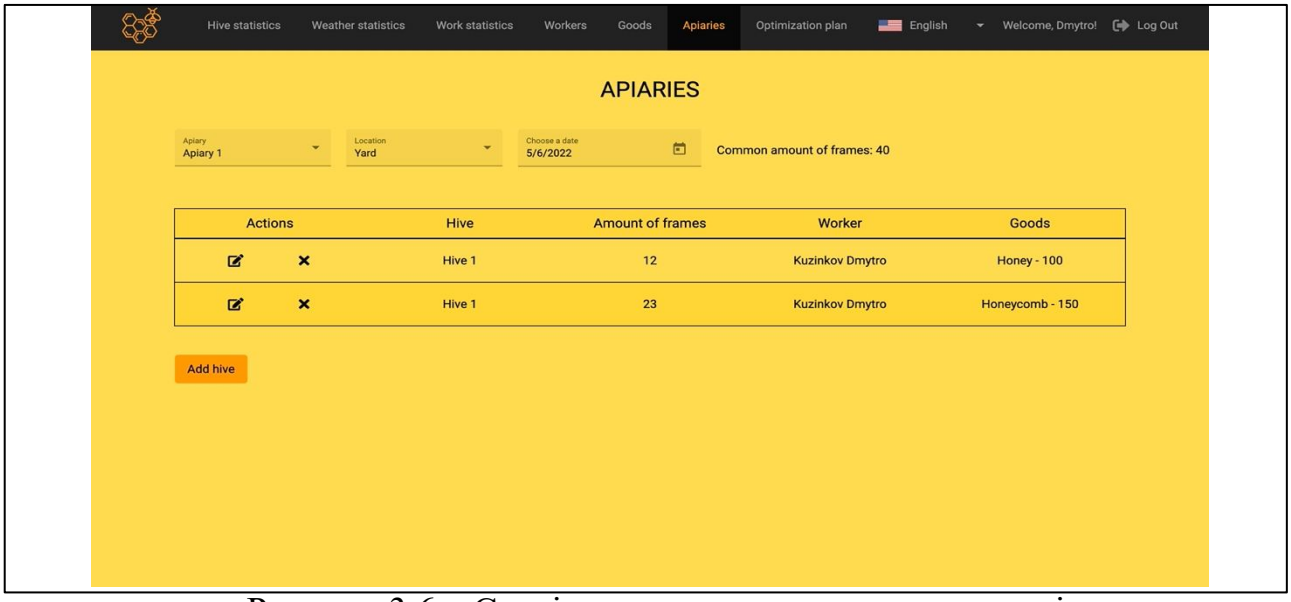

Рисунок 3.6 – Сторінка редагування даних про пасіку

Після перевірки усіх необхідних для розрахунків даних можна отримати оптимальний план щодо видобування та продажу (див. рис. 3.7).

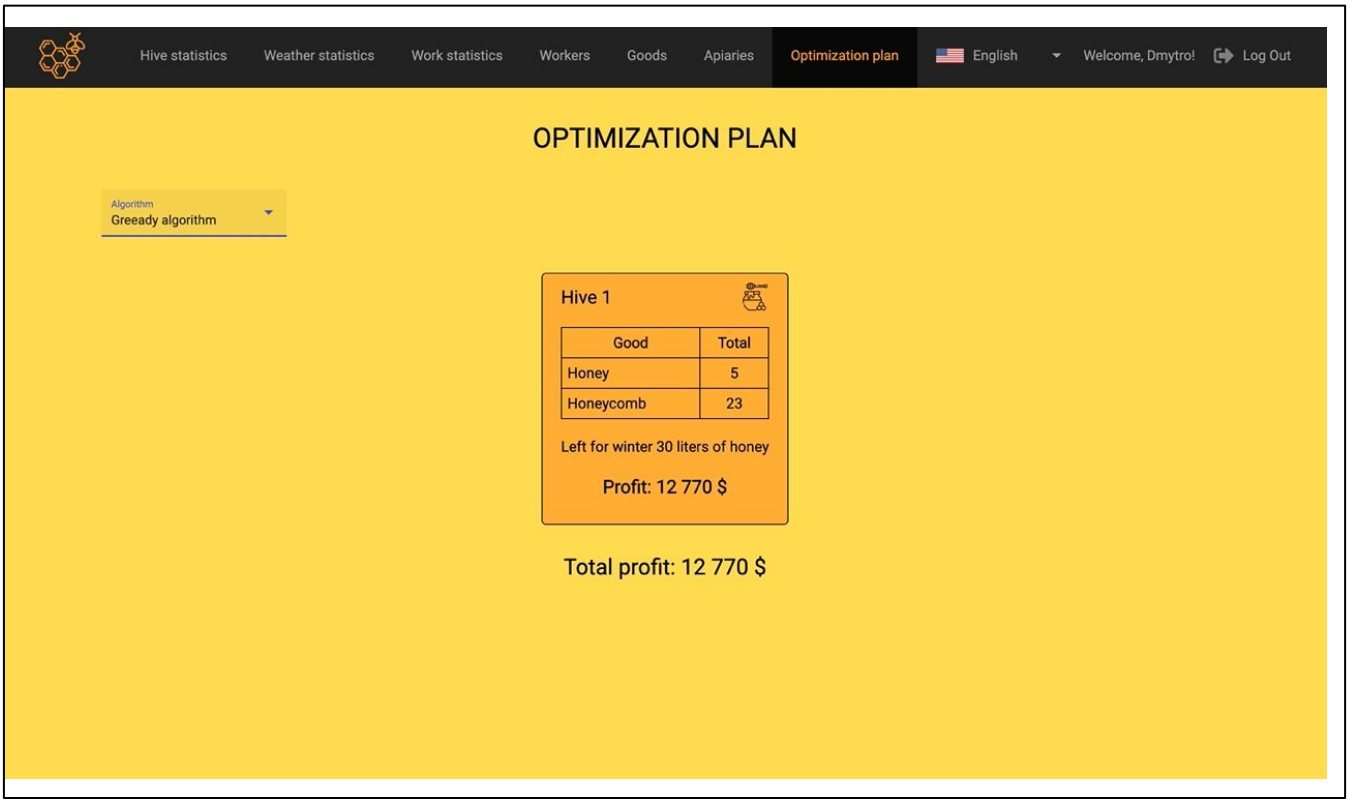

Рисунок 3.7 – Сторінка результатів виконання алгоритмів

Для проведення порівняння роботи двох реалізованих алгоритмів на сайті є можливість за допомогою фільтру обрати бажаний алгоритм.

4 ОПИС ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Планування експерименту

Для перевірки ефективності створених методів вирішення оптимізаційних задач було сплановано та проведено ряд експериментів. Під час планування було використано наступні передумови :

– пасіки містять різний тип вуликів;

– пасіки знаходяться в однакових умовах;

– з пасіки можна отримати однаковий тип товарів. Було використано 3 типи товарів;

– у якості методу вирішення оптимізаційної задачі були використані методи вирішення задач цілочисельного програмування.

Заміри під час експериментів проводились за наступними критеріями :

– кількість змінних у математичній моделі;

– час роботи алгоритму;

– навантаження на процесор;

– використання оперативної пам’яті.

Під час дослідження було заплановано та проведено серію експериментів :

– знаходження оптимального плану отримання продукції з пасіки для отримання максимального прибутку від реалізації жадібним алгоритмом (G1, G2, G3);

– знаходження оптимального плану отримання продукції з пасіки для отримання максимального прибутку від реалізації методом гілок та кордонів (B1, B2, B3).

Тестування передбачало використання даних з трьох пасік, що містили різну кількість вуликів та рамок :

– пасіка 1 : кількість вуликів 10, кількість рамок 240;

– пасіка 2 : кількість вуликів 20, кількість рамок 480;

– пасіка 3 : кількість вуликів 30, кількість рамок 720.

З метою уникнення похибки у результатах, усі експерименти проводились на одному сервері, який мав наступні характеристики :

- центральний процесор – Intel Core i5;
- частота процесору – 2 GHz%
- 4-ядра;
- 16 ГБ операційної пам’яті.

Експериментальна апробація на інших серверах можлива, проте не є обов’язковою, оскільки головною метою проведення зазначених експериментів було виявлення певних закономірностей. Окремо зазначимо, що при проведенні експериментів на різних серверах необхідно привести їх характеристики до одного виду, щоб уникнути впливу сторонніх характеристик на роботу алгоритмів.

4.2 Результати проведення експерименту

Щоб уникнути похибок під час розрахунків для експериментальної апробації системи було обрано наступний ряд товарів, які використовувались для розрахунків : мед, віск, стільники з медом. Було проведено три серії експериментів з різним набором початкових даних, що передбачали 50 запусків обчислення цільової функції.

За результатами проведених експериментів у таблицю були внесені дані щодо часу виконання алгоритмів, використання оперативної пам’яті та результату обчислення цільової функції. Результати проведення експериментів записані у таблицях 4.1, 4.2, 4.3.

Таблиця 4.1 – Результати серії експериментів для «Пасіка 1»

Алгоритм/Показники	Середній час роботи алгоритму, мс	Середнє використання оперативної пам’яті, Кб	Середнє значення максимального прибутку
Жадібний алгоритм, G1	11	478	20800
Метод гілок та кордоні (власна реалізація), B1	55	859	20798

Таблиця 4.2 – Результати серії експериментів для «Пасіка 2»

Алгоритм/Показники	Середній час роботи алгоритму, мс	Середнє використання оперативної пам’яті, Кб	Середнє значення максимального прибутку
Жадібний алгоритм, G2	23	643	56540
Метод гілок та кордоні (власна реалізація), B2	68	1051	56540

Таблиця 4.3 – Результати серії експериментів для «Пасіка 3»

Алгоритм/Показники	Середній час роботи алгоритму, мс	Середнє використання оперативної пам’яті, Кб	Середнє значення максимального прибутку
Жадібний алгоритм, G3	34	874	90626

Кінець таблиці 4.3

Алгоритм/Показники	Середній час роботи алгоритму, мс	Середнє використання оперативної пам’яті, Кб	Середнє значення максимального прибутку
Метод гілок та кордоні (власна реалізація), В3	79	1300	90600

Як бачимо з результатів експериментів усі значення є закономірними для пасік різної величини. Найбільш ефективним виявився жадібний алгоритм, він перемагає за рахунок більш швидкого виконання обчислення та більш кращого обчислення цільової функції.

Метод гілок та кордонів показав гірші результати через велику кількість лінійних операцій, які необхідно виконувати під час розрахунків, не зважаючи на це, результати використання цього методу є досить близькими до результатів жадібного алгоритму.

4.3 Висновки та рекомендації з експериментального дослідження

За результатами проведення експериментів можна зробити висновок про їх закономірність. Жадібний алгоритм виявився найбільш ефективним під час проведення експериментів, оскільки він провів більш швидкі розрахунки та отримав найкращі результати цільової функції.

Метод гілок та кордонів виявився не таким ефективним, що зумовлено необхідністю проведення великої кількості кількість лінійних операції. Не зважаючи, на тривалий час виконання метод гілок та кордонів показав гарні результати в обчисленні цільової функції.

На основі проведених експериментів сформулюємо рекомендації стосовно використання методів цілочисельного програмування для системи керування пасікою :

– жадібний алгоритм варто використовувати для розрахунку цільової функції, оскільки він має невеликий час виконання обчислення та показує гарні результати. Це обумовлено тим, що він розробляється конкретно під задачу;

– метод гілок та кордонів є досить ефективним у порівнянні з іншими методами вирішення задач цілочисельного програмування. Так, у порівнянні з методом Гоморі, він має менше похибок під час округлення результатів, проте беручи до уваги велику кількість лінійних операцій, які виконуються під час обчислення, метод гілок та кордонів потребує тривалішого часу виконання, ніж жадібний алгоритм.

ВИСНОВКИ

Результати кваліфікаційної роботи полягають у теоретичному дослідженні, розробці та практичній реалізації методів вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою та проєктування програмної системи.

В результаті роботи було досліджено стан розробленості існуючих аналогів реалізованої системи на ринку продажів, з'ясовано обмеженість їх функціоналу, що не задовольняє існуючі вимоги сьогодення, проведено аналіз проблемної галузі, теоретично обґрунтовано вибір експериментальної бази дослідження, реалізовано та експериментально перевірено обрані алгоритми.

Для експериментальної апробації було обрано два методи : жадібний алгоритм та метод гілок і кордонів. Реалізація усіх методів знаходиться на серверній частині системи. За результатами експериментів встановлено, що використання жадібного алгоритму для вирішення оптимізаційних задач цілочисленого програмування для системи керування пасікою є найбільш доцільним.

Серверну частину реалізовано за допомогою мови програмування Java, фреймворку Spring Boot. Вебсайт реалізовано за допомогою Angular 11, Angular Material.

Досягнення поставленої мети було зумовлено виконанням ряду завдань, а саме :

- аналізом існуючих методів вирішення оптимізаційних задач цілочисленого програмування та обрати кращі для подальшого дослідження;
- проведеням аналізу та моделювання предметної області роботи пасіки;
- побудовою математичної моделі оптимізаційної задачі в області роботи пасіки;
- розробкою схеми бази даних для зберігання даних моделей;

– розробкою алгоритмів обраних методів вирішення оптимізаційних задач та їх реалізацією;

– розробкою програмної системи, яка складається з серверної та частини та вебсайту;

– проведенням експериментального порівняння методів.

За результатами роботи було розроблено презентацію, слайди якої наведено у додатку Г. Результати кваліфікаційної роботи магістра пройшли попередню апробацію шляхом публікації тез доповідей на дванадцятій міжнародній науково-технічній конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» (див. додаток Д) та на Міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті» (див. додаток Е).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЬ

1. Бджільництво України: стан, проблеми, шляхи розв'язання – НААН. URL : http://naas.gov.ua/news/?ELEMENT_ID=5061 (дата звернення : 07.02.2022).
2. Verkhohliadova N. , Levchynskyi D. , Kolesnykov D. Espacios, Deregulation of economic activities: Domestic realities and foreign experience, 2019, 40(24)
3. Руцька Т. Пчёлы. Разведение и содержание. М., 2012. С.10.
4. Оптимизационные задачи. Общие сведения . URL : https://simenergy.ru/math-analysis/solution-methods/93-optimization-methods (дата звернення : 17.01.2022).
5. Введення у математичне програмування. URL : https://intuit.ru/studies/courses/1020/188/lecture/4917 (дата звернення : 23.01.2022).
6. Цілочислове програмування URL : https://studopedia.org/2-26775.html (дата звернення : 13.03.2022).
7. Клієнт-серверна архітектура URL : https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення : 14.03.2022).
8. Hivemanager Beekeeping App URL : https://www.beequip.nz/pages/hivemanager-beekeeping-app (дата звернення : 23.02.2022).
9. Nectar URL: https://www.nectar.buzz/ (дата звернення: 15.02.2022).
10. InspectNext URL : https://apps.apple.com/app/id1524702495 (дата звернення : 07.03.2022).
11. James E. Tew The Beekeeper's Problem Solver: 100 Common Problems Explored and Explained, Quarto. 2015. 224 p., photos. index. ISBN 9781631590351.
12. Жалдак М. І., Триус Ю. В. Основи теорії і методів оптимізації : навч. посіб. Черкаси: Брама-Україна, 2005. 605 с.

13. Mazurova O., Samantsov O., Topchii O., Shirokopetleva M. 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, doi: 10.1109/PICST51311.2020.9468057.

14. Сутність лінійного програмування URL : https://pidru4niki.com/12340714/menedzhment/sutnist_liniynogo_programuvannya (дата звернення : 16.03.2022).

15. Методи розв’язання задач цілочисельного лінійного програмування URL : https://studfile.net/preview/5193567/page:15/ (дата звернення : 15.03.2022).

16. Симплексный метод решения ЗЛП. URL : https://math.semestr.ru/simplex/simplex.php (дата звернення : 10.02.2022).

17. Симплекс метод решения задач линейного программирования. URL : https://function-x.ru/simplex_method_example_algorithm.html (дата звернення : 05.01.2022).

18. Метод штучного базису URL : https://wiki.cuspu.edu.ua/index.php/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B1%D0%B0%D0%B7%D0%B8%D1%81%D1%83 (дата звернення : 14.03.2022).

19. Жадібні алгоритми URL : https://devzone.org.ua/post/zhadibni-algoritmi (дата звернення : 09.03.2022).

20. Метод Гоморі (метод відсікаючих площин) URL : https://www.mathros.net.ua/metod-gomori-metod-vidsikajuchyh-ploshhyn.html (дата звернення : 13.03.2022).

21. Поняття про метод гілок і кордонів. URL : https://stud.com.ua/80792/ekonomika/ponyattya_metod_gilok (дата звернення : 16.02.2021).

22. Поняття про метод гілок і меж URL : https://stud.com.ua/80792/ekonomika/ponyattya_metod_gilok (дата звернення : 15.03.2022).

23. Метод свёртывания критериев. URL : https://studfile.net/preview/4031478/page:6/ (дата звернення : 21.02.2022).

24. Вирішення задач цілочисельного програмування URL : https://cyberleninka.ru/article/n/reshenie-zadach-tselochislennoy-lineynoy-optimizatsii/viewer (дата звернення : 17.03.2022).

25. The RxJS library URL : https://angular.io/guide/rx-library (дата звернення : 19.04.2022).

26. About PostgreSQL URL : https://www.postgresql.org/about/ (дата звернення : 05.04.2022).