

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПЛАНУВАННЯ РОБОТИ ЗООМАГАЗИНУ.

RESEARCH AND SOFTWARE IMPLEMENTATION OF

METHODS FOR PLANNING OPERATION OF A PET STORE

Виконав(ла): студент(ка) 2 курсу, групи КНТ-213м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

КОЧЕВ Г.Г.

(ПРІЗВИЩЕ та ініціали)

Керівник ЛЬОВКІН В.М.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2024_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ КНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 122 Комп'ютерні науки _____
(код і найменування)
Освітня програма (спеціалізація) _____ Системи штучного інтелекту _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« _____ » _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОЧЕВА Геннадія Геннадійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів планування роботи зоомагазину. Research and Software Implementation of Methods for Planning Operation of a Pet Store

керівник проєкту (роботи) к.т.н., доцент ЛЬОВКІН Валерій Миколайович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 7 » жовтня 2024 року № 398

2. Строк подання студентом проєкту (роботи) 01 грудня 2024 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації програмного забезпечення. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	ЛЬОВКІН В.М., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 06 » вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-5 тижні	Розділ 2
4	Проектування та розробка архітектури програми	6 тиждень	Розділ 3
5	Розробка програми	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження	9-10 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	11 тиждень	Додатки
8	Захист роботи	12 тиждень	

Студент(ка)


(підпис)

Геннадій КОЧЕВ

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис)

Валерій ЛЬОВКІН

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
111 с., 27 рис., 7 табл., 3 дод., 12 джерел.

ЗООМАГАЗИН, ЗАМОВЛЕННЯ, ВОЛОГИЙ КОРМ, ПІДПИСКА,
ВЕБЗАСТОСУНОК, КЛАСИФІКАЦІЯ.

Об'єкт роботи – процес роботи зоомагазину. Предмет роботи – методи планування роботи зоомагазину. Мета роботи – розробити програмне забезпечення підтримки роботи зоомагазину для підвищення інформованості прийняття рішень стосовно планування роботи зоомагазину.

Матеріали, методи та технічні засоби: мова Python разом з набором пакетів для роботи з даними та машинного навчання, фреймворк веброзробки Django разом з системою керування базами даних PostgreSQL.

Результати. Вебзастосунок зоомагазину реалізований і призначений для пошуку, перегляду інформації, яка стосується кормів для тварин, замовлення цих кормів та керування такими замовленнями, визначення наявної кількості кормів у зоомагазині на основі даних про продані корми, наявні запаси та очікувані замовлення, що реалізовано за допомогою виконання класифікації замовлень на основі навченої моделі з використанням запропонованого методу.

Висновки. Наукова новизна роботи полягає в методі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи, який виконує класифікацію замовлень на основі моделі, побудованої методом Random Forest, визначає виконання відбору ознак для побудови даної моделі для конкретного типу вологого корму, підготовку вибірки даних для відповідного навчання моделі.

Галузь використання. Підтримка роботи зоомагазинів стосовно продажу кормів для тварин та планування даної роботи.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 111 pages, 27 figures, 7 tables, 3 appendixes, 12 sources.

PET STORE, ORDER, WET FOOD, SUBSCRIPTION, WEBSITE, CLASSIFICATION.

The object of the work is the pet store's work process. The subject of the work is methods of planning the work of a pet store. The purpose of the work is to develop software to support the operation of a pet store to increase the awareness of decision-making regarding the planning of the operation of a pet store.

Materials, Methods, and Techniques: Python language along with a set of data and machine learning packages, Django web development framework along with PostgreSQL database management system.

The results. The pet store web application is implemented and designed to search, view information related to animal feed, order and manage such orders, determine the amount of feed available in a pet store based on data on sold feed, available stock and expected orders, which is implemented by performing classification orders based on the trained model created by the proposed method.

Conclusions. The scientific novelty of the work is in the method of classifying orders in a pet store in relation to the subsequent order of wet food for work planning, which performs the classification of orders based on the model built by the Random Forest method, determines the execution of the selection of features to build a given model for a specific type of wet food, the preparation of a sample of data for appropriate model training.

Field of use. Support of the work of pet stores regarding the sale of animal feed and the planning of this work.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Огляд вебзастосунків підтримки роботи зоомагазину.....	10
1.2 Порівняння аналогів розробки вебзастосунку підтримки роботи зоомагазину	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	17
2.1 Задача класифікації замовлень стосовно подальшого замовлення вологого корму для планування роботи зоомагазину.....	17
2.2 Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи	20
2.3 Висновки за розділом 2	23
3 Проєктування програмного забезпечення	24
3.1 Вибір мови та засобів розробки.....	24
3.2 Моделювання роботи з програмою за функціональними вимогами	25
3.3 Проєктування структури бази даних.....	29
3.4 Висновки за розділом 3	41
4 Основні рішення щодо реалізації програмного забезпечення	42
4.1 Опрацювання замовлень корму для тварин у вебзастосунку.....	42
4.2 Рішення з реалізації вебзастосунку	43
4.3 Висновки за розділом 4	60
5 Експлуатація, тестування та експериментальне дослідження програми	61
5.1 Опис програми.....	61
5.2 Експлуатація програми.....	61
5.3 Експериментальне дослідження	66
5.4 Висновки за розділом 5	70
Висновки	72

Перелік джерел посилання	73
Додаток А Технічне завдання	75
Додаток Б Текст програми	81
Додаток В Слайди презентації.....	103

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

CART – Classification and Regression Tree;

CSV – Comma-Separated Values;

БД – база даних;

ВС – вебсайт;

ПЗ – програмне забезпечення.

ВСТУП

Організація та планування роботи зоомагазину потребує врахування багатьох факторів для розуміння власної клієнтської бази:

- з однієї сторони ця клієнтська база визначається тими замовленнями, які вже було виконано;
- з другого боку вона пов'язана з тими замовленнями, які принесуть повторні замовлення в майбутньому;
- з третьої сторони цього замало для планування роботи зоомагазину, адже потрібно розуміти, які корми клієнт в подальшому замовить: розуміння хоча би групи кормів дозволяє планувати очікувану кількість замовлень за групою кормів, а відповідно і замовляти такі корми для продажу в зоомагазині (якщо кількість кормів буде недостатньою, клієнт може не повернутися в цей зоомагазин, а обрати конкурента, проте якщо кількість кормів є надмірною, то це додаткові витрати для зоомагазину).

До того ж всі ці сторони мають підтримуватися вебзастосунком зоомагазину, таким чином підкреслюючи, що забезпечення продажу кормів на даний момент вже недостатньо, зважаючи на конкурентність такого бізнесу. Врахування особливостей тварини, для якої користувач вебзастосунку як її господар збирається придбати корм, є важливим і сприяє зручності його роботи, а значить і задоволеності взаємодії з зоомагазином загалом. Тож цей набір факторів має бути врахований під час розробки вебзастосунку в цій роботі, щоб сприяти задоволеності клієнтів зоомагазину з одного боку та інформованості для реалізації планування роботи зоомагазину з іншого боку.

Результатом цього попереднього огляду проблеми, який далі деталізовано оглядом можливостей, які доступні в сучасних вебзастосунках зоомагазинів, є визначення мети роботи, що передбачає розробку програмного забезпечення (ПЗ) підтримки роботи зоомагазину для підвищення інформованості прийняття рішень стосовно планування роботи зоомагазину.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд вебзастосунків підтримки роботи зоомагазину

Турбота за тваринами в Україні знаходиться на відносно високому рівні на даний момент, незважаючи на війну в країні. Звісно, що війна призвела до величезної кількості проблем [1], а зокрема велика кількість тварин загинула через бойові дії [2]. Проте те, які виїжджаючи з країни, українці забирали за собою тварин, долаючи разом величезні відстані, стало джерелом навіть для кінематографу [3]. Під час переселення з зони бойових дій українці забирали з собою тварин, шукали для них прихисток [4]. Проте турбота про умови, в яких проживають тварини, в Україні є широковідомою і ще до повномасштабного вторгнення [5]. У країні є велика кількість зоозахисників, що борються за права тварин. Проте звичайне життя домашніх тварин, що знаходиться поза такими важкими і доленосними подіями, пов'язано з забезпеченням достатньо харчування зокрема. А для цього потрібні зоомагазини, які достатньо сильно поширені в Україні на даний момент у різних містах. Відповідно майже кожен такий зоомагазин має вебсайт (ВС), який є точкою доступу до інформації про наявні корми, забезпечення замовлення кормів для власної тварини, контролю, керування виконаними замовленнями. Зважаючи на особливості планування роботи зоомагазину, що мають бути представлені та розроблені в роботі, важливо також зрозуміти, які саме базові функції подібні ВС пропонують користувачам. Такі функції з одного боку надають доступ до інформації, але з іншого забезпечують накопичення замовлень, стаючи джерелом для прийняття рішень щодо планування роботи зоомагазинів.

ВС зоомагазину Master ZOO (рис. 1.1) представляє не окремий зоомагазин, а мережу таких зоомагазинів, що розташовані в різних містах України [6]. Ця мережа має свою службу підтримки, що достатньо важливо під час роботи з клієнтами та сприяє задоволеності клієнтів. ВС надає доступ до акційних товарів, товарів за призначенням для груп тварин (собак, котів,

птахів, гризунів тощо). Слід зазначити, що товари представлені не тільки кормами, але і різними іграшками для тварин, додатковим приладдям (переноски, сумки). У розробці, що виконується в роботі, функціонально має бути забезпечено роботу саме з кормами для тварин. Це важливо враховувати.

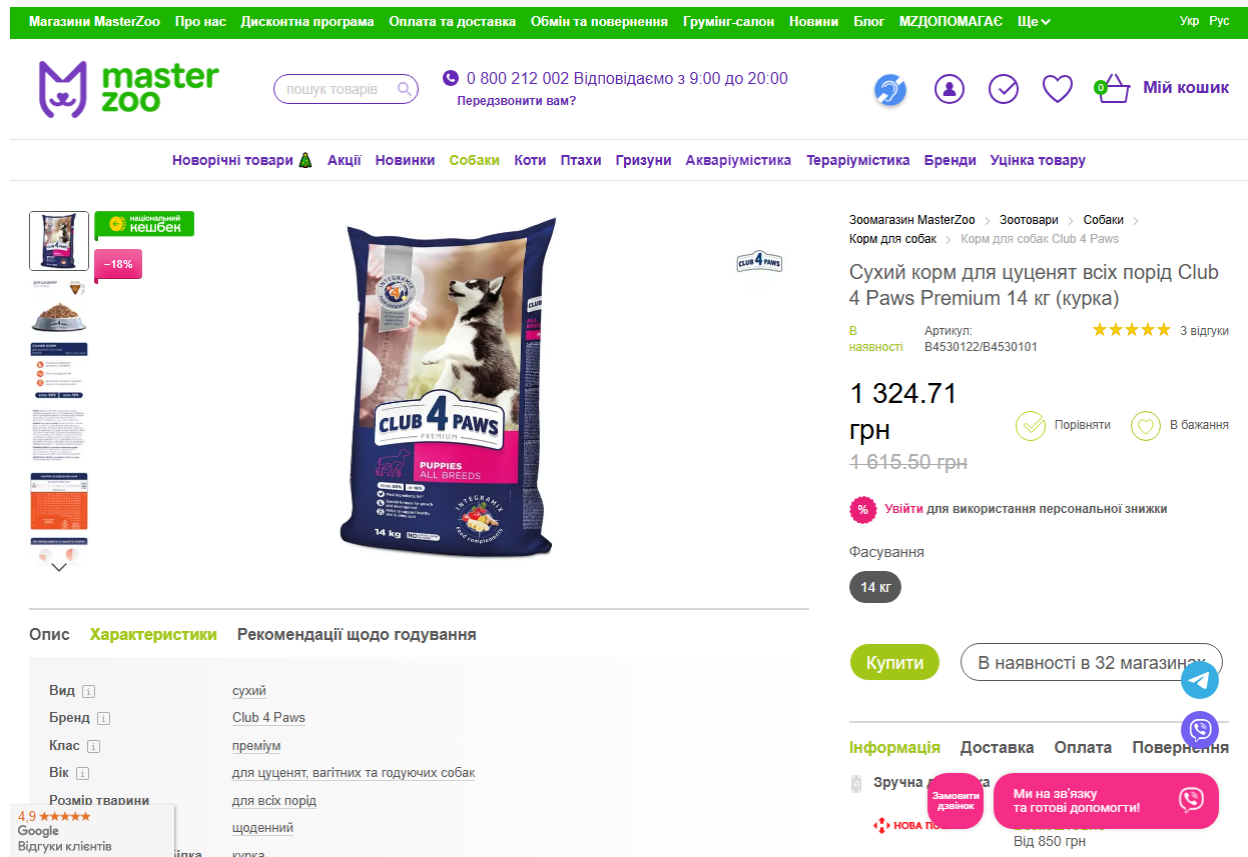


Рисунок 1.1 – Інтерфейс сторінки з даними корму ВС зоомагазину Master ZOO

ВС зоомагазину Master ZOO дозволяє користувачам замовляти корм та інші товари, переглядати виконані замовлення, зберігати корм та інші товари в бажаннях, а тоді переглядати такий перелік бажань. Реалізовано також і можливість пошуку за товарами (рис. 1.2), проте функціонально ця можливість дуже обмежена. Пошук реалізовується фактично за назвою, а всі результати розділяються за типом тварин. При цьому встановити окремі значення або інтервали значень для певних характеристик неможливо. Окрім того не можна налаштувати параметри пошуку окремо, зважаючи на те, що

вони окремо не виставляються, а відповідно не можна такі параметри зберігати або визначати за власною твариною.

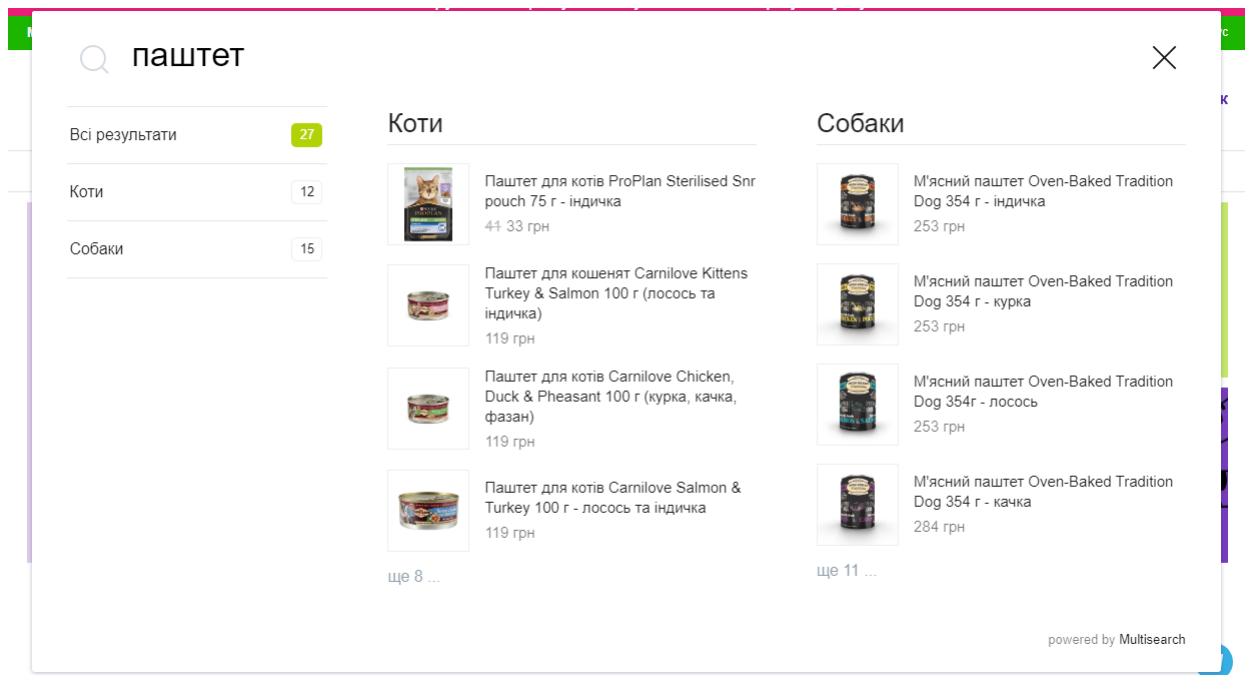


Рисунок 1.2 – Інтерфейс пошуку ВС зоомагазину Master ZOO

У той же час сама сторінка даних про корм містить різні окремі характеристики. Окрім опису, фотографій це також і назва бренду, і основне джерело білка, і вікова група тварини, і тип корму, і призначення, країна виробник, фасування. Тобто характеристики, за якими такий пошук можна було б реалізувати, присутні. Однак сам пошук не реалізовано, а це сприяє тому, що в підсумку кількість результатів є більшою, а користувачу потрібно переглядати окремі результати, вручну визначаючи, чи це підходящий товар, чи ні. Задати ж таку вікову групу для власної тварини, щоб відображались результати тільки для неї, неможливо.

На сторінці товару можна переглянути перелік коментарів від користувачів, залишити власний коментар, оформити замовлення, а також отримати доступ до перегляду доступних варіантів фасувань та обрати потрібний для замовлення. Тобто сама сторінка корму або іншого товару є ключовим елементом взаємодії користувача з вебзастосунком зоомагазину.

ВС зоомагазину Pethouse (рис. 1.3) організований подібно до попереднього рішення, що загалом говорить про подібність логіки роботи та можливість виділення основних можливостей, проте має і певні особливості [7].

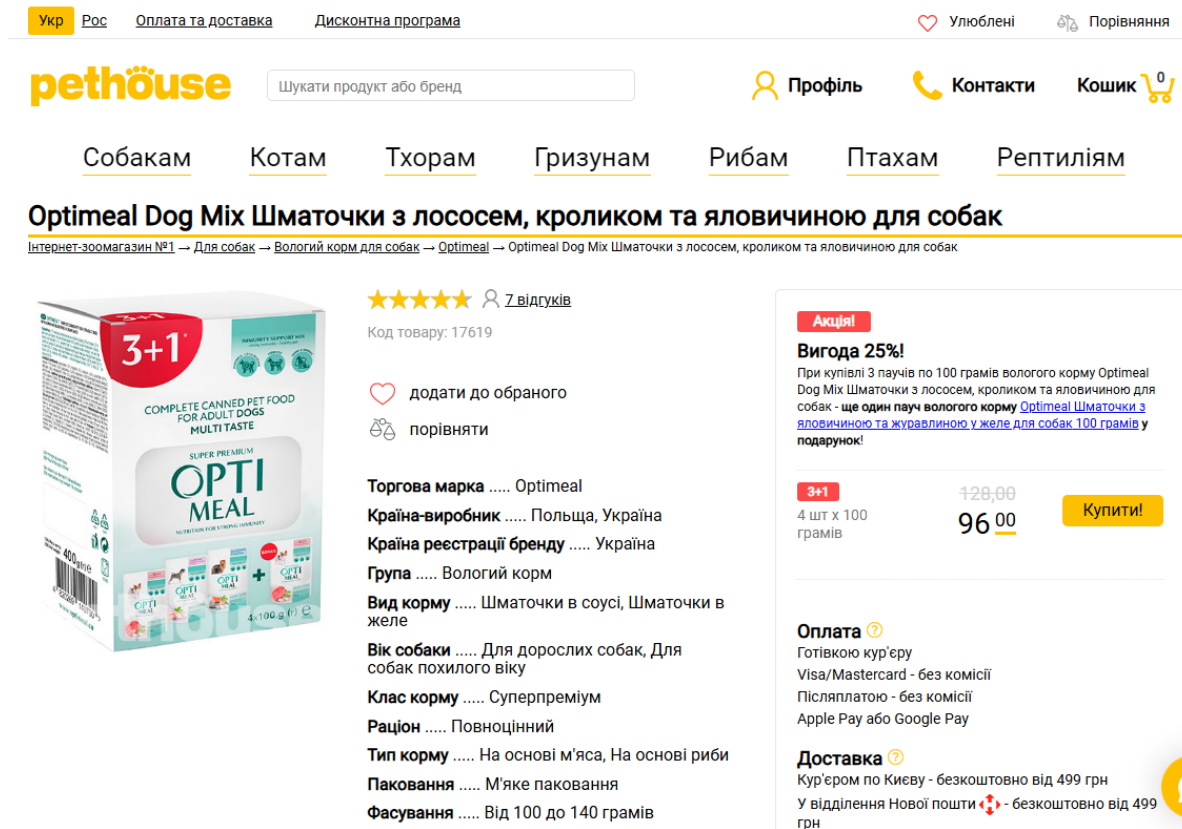


Рисунок 1.3 – Інтерфейс сторінки з даними корму ВС зоомагазину Pethouse

Виділяючи особливості, потрібно вказати на те, що загальний пошук у даному випадку не дуже відрізняється від минулого варіанта, проте обираючи перелік кормів та інших товарів, можна обрати потрібну категорію (наприклад, вологі корми), а тоді у результатах, що відображаються, можна задати конкретні потрібні характеристики (рис. 1.4), що включають бренди, варіанти фасування, тип корму, особливі потреби, вікову групу, вид корму, клас корму.

Укр

Рос

Оплата та доставка

Дисконтна програма

Улюблені

Порівняння

pethöuse

Шукати продукт або бренд

Профіль

Контакти

Кошик

Собакам

Котам

Тхорам

Гризунам

Риbam

Птахам

Рептиліям

Вологий корм для собак — консерви та паучі

Інтернет-зоомагазин №1 → Для собак → Вологий корм → Вологий корм для собак — консерви та паучі

Бренд

Сортувати: за акціями

Рекомендуємо

☐ Brit
 ☐ Savory
 ☐ Miocene
 ☐ Brit Premium
 ☐ Royal Canin
 ☐ Baskerville
 ☐ Carnilove
 ☐ Nature's Protection
 ☐ Hubertus Gold

☐ Almo Nature
 ☐ Alpha Spirit
 ☐ Animonda
 ☐ Belcando
 ☐ Brit Care
 ☐ Brit Fresh
 ☐ Carnie
 ☐ Dolina Noteci
 ☐ Dr. Clauder's
 ☐ Edel

★★★★★ 7 відгуків про товар

Optimeal Dog Mix Шматочки з лососем, кроликом та яловичиною для собак



3+1

додати до обраного

порівняти

Код товару: 17619

Акція!

128,00

4 шт x 100 грамів

96 00

Купити!

При купівлі 3 паучів по 100 грамів вологого корму Optimeal Dog Mix Шматочки з лососем, кроликом та яловичиною для собак - ще один пауч вологого корму Optimeal Шматочки з яловичиною та журавлиною у желе для собак 100 грамів у подарунок!

Набір із 4 апетитних вологих кормів з лососем і лохиною у крем-соусі, з кроликом і чорницею у соусі та з яловичиною та журавлиною у желе для дорослих собак.

Вид корму Шматочки в соусі, Шматочки в желе

Вік собаки Для дорослих собак, Для собак похилого віку

Тварина Собакам

Категорія Вологий корм

Рисунок 1.4 – Інтерфейс пошуку ВС зоомагазину Pethouse

1.2 Порівняння аналогів розробки вебзастосунку підтримки роботи зоомагазину

Основною відмінністю вебзастосунку підтримки роботи зоомагазину, який створюється в роботі, є наявність можливості планування роботи зоомагазину на основі класифікації замовлень стосовно подальшого замовлення вологого корму, що відсутнє у аналогів (табл. 1.1). Окрім того при пошуку кормів забезпечено можливість вибору параметрів, що у аналогів присутня тільки при виконання відтворення переліку кормів. Також забезпечено принцип підтримки роботи зоомагазину на основі підписки, коли корм потім доставляється замовнику з інтервалом у 31 день у подальшому, допоки замовник не відмовиться від підписки.

Таблиця 1.1 – Порівняння аналогів розробки вебзастосунку підтримки роботи зоомагазину

Показник	Master ZOO	Pethouse	Розробка
Пошук кормів	За назвою	За назвою	За параметрами
Відтворення переліку кормів	За параметрами	За параметрами	За параметрами
Замовлення кормів з оформленням підписки	—	—	+
Визначення власних тварин	—	—	+
Пошук кормів з врахуванням характеристик тварин	—	—	+
Класифікація замовлень стосовно подальшого замовлення вологого корму	—	—	+

Ряд особливостей розробки також стосується концепції визначення власних тварин у програмі користувачем, що дозволяє потім враховувати характеристики цих тварин під час виконання пошуку за характеристиками.

1.3 Висновки за розділом 1

Ряд завдань, які необхідно виконати у даній роботі та який має забезпечити досягнення мети, складається з наступних:

- виконати аналіз ПЗ підтримки роботи зоомагазину;
- розробити метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи;
- виконати проєктування ПЗ підтримки роботи зоомагазину;
- реалізувати ПЗ підтримки роботи зоомагазину;

– виконати експериментальне дослідження класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи зоомагазину.

Перше завдання аналізу ПЗ підтримки роботи зоомагазину реалізовано у цьому розділі на основі огляду аналогів, включаючи ВС Master ZOO, Pethouse, визначення відмінностей у функціях розробки, що включають можливості класифікації замовлень стосовно подальшого замовлення вологого корму для планування роботи зоомагазину, визначення власних тварин та виконання пошук на основі характеристик тварини тощо.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача класифікації замовлень стосовно подальшого замовлення вологого корму для планування роботи зоомагазину

Задача класифікації замовлень стосовно подальшого замовлення вологого корму для планування роботи зоомагазину полягає в тому, щоб за кожним новим замовленням визначити, чи в найближчому майбутньому клієнт, який сформував це замовлення, замовить вологий корм з конкретної групи. Тобто встановлюється те, яка сама група вологих кормів розглядається. Також встановлюється період часу, який вважається найближчим. Тобто кількість днів, замовлення через яку ще можна вважати повторним, а не новим окремим замовленням. Це дозволяє за кожним реалізованим замовленням розуміти, чи слід очікувати додаткове замовлення в майбутньому.

Саме вологий корм виділено окремо, тому що сухий корм у такому варіанті зоомагазину розповсюджується за підпискою. Клієнт формує підписку раніше та кожен 31 день вона продовжується. Це дозволяє планувати роботу з сухими кормами наперед. У той же час вологий корм клієнт замовляє додатково, обираючи, який саме варіант вологого корму йому потрібен. При цьому сухий корм є обов'язковою складовою.

Таким чином за кожним виконанням замовленням визначається клас стосовно подальшого можливого замовлення вологого корму. Це дозволяє далі очікувати придбання конкретного варіанта вологого корму або не очікувати. А таким чином відповідно планувати діяльність роботи зоомагазину, розуміючи, скільки майбутніх покупок вологого корму може реалізуватись та скільки стабільних покупок сухого корму слід очікувати.

Виконувану задачу в даній роботі було конкретизовано вибіркою даних про замовлення в зоомагазині [8], за допомогою якої і мало реалізовуватись дослідження. Логіка роботи даного зоомагазину з клієнтами базується на формі підписок. За цією формою власник тварини спочатку оформлює підписку, а потім за нею отримує харчування від зоомагазину. Харчування

надається порціями кожен 31 день. Тобто кожен 31-й день виконується замовлення за підпискою, готуючи його та доставляючи до власника. Так продовжується до моменту, коли власник тварини скасує підписку остаточно. Однак окрім даного варіанта у замовника є також можливість змінити склад замовлення. Харчування складається з обов'язкової частини, яка є сухим харчуванням, до якої власник тварини може додати вологе харчування. У подальшому клієнт може впливати на підписку, наприклад, видаляючи або додаючи навпаки вологе харчування, збільшуючи склад сухого корму, проте базова складова з сухим кормом має весь час забезпечуватися в підписці, тобто є обов'язковою умовою.

Вибірка даних складається з екземплярів, кожен з яких містить значення 36 ознак, які описані таким чином:

- номер ідентифікації замовника (`customer_id`);
- номер ідентифікації тварини (`pet_id`);
- порядковий номер замовлення за даною твариною (`pet_order_number`);
- порядковий номер замовлення за даною твариною серед всіх її замовлень, в яких було визначено вологий корм додатково до сухого (`wet_food_order_number`);
- кількість замовлень, які були оформлені після першого замовлення вологого корму цим користувачем для своєї тварини (`orders_since_first_wet_trays_order`);
- наявність активної підписки на доставку харчування (`pet_has_active_subscription`): деякі покупки здійснюються в зоомагазині без такої підписки, тобто можуть розглядатися як разові;
- рівень корму для тварин, який замовлено в даному замовленні (`pet_food_tier`);
- момент часу реєстрації тварини (`pet_signup_datetime`);
- перелік алергенів, який зазначено за твариною, що дозволяє контролювати, який корм є непридатним для тварини (`pet_allergen_list`);

- перелік улюблених ароматів тварини (pet_fav_flavour_list);
- перелік проблем зі здоров'ям, які має тварина і які було зазначено (pet_health_issue_list);
- позначка того, чи є ця тварина кастрованою чи ні (neutered);
- стать тварини (gender);
- тип породи тварини за розміром (pet_breed_size);
- варіант промоції, через який було реалізовано звернення (signup_promo);
- позначка про те, чи тварина споживала вологий корм раніше (ate_wet_food_pre_tails);
- бренд сухого корму (dry_food_brand_pre_tails);
- стадія життя тварини на той момент, коли було сформовано замовлення (pet_life_stage_at_order);
- дата сплати за замовлення (order_payment_date);
- кілокалорії сухого корму (kibble_kcal);
- кілокалорії вологого корму (wet_kcal);
- сумарна кількість кілокалорій у сформованому замовленні (total_order_kcal);
- кількість позицій вологого корму в замовленні (wet_trays);
- застосований рівень дисконту на вологий корм у відсотках (wet_food_discount_percent);
- обсяг лотка, який має використовуватися для доставки вологого корму, що визначається в грамах (wet_tray_size);
- наявність преміального пакету ласощів у замовленні (premium_treat_packs);
- кількість спеціально застосовуваних пакетів для лікування зубів (dental_treat_packs);
- тип вологого корму, який внесено в замовлення (wet_food_textures_in_order);

- сумарна кількість сесій (авторизацій) власника тварини як користувача на вебсайті (total_web_sessions);
- сумарна кількість сесій (авторизацій) власника тварини як користувача на вебсайті з моменту останнього замовлення (total_web_sessions_since_last_order);
- сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті (total_minutes_on_website);
- сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті з моменту останнього замовлення (total_minutes_on_website_since_last_order);
- сумарна кількість змін в позиціях вологого корму в замовленнях цього користувача для цієї тварини (total_wet_food_updates);
- сумарна кількість змін в позиціях вологого корму в замовленнях цього користувача для цієї тварини з моменту останнього замовлення (total_wet_food_updates_since_last_order);
- час і дата останнього звернення в службу підтримки (last_customer_support_ticket_date);
- категорія звернення в службу підтримки (customer_support_ticket_category).

2.2 Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи передбачає реалізацію послідовності етапів.

На першому етапі дані мають бути вивантажені зі сховища. Таким сховищем зокрема може бути база даних (БД). Кожен екземпляр даних представляє замовлення корму для тварин у зоомагазині на основі структури,

описаної в підрозділі 2.1. Встановлюється перелік типів вологого корму, за якими реалізується класифікація.

Тоді формується вибірка даних під всі типи вологого корму. За кожним замовленням має бути визначено його клас стосовно подальшого замовлення вологого корму заданого типу (кожного з наявного переліку):

- не продовжене замовлення (встановлюється значення 0): таке замовлення протягом заданої кількості днів не призвело до замовлення вологого корму заданого типу;

- подальше замовлення (встановлюється значення 1): таке замовлення протягом заданої кількості днів призвело до замовлення вологого корму заданого типу.

Для цього за кожною твариною переглядаються всі замовлення, виконані для неї, впорядковуючи їх за часом виконання від найбільш раннього. Тоді за цим порядком кожне таке замовлення переглядається і якщо воно містить вологий корм заданого типу, то всі замовлення, які виконувались протягом заданого періоду до нього, але не були раніше іншого замовлення, що містить вологий корм того ж типу, отримують в якості класу для цього типу корму 1. Усі замовлення, для яких не було встановлено 1, отримують 0 в якості значення відповідного класу.

При формуванні такої вибірки ознаки, які містять дати (зокрема дати замовлення та дати останнього звернення до служби підтримки) перетворюються на кількість днів. Це кількість днів, яка пройшла з найбільш раннього замовлення, що є у сховищі, до даного, або яка пройшла з найбільш раннього звернення, що є у сховищі, до даного замовлення.

Для кожного типу вологих кормів на наступному етапі створюється модель за допомогою методу Random Forest [9], що представляє собою ансамбль дерев рішень (рис. 2.1). За результатами дослідження до ансамблю вирішено було включати 50 дерев.

Для кожної моделі формується підвибірка даних, в яку входять екземпляри у такій кількості, щоб екземпляри одного класу за даним типом

вологого корму не більше ніж у 20 разів перевищували екземпляри іншого класу. Далі відкидаються дані про класи інших типів кормів окрім того, що відповідає самій моделі.

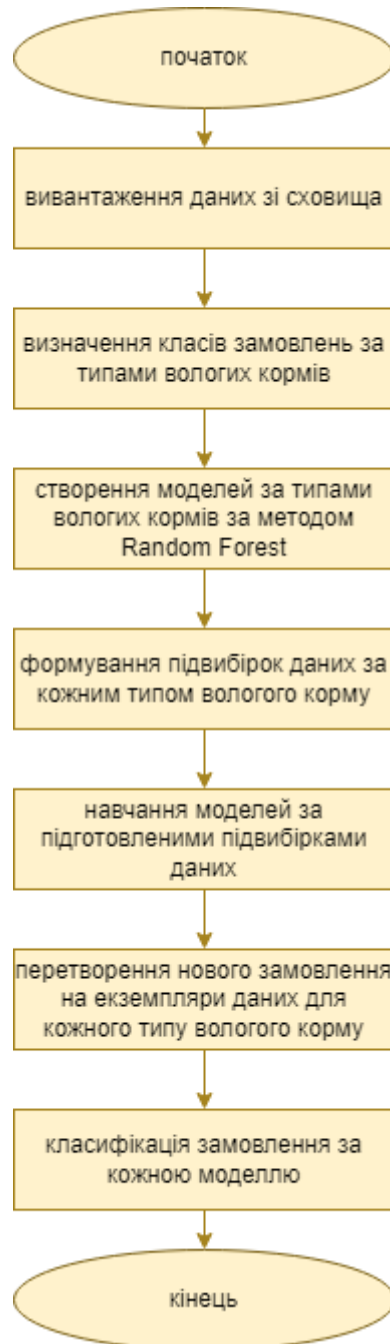


Рисунок 2.1 – Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

Для отриманої підвбірки на основі критерія Пірсона виконується оцінювання всіх вхідних ознак (до них не входить тільки цільова, яка визначає

відповідний клас). Шість ознак, які для відповідного типу вологого корму мають найбільше таке значення, виділяються в якості вхідних ознак самої моделі. Значення цих ознак разом з цільовим класом виділяються з підвибірки даних, а тоді реалізується навчання моделі на основі такої підвибірки. Це повторюється для кожного типу вологого корму. Це повторюється для кожного наявного типу вологого корму.

Останній етап реалізується вже для готових навчених моделей у кількості, що дорівнює кількості типів вологого корму. На даному етапі отримується нове замовлення та для нього реалізуються перетворення, на основі яких формуються значення вхідних ознак для кожної моделі. Тоді кожна модель виконує класифікацію, надаючи на виході значення класів за кожним типом вологого корму для даного замовлення. Даний етап повторюється для кожного нового замовлення.

2.3 Висновки за розділом 2

Наукова новизна роботи полягає в методі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи, який запропоновано за результатами виконання даного розділу та який виконує класифікацію замовлень на основі моделі, побудованої методом Random Forest, визначає виконання відбору ознак для побудови даної моделі для конкретного типу вологого корму, підготовку вибірки даних для відповідного навчання моделі.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови та засобів розробки

Найпотужнішою мовою для реалізації машинного навчання, зокрема роботи з деревами рішень та подібними засобами, додатково роботи з даними на даний момент є Python, тому вибір засобів розробки було зведено до порівняння основних фреймворків цієї мови для веброзробки – Django [10] і Flask [11] (табл. 3.1). Django має багато вбудованих засобів, що підвищують швидкість створення рішень, безпечність, зменшують кількість рядків коду, які мають бути написані. Для того щоб отримати відповідний рівень у Flask потрібно порівнювати альтернативи, обирати, встановлювати та налаштовувати кожне таке рішення окремо. До того ж і сам час розробки триваліший за рахунок інтегрованих підходів. Тому Django виділено для реалізації ПЗ підтримки роботи зоомагазину.

Таблиця 3.1 – Вибір вебфреймворку для розробки ПЗ підтримки роботи зоомагазину

	Django	Flask
Придатність для реалізації машинного навчання	Розробка ведеться мовою Python, доступні всі Python-пакети для роботи з даними та машинного навчання	
Наявні готові засоби розробки	Авторизація, механізм шаблонізації, адміністративна панель, об'єктно-реляційне відображення	Потребує встановлення і налаштування таких засобів, а відповідно і вибору
Час розробки	Пришвидшений	Порівняно більший

3.2 Моделювання роботи з програмою за функціональними вимогами

Функціональні вимоги до ПЗ підтримки роботи зоомагазину розподілено за 2 основними ролями, що включають менеджера як представника зоомагазину та базового користувача вебзастосунку зоомагазину як представника сторони покупців.

Для базового користувача вебзастосунку зоомагазину (покупця) мають надаватися сценарії використання:

- виконання авторизації та як альтернативний сценарій – виконання реєстрації, умовою реалізації якого є необхідність нового облікового запису за відсутності існуючого або потребі в додатковому;
- відтворення власного профілю, що включає сценарій відтворення власних тварин користувача, а альтернативним сценарієм має оновлення даних профілю;
- внесення даних власної тварини, включаючи сценарії визначення алергенів тварини, визначення улюблених смаків тварини та визначення наявних проблем зі здоров'ям у тварини, а альтернативою для даного сценарію є змінювання даних власної тварини для випадку, коли дані тварини у програмі не відповідають актуальному стану;
- відтворення даних власної тварини;
- відтворення повного переліку замовлень кормів за власною твариною;
- відтворення повного переліку замовлень кормів за власними тваринами;
- пошук кормів у зоомагазині;
- відтворення наявних кормів у зоомагазині;
- коментування корму для тварин (рис. 3.1).

- замовлення кормів для харчування тварини, що включає сценарій класифікації замовлення стосовно подальшого замовлення вологого корму;
- керування підпискою на харчування в зоомагазині;
- звернення до служби підтримки споживачів зоомагазину;
- відтворення вмісту власного замовлення;
- відтворення власних звернень до служби підтримки споживачів зоомагазину;
- відтворення переліку кормів, збережених за власною твариною;
- збереження корму для подальших замовлень за твариною, а альтернативою для даного сценарію є скасування збереження корму для подальших замовлень за твариною, що виконується за умови наявності такого збереження;
- відтворення даних корму для тварин, що включає відтворення наявності фасувань корму для тварин;
- керування власним замовленням.

Для менеджера зоомагазину мають надаватися сценарії використання:

- виконання авторизації та як альтернативний сценарій – виконання реєстрації, умовою реалізації якого є необхідність нового облікового запису за відсутності існуючого у менеджера;
- відтворення профілю користувача разом з його власними тваринами;
- відтворення даних тварини користувача вебзастосунку зоомагазину;
- відтворення повного переліку замовлень кормів за твариною;
- відтворення повного переліку замовлень кормів за тваринами користувача вебзастосунку зоомагазину;
- пошук кормів у зоомагазині;
- відтворення наявних кормів у зоомагазині;
- відтворення даних корму для тварин, що включає відтворення наявності фасувань корму для тварин;
- відтворення переліку звернень до служби підтримки споживачів зоомагазину (рис. 3.2).

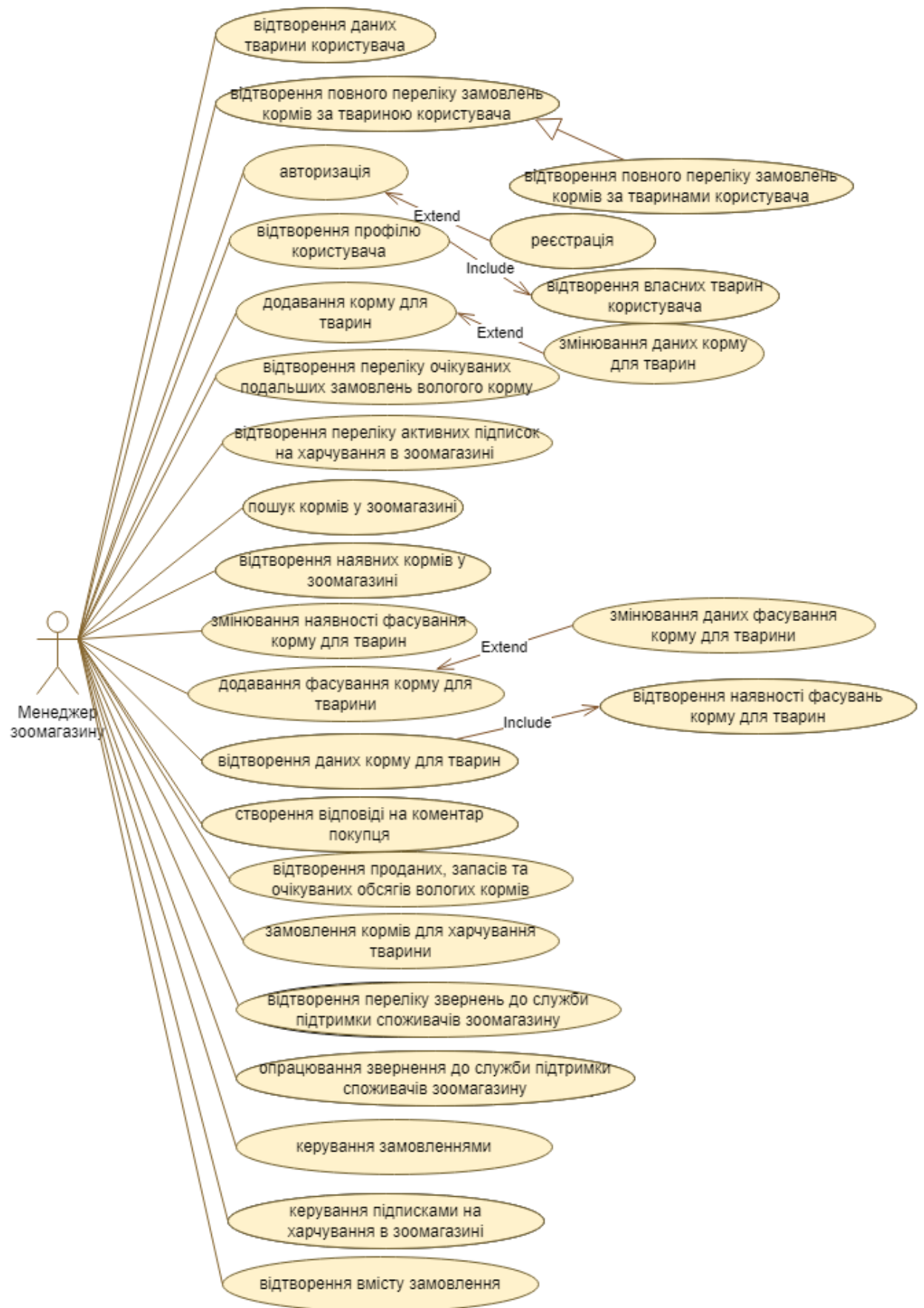


Рисунок 3.2 – Діаграма варіантів використання програми менеджером зоомагазину

Також для менеджера зоомагазину мають забезпечуватись сценарії:

- опрацювання звернення до служби підтримки споживачів зоомагазину;
- відтворення вмісту замовлення;
- замовлення кормів для харчування тварин;
- відтворення переліку активних підписок на харчування в зоомагазині;
- відтворення переліку очікуваних подальших замовлень вологого корму;
- додавання фасування корму для тварини, а альтернативою для даного сценарію є змінювання даних фасування корму для тварини, коли фасування змінилось;
- змінювання наявності фасування корму для тварин;
- додавання корму для тварин, а альтернативою для даного сценарію є змінювання даних корму для тварин, коли дані змінились;
- керування підписками на харчування в зоомагазині;
- створення відповіді на коментар покупця;
- відтворення проданих, запасів та очікуваних обсягів вологих кормів;
- керування замовленнями.

3.3 Проєктування структури бази даних

БД сконструйована на основі системи PostgreSQL. Створена структура подана нижче.

Збереження облікових записів користувачів вебзастосунку зоомагазину вирішено виконати в БД таблицею Petstoreuser з полями:

- id – номер ідентифікації користувача вебзастосунку зоомагазину;
- surnameandname – прізвище та ім'я (або альтернативний підпис) для відображення користувача у вебзастосунку зоомагазину;
- registereddate – дата, а також і час, коли було зареєстровано обліковий запис користувача вебзастосунку зоомагазину;

- lastvisitdate – дата, а також і час, коли було востаннє авторизовано обліковий запис користувача вебзастосунку зоомагазину;
- loginacc – логін користувача вебзастосунку зоомагазину;
- passwordacc – пароль користувача вебзастосунку зоомагазину;
- phoneforcontact – номер телефону для контактів з користувачем вебзастосунку зоомагазину;
- emailforcontact – електронна пошта для контактів з користувачем вебзастосунку зоомагазину;
- storerepresenter – тип користувача вебзастосунку зоомагазину: покупець або представник самого зоомагазину;
- promotype – тип промоції, за якою було виконано реєстрацію облікового запису користувача зоомагазину (самостійно, за рекомендацією друга, за рекомендацією ветеринара тощо);
- discountlevel – відсоток дисконту, який діє для даного облікового запису користувача зоомагазину.

Типи тварин, з якими працює зоомагазин, визначають котів, собак, пташок, гризунів і для збереження відповідних даних винесені в таблицю БД Pettype з полями:

- id – номер ідентифікації типу тварини, з якою працює зоомагазин;
- pettypename – назва типу тварини (тобто конкретно собака, кіт, пташка тощо).

Збереження порід тварин, з якими потенційно працює зоомагазин, вирішено виконати в БД таблицею Petbreed з полями:

- id – номер ідентифікації породи тварини, з якою потенційно працює зоомагазин;
- pettype – тип тварини, породою якої є дана (зовнішній ключ до таблиці Pettype);
- correctbreed – назва породи тварини;
- breedsizelevel – категорія розміру тварини, до якої належить ця порода (маленька, середня, велика).

Збереження власних тварин користувачів вебзастосунку зоомагазину вирішено виконати в БД таблицею Ownpet з полями:

- id – номер ідентифікації тварини користувача вебзастосунку зоомагазину;
- petname – ім'я цієї тварини користувача вебзастосунку зоомагазину (необхідно фактично для того, щоб її власник міг швидко ідентифікувати тварину в програмі, якщо він має їх декілька);
- petbreed – порода цієї тварини користувача вебзастосунку зоомагазину (зовнішній ключ до таблиці Petbreed);
- petuserowner – власник тварини, визначений як користувач вебзастосунку зоомагазину (зовнішній ключ до таблиці Petstoreuser);
- gender – стать цієї тварини користувача вебзастосунку зоомагазину;
- petbirth – приблизна або точна дата народження цієї тварини користувача вебзастосунку зоомагазину;
- neutered – позначка про те, чи є ця тварина користувача вебзастосунку зоомагазину кастрованою, чи ні;
- wetfoodate – позначка про те, чи споживала ця тварина користувача вебзастосунку зоомагазину вологий корм;
- activesubscription – наявність активно діючої підписки на доставку харчування з зоомагазину для цієї тварини користувача;
- registereddate – дата, а також і час, коли цей користувач вебзастосунку зоомагазину зареєстрував у програмі свою тварину.

Збереження алергенів для конкретної тварини користувача вебзастосунку зоомагазину вирішено виконати в БД таблицею Ownpetallergen з полями:

- id – номер ідентифікації запису про алерген конкретної тварини користувача вебзастосунку зоомагазину;
- ownpet – тварина, в якій ця речовина викликає алергію (є алергеном, зовнішній ключ до таблиці Ownpet);

– foodcomponent – речовина, яка є алергеном для цієї тварини (зовнішній ключ до таблиці Foodcomponent).

Таблиця Ownpetallergen є реалізацією для реляційної БД зв'язку між таблицями Ownpet і Foodcomponent, що належить до «багато-до-багатьох».

Збереження смаків, які до вподоби конкретній тварині, вирішено виконати в БД таблицею Ownpetflavour з полями:

– id – номер ідентифікації смаку конкретної тварини користувача вебзастосунку зоомагазину;

– ownpet – тварина, якій до вподоби цей смак (зовнішній ключ до таблиці Ownpet);

– flavour – смак, який до вподоби цій тварині користувача вебзастосунку зоомагазину (зовнішній ключ до таблиці Flavour).

Таблиця Ownpetflavour є реалізацією для реляційної БД зв'язку між таблицями Ownpet і Flavour, що належить до «багато-до-багатьох».

Збереження кормів для тварин у зоомагазині, які до вподоби конкретній тварині, вирішено виконати в БД таблицею Ownpetstorefood з полями:

– id – номер ідентифікації корму для тварин, що є в цьому зоомагазині;

– ownpet – тварина, якій до вподоби цей корм (зовнішній ключ до таблиці Ownpet);

– storefood – корм, який до вподоби цій тварині (зовнішній ключ до таблиці Petstorefood).

Таблиця Ownpetstorefood є реалізацією для реляційної БД зв'язку між таблицями Ownpet і Petstorefood, що належить до «багато-до-багатьох».

Збереження виробників кормів для тварин, з якими працює зоомагазин, вирішено виконати в БД таблицею Petstorefoodbrand з полями:

– id – номер ідентифікації виробника кормів для тварин, з яким працює зоомагазин;

– foodbrandname – назва виробника кормів для тварин, з яким працює зоомагазин.

Збереження типів (певних профілів) вологого корму, що є в зоомагазині на продажу, вирішено виконати в БД таблицею Wetfoodtype з полями:

- id – номер ідентифікації типу вологого корму, що є в зоомагазині на продажу;
- wetfoodname – назва типу вологого корму, що є в зоомагазині на продажу.

Збереження кормів для тварин у зоомагазині вирішено виконати в БД таблицею Petstorefood з полями:

- id – номер ідентифікації корму для тварин, що є в цьому зоомагазині;
- foodtitle – скорочена назва корму для тварин у зоомагазині;
- foodcaption – повна назва, якою підписується цей корм для тварин у зоомагазині;
- pettypefood – тип тварин, для яких призначений цей корм (зовнішній ключ до таблиці Pettype);
- foodtype – загальний тип корму для тварин, до якого належить цей корм: сухий або вологий;
- wetfoodtype – тип вологого корму, до якого належить цей корм (зовнішній ключ до таблиці Wetfoodtype);
- storefoodbrand – виробник цього корму для тварин (зовнішній ключ до таблиці Petstorefoodbrand);
- breedszelevel – розмір породи, для якого може використовуватися цей корм для тварин (або будь-який, або вказаний конкретний);
- countryproduced – країна, в якій виготовлено цей корм для тварин;
- proteincomponent – основне джерело білка (зовнішній ключ до таблиці Foodcomponent);
- mainfavour – основний смак корму (зовнішній ключ до таблиці Flavour);
- onpurpose – основне призначення цього корму для тварин, яке виражається або цільовим використанням, або звичайним щоденним;

- classtier – клас, до якого належить цей корм для тварин, що може бути економ, медіум, преміум, супер-преміум;
- calories – кількість кілокалорій, яку можна отримати при споживанні 100 г даного корму;
- detaildescription – детальний опис корму для тварин.

Збереження проблем зі здоров'ям тварин, за яких рекомендовано або навпаки не рекомендовано використання корму, вирішено виконати в БД таблицю Petstorefoodhealthissue з полями:

- id – номер ідентифікації проблем зі здоров'ям, за яких рекомендовано або навпаки не рекомендовано використання даного корму з зоомагазину;
- storefood – корм для тварин, що є в цьому зоомагазині та для якого співставлено цю проблему зі здоров'ям тварин (зовнішній ключ до таблиці Petstorefood);
- healthissue – проблема зі здоров'ям, за якої рекомендовано або не рекомендовано використання даного корму (зовнішній ключ до таблиці Healthissue);
- positive – позначка, яка визначає за правдивого значення, що цей корм рекомендовано використовувати при цій проблемі зі здоров'ям у тварини, а за хибного значення відповідно, що не рекомендовано використовувати цей корм взагалі.

Таблиця Petstorefoodhealthissue є реалізацією для реляційної БД зв'язку між таблицями Petstorefood і Healthissue, що належить до «багато-до-багатьох».

Збереження рівнів зрілості тварин, за яких можливе споживання конкретного корму з зоомагазину, вирішено виконати в БД таблицю Petstorefoodmature з полями:

- id – номер ідентифікації рівня зрілості тварин, за якого можливе споживання конкретного корму з зоомагазину;
- storefood – корм для тварин з зоомагазину, який можна споживати за даного рівня зрілості тварини (зовнішній ключ до таблиці Petstorefood);

– `maturelevel` – рівень зрілості тварини, за якого можна споживати цей корм для тварин.

Збереження можливих фасувань, за якими і продається корм для тварин у зоомагазині, вирішено виконати в БД таблицею `Petstorefoodtray` з полями:

- `id` – номер ідентифікації фасування корму для тварин;
- `storefood` – корм для тварин, що продається в цьому зоомагазині та для якого визначено це фасування (зовнішній ключ до таблиці `Petstorefood`);
- `trayweight` – вага даного фасування корму для тварин;
- `baseprice` – стандартна вартість даного корму для тварин у цьому фасуванні;
- `currentprice` – вартість даного корму для тварин у цьому фасуванні, за якою його можна купити в поточний момент (знижена вартість за умовами акцій);
- `availablepositions` – кількість цього корму в цьому фасуванні, яка доступна для продажу;
- `topdiscount` – максимальний дисконт (у відсотках), який може бути застосований при продажу цього корму в цьому фасуванні.

Збереження фотографій корму для тварин вирішено виконати в БД таблицею `Petstorefoodphoto` з полями:

- `id` – номер ідентифікації фотографії корму для тварин, доступного у цьому зоомагазині;
- `storefood` – корм для тварин, який представлений на фотографії (зовнішній ключ до таблиці `Petstorefood`);
- `photo` – сама фотографія корму для тварин (завантажений на сервер файл);
- `isposter` – позначка про те, чи ця фотографія корму для тварин використовується як постер для цього корму, тобто як базова фотографія зокрема в результатах відображення різних кормів, чи це додаткова фотографія (зі всіх фотографій для одного корму може бути тільки одна встановлена як постер).

Збереження коментарів користувачів вебзастосунку зоомагазину про корми для тварин вирішено виконати в БД таблицею Petstorefoodcomment з полями:

- id – номер ідентифікації коментаря користувача вебзастосунку зоомагазину про корм для тварин;
- storefood – корм для тварин, до якого відноситься коментар (зовнішній ключ до таблиці Petstorefood);
- petstoreuser – користувач вебзастосунку зоомагазину, який зберіг цей коментар (зовнішній ключ до таблиці Petstoreuser);
- commentcontent – текст коментаря, який користувач вебзастосунку зоомагазину залишив про цей корм;
- reactioncomment – текст відповіді на коментар, який користувач вебзастосунку зоомагазину залишив про цей корм, створений представником зоомагазину;
- maincommenttime – дата, а також і час, коли було створено коментар користувачем вебзастосунку зоомагазину про корм для тварин;
- maincommenttime – дата, а також і час, коли було створено відповідь на коментар користувачем вебзастосунку зоомагазину про корм для тварин представником зоомагазину.

Збереження смаків, доступних до вибору під час роботи з кормами зоомагазину, вирішено виконати в БД таблицею Flavour з полями:

- id – номер ідентифікації смаку;
- flavourname – назва смаку.

Збереження складу корму для тварин вирішено виконати в БД таблицею Petstorefoodcomponent з полями:

- id – номер ідентифікації частини складу корму для тварин;
- storefood – корм для тварин, вміст речовин до якого визначено (зовнішній ключ до таблиці Petstorefood);
- foodcomponent – речовина, яка входить до цього корму для тварин (зовнішній ключ до таблиці Foodcomponent);

– percentfromall – відсоток вмісту даної речовини у складі цього корму для тварин;

– descriptioncomment – опис особливостей вмісту цієї речовини до складу цього корму для тварин, якщо такі особливості є.

Збереження речовин, які можуть входити до складу кормів для тварин, які продаються в цьому зоомагазині, вирішено виконати в БД таблицею Foodcomponent з полями:

– id – номер ідентифікації речовини, яка може входити до складу кормів для тварин, які продаються в цьому зоомагазині;

– componentname – назва речовини, яка може входити до складу кормів для тварин, які продаються в цьому зоомагазині.

Збереження замовлень кормів з зоомагазину вирішено виконати в БД таблицею Petstoreorder з полями:

– id – номер ідентифікації замовлення кормів з зоомагазину;

– ownpet – тварина, для якої її власник як користувач вебзастосунку зоомагазину замовив корми (зовнішній ключ до таблиці Ownpet);

– suscriptionorder – позначка про те, що це замовлення є або не є реалізацією підписки на забезпечення харчування тварини зоомагазином;

– typicalsubscription – позначка про те, що це замовлення має в подальшому використовуватися як основа (шаблон) для реалізації активної підписки;

– timeordered – дата, а також і час, коли було оформлено це замовлення кормів з зоомагазину для власної тварини;

– timefordelivery – дата, а також і час, коли має бути доставлено (виконано) це замовлення кормів з зоомагазину для тварини;

– delivaddr – адреса доставки замовленого корму для тварин;

– pricepayed – вартість, яку було сплачено за замовлення кормів з зоомагазину для тварини;

– transactionnmbr – номер транзакції, через яку було виконано оплату замовлення;

– discountused – дисконт у відсотках, який було використано для сплати за це замовлення;

– repeatwaited – результат класифікації замовлення щодо подальшого замовлення вологого корму (число інтерпретується як набір 1 та 0, що відповідають результату класифікації за кожним типом вологого корму);

– statusorder – стан замовлення кормів з зоомагазину.

Збереження складу замовлень кормів з зоомагазину, адже в замовлення може входити декілька різних позицій, вирішено виконати в БД таблицею Petstoreordercomponent з полями:

– id – номер ідентифікації складу замовлення кормів з зоомагазину;

– petstoreorder – замовлення кормів з зоомагазину, до якого відноситься ця складова (зовнішній ключ до таблиці Petstoreorder);

– storefoodtray – фасування корму для тварин, яке є складовою цього замовлення кормів з зоомагазину (зовнішній ключ до таблиці Petstorefoodtray);

– trayquantity – кількість одиниць цього корму в даному фасуванні, яку було замовлено;

– discountused – використаний дисконт для визначення вартості одиниці цього фасування корму у відсотках;

– componentprice – фактична вартість одиниці цього фасування корму в замовленні (після застосування дисконту).

Збереження можливих проблем зі здоров'ям, що доступні у вебзастосунку, вирішено виконати в БД таблицею Healthissue з полями:

– id – номер ідентифікації можливої проблеми зі здоров'ям тварини;

– healthissuetitle – назва можливої проблеми зі здоров'ям тварини.

Збереження проблем зі здоров'ям конкретних тварин їхніми власниками вирішено виконати в БД таблицею Ownpethealthissue з полями:

– id – номер ідентифікації проблеми зі здоров'ям тварини власника, який є користувачем вебзастосунку зоомагазину;

– ownpet – тварина, яка має цю проблему зі здоров'ям (зовнішній ключ до таблиці Ownpet);

– healthissue – проблема зі здоров’ям, яка була виявлена у цієї тварини (зовнішній ключ до таблиці Healthissue).

Таблиця Ownpethealthissue є реалізацією для реляційної БД зв’язку між таблицями Ownpet і Healthissue, що належить до «багато-до-багатьох».

Збереження категорій, за якими можуть відбуватися звернення до служби підтримки споживачів у зоомагазині, вирішено виконати в БД таблицею Customersupportticketcategory з полями:

- id – номер ідентифікації категорії звернення до служби підтримки споживачів у зоомагазині;
- csticketcategoryname – назва категорії звернення до служби підтримки споживачів у зоомагазині.

Збереження звернень до служби підтримки споживачів у зоомагазині вирішено виконати в БД таблицею Customersupportticket з полями:

- id – номер ідентифікації звернення до служби підтримки споживачів у зоомагазині;
- csticketcategory – категорія, до якої належить звернення до служби підтримки споживачів у зоомагазині (зовнішній ключ до таблиці Customersupportticketcategory);
- petstoreuser – користувач вебзастосунку зоомагазину, який звернувся до служби підтримки (зовнішній ключ до таблиці Petstoreuser);
- ticketcontent – суть звернення до служби підтримки споживачів у зоомагазині;
- ticketanswercontent – суть відповіді на звернення до служби підтримки споживачів у зоомагазині;
- answerpetstoreuser – користувач вебзастосунку зоомагазину, який є представником магазину та який надав відповідь на звернення до служби підтримки (зовнішній ключ до таблиці Petstoreuser);
- tickettimecreated – дата, а також і час, коли було подано звернення до служби підтримки зоомагазину;

- petstoreuser – користувач вебзастосунку зоомагазину, який відвідав вебзастосунок зоомагазину (зовнішній ключ до таблиці Petstoreuser);
- timevisit – дата, а також і час, коли таке відвідування було зафіксовано;
- urlopened – посилання, яке було відкрито користувачем вебзастосунку зоомагазину;
- newvisit – відвідування, яке відкриває нову сесію під час роботи з вебзастосунком або яке продовжує роботу через відкриття наступного посилання за тієї ж самої авторизації.

3.4 Висновки за розділом 3

За результатами успішного виконання даного розділу запроєктовано ПЗ, включаючи вибір мови Python з пакетами для роботи з даними і машинного навчання та вебфреймворку Django, моделювання сценаріїв роботи менеджера зоомагазину та покупця, визначення структури БД.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Опрацювання замовлень корму для тварин у вебзастосунку

Замовлення корму для тварин у вебзастосунку об'єднує у взаємодії сторону клієнтів за стороною менеджерів (рис. 4.1).

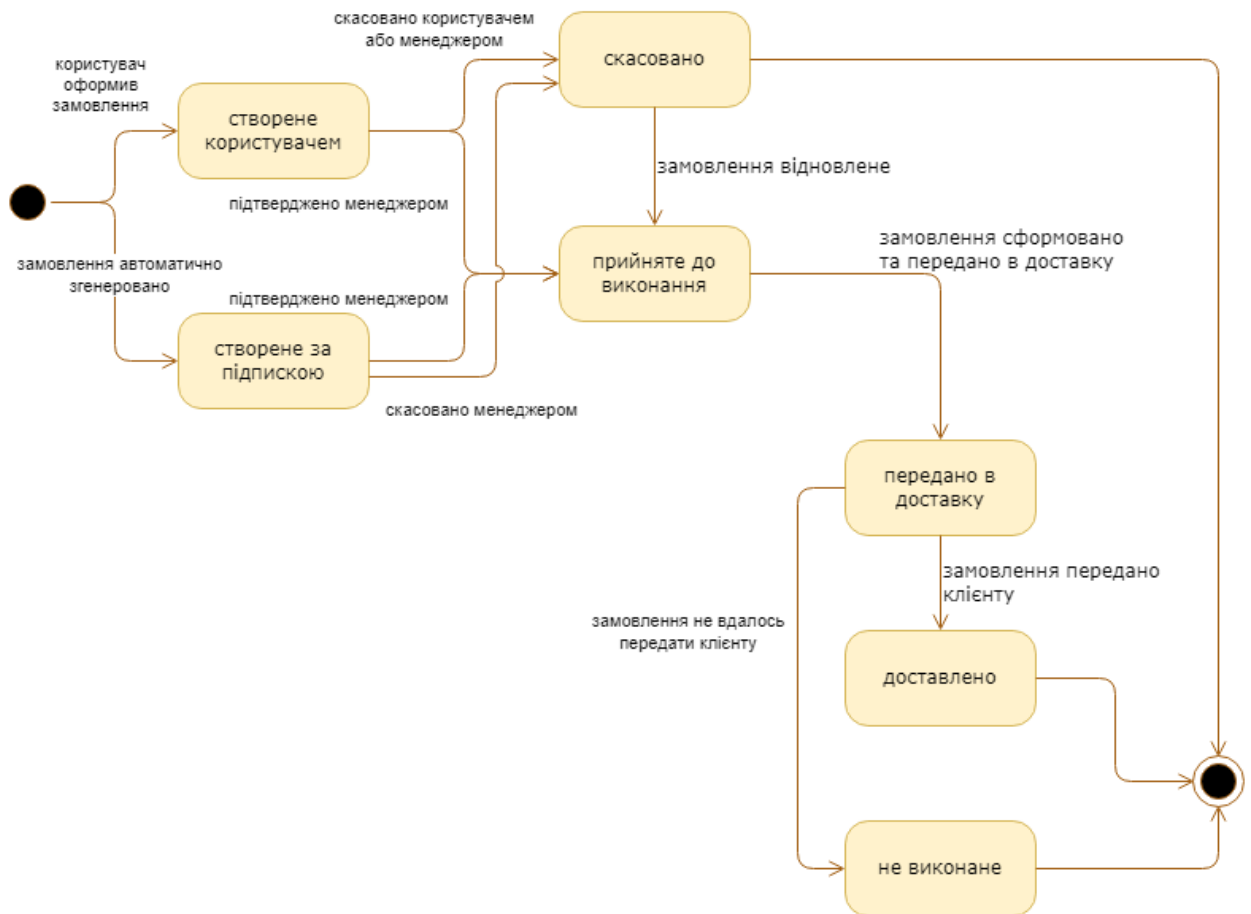


Рисунок 4.1 – Діаграма станів замовлення корму для тварин

Є певний порядок роботи за успішного виконання, проте перехід до кожного наступного етапу передбачає інформування обох сторін про це. Клієнт повинен розуміти, як відбувається робота над його замовленням та які замовлення відповідно ще не опрацьовані, які виконуються, а які вже завершені. Менеджер зоомагазину повинен розуміти, які замовлення на яких етапах знаходяться та що він може зробити в поточний момент щодо тих замовлень, які ще не завершено. Тому замовлення корму для тварин у

зоомагазині є зв'язуючою сутністю, а стани цього замовлення визначають порядок роботи у вебзастосунку зокрема щодо тих функцій, які для них доступні та що саме ці функції можуть виконувати.

4.2 Рішення з реалізації вебзастосунку

Розроблений вебзастосунок структурно представлений з основних наступних одиниць:

- моделі вебзастосунку, що реалізують класи, які забезпечують створення відповідних таблиць БД;
- представлення вебзастосунку, що реалізовані функціями, які забезпечують виділення даних та заповнення даними шаблонів вебсторінок для відображення результатів виконання запланованих функцій користувачу;
- шаблони вебсторінок, які визначають структуру тих вебсторінок, які у відповідь на виконання відповідної функції користувачем демонструються йому;
- скрипт для витягання з БД даних для формування вибірки, опрацювання цих даних та збереження відповідної вибірки;
- скрипт для виконання навчання моделі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи, який має запускатися за відповідною командою користувачем;
- скрипт для автоматичного генерування замовлень корму для тварин за підпискою, який автоматично запускається кожен день.

Безпосередньо класифікація замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи зоомагазину на основі навченої моделі реалізується під час збереження даних у БД, коли користувач сформував замовлення. Тоді вибираються відповідні значення вхідних даних, подаються на вхід, а результат класифікації фіксується у БД, після чого відповідні накопичені дані відображаються менеджеру зоомагазину на відповідних вебсторінках.

Для реалізації скрипта для виконання навчання моделі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму використано наступні пакети.

Пакет `Numpy` головним чином використаний для опрацювання процесу перетворення інтервалу часу на кількість повних днів.

Пакет `Pandas` забезпечував, по-перше, прочитання даних з `Comma-Separated Values (CSV)`-файлу, де дані початково зберігались у вибірці даних, яка використовувалась для навчання, а також зберігались у подальшому за результатами роботи скрипта для витягання з БД даних для формування вибірки, опрацювання цих даних та збереження відповідної вибірки. Для цього використана функція `read_csv`. По-друге, на основі даного пакета формувався об'єкт класу `DataFrame` у вигляді таблиці, за допомогою якої і реалізовувалась робота з даними.

Пакет `Scikit-learn` містить клас `RandomForestClassifier`, який використаний для створення моделі з визначенням через параметр `n_estimators` кількості дерев у методі `Random Forest`. Навчання виконано з передачею `DataFrame`-об'єктів до методу `fit` даної моделі, а сама класифікація нових об'єктів та об'єктів під час тестування виконана через виклик методу `predict`.

Представлення відтворення даних тварини користувача вебзастосунку зоомагазину виконано функцією `ownpet_view`, яка має параметр `request`, який є стандартним для всіх представлень, та `ownpet_id` з номером ідентифікації самої тварини. У представленні реалізується перевірка, що ця тварина належить користувачу, який авторизувався. За твариною окрім базових даних вибираються також дані про її алергени, улюблені смаки, збережені корми, проблеми зі здоров'ям. Усі ці дані містяться в об'єкті за ключем `ownpet` з контексту даних, який передається в результаті роботи цієї функції для заповнення шаблону вебсторінки `ownpet.html`. За ключем `ownpets` знаходиться перелік всіх тварин користувача, який авторизувався, що дозволяє за необхідності перейти до іншої власної тварини. За ключем `orders` знаходяться 5 останніх замовлень кормів для цієї тварини. За ключем `neworders`

знаходиться кількість нових замовлень кормів за цією твариною, тобто таких, які не були ще успішно виконані. Ще один ключ називається `message` та містить текст повідомлення про помилку. При виконанні звернення забезпечується збереження запису до БД про відвідування сторінки на основі класу `Petstoreuservisit`. Та ж сама ідея використовуються і в інших представленнях. А у методі підтримки авторизації реалізується створення об'єкта того ж класу, але з позначкою нового відвідування.

Представлення відтворення профілю користувача разом з його власними тваринами реалізовано функцією `pet_store_user_profile_view`, при цьому вхідні дані визначаються авторизацією користувача вебзастосунку зоомагазину.

Представлення внесення даних власної тварини користувачем вебзастосунку зоомагазину реалізовано функцією `own_pet_add_view` на основі підтримки класів форм за структурою відповідних моделей.

Представлення відтворення повного переліку замовлень кормів за твариною виконано функцією `ownpet_orders_view`. Номер ідентифікації тварини передається через параметр `request` цієї функції в атрибуті `GET`, який дозволяє отримати доступ до всіх параметрів, переданих через `GET`-запит до сторінки. Параметр `ownpet` з цього атрибута містить номер ідентифікації тварини. Параметр `status` дозволяє визначити стан замовлень, які мають бути відображені. Виділені таким чином замовлення розділяються для відображення по 10 на сторінці. Номер сторінки передається через параметр `page`. Усі результати передаються у шаблон вебсторінки `ownpetorders.html` у вигляді словника, що містить ключі `ownpet` з даними самої тварини, `ownpets` зі всіма власними тваринами користувача, `orders` зі сторінкою з виділеними замовленнями, `orders_all` з сумарною кількістю замовлень, `orders_number` з сумарною кількістю замовлень, що відповідають заданим станам, `proc_orders_number` з незавершеними замовленнями, `message` з повідомленням про помилку. Шаблони сторінок створені на базі безкоштовного шаблону [12].

Відтворення повного переліку замовлень кормів за тваринами користувача вебзастосунку зоомагазину реалізовано представленням через функцію `petstore_orders_view`. При цьому відбираються всі замовлення у зоомагазині, які стосуються тварин, які належать авторизованому користувачу вебзастосунку зоомагазину.

Представлення пошуку кормів у зоомагазині реалізовано функцією `pet_store_food_search_view`. Усі параметри пошуку задаються через GET-запит, а тому отримуються через параметр `request` цієї функції, включаючи зокрема `brands` для визначення переліку з номерами ідентифікації виробників, розділених комами, `title` для визначення назви корму для тварин або її частини, `flav` для визначення назви основного смаку корму, `ftypes` для визначення переліку типів кормів (вологий, сухий або обидва одразу), `wtypes` для визначення переліку типів вологого корму, якщо серед варіантів у попередньому параметрів є вологий корм, `breedsize` для визначення категорій порід за розміром, перелічених через кому, `countries` для визначення переліку країн виготовлення корму, розділених через кому, `protein` для визначення переліку основних джерел білка, розділених через кому, `classtier` для визначення класів корму, розділених через кому, `calorymin` для визначення мінімальної кількості калорій, `calorymax` для визначення максимальної кількості калорій, `health` для визначення проблем зі здоров'ям (`positive` визначає необхідність позитивного впливу на ці хвороби), розділених комами, `mature` для визначення рівнів зрілості тварин, для яких призначений корм, розділених комами, `comproments` для визначення переліку речовин у вмісті корму, розділених через кому, `allergens` для визначення переліку алергенів, які не мають входити до складу корму, розділених комами, `order` для визначення напрямку сортування результатів, `page` для визначення номеру сторінки з результатами (відображення по 10 на сторінці). Забезпечується також індивідуалізований пошук кормів для придбання для власної тварини користувача вебзастосунку зоомагазину. Тварина визначається номером ідентифікації, що задається параметром запиту `ownpet`.

Відтворення наявних кормів у зоомагазині за обраним типом тварин та у відповідності з обраною власною твариною користувача вебзастосунку зоомагазину реалізовано функцією представлення `pet_type_food_view`, що влаштована подібно до того, як організовані параметри у попередній функції.

Представлення змінювання даних власної тварини користувачем вебзастосунку зоомагазину реалізовано функцією `own_pet_edit_view`, де параметром `ownpetId` визначається номер ідентифікації тварини, реалізуючи перевірку того, що тварина з вказаним номером ідентифікації належить користувачу вебзастосунку зоомагазину, який авторизувався.

Представлення збереження корму для подальших замовлень за твариною реалізовано сумісно зі скасуванням на основі функції `pet_food_save`, що має параметр `petfoodId`, який визначає номер ідентифікації корму для тварин, який зберігається або вилучається користувачем вебзастосунку зоомагазину як власником тварини, перевіряючи, що він є власником тварини з номером ідентифікації `ownpetId`, що є ще одним параметром функції.

Представлення відтворення переліку кормів, збережених за твариною, реалізовано функцією `own_pet_saved_food_view`, яка має параметр `ownpetId`, що задає номер ідентифікації самої тварини, при цьому дані передаються відібрані тільки тоді, коли власником тварини є авторизований користувач вебзастосунку зоомагазину. Також до результатів у шаблон вебсторінки `ownpetfoodsv` передається перелік самих тварин користувача.

Представлення відтворення даних корму для тварин реалізовано функцією `pet_food_view`. Номер ідентифікації корму для тварин передається через параметр `petfood` з GET-запиту. До результатів входять дані корму для тварин, перелік фасувань, перелік проблем зі здоров'ям у тварин, на які впливає цей корм, перелік фотографій, перелік коментарів до корму до тварин, склад корму для тварин. Таким чином реалізовано сценарій відтворення наявності фасувань корму для тварин.

Представлення змінювання наявності фасування корму для тварин реалізовано функцією `pet_store_food_tray_change_view`, де параметр `trayId`

визначає номер ідентифікації фасування, а параметр `number` визначає оновлену кількість. Представлення змінювання даних фасування корму для тварини реалізовано функцією `pet_store_food_tray_edit_view`, де параметр `trayId` визначає номер ідентифікації фасування. Представлення додавання фасування корму для тварини реалізовано функцією `pet_store_food_tray_add_view` за допомогою тієї ж форми, що і в попередньому представленні, на основі класу відповідної моделі з визначенням номеру ідентифікації корму для тварин через параметр `petfoodId`.

Представлення додавання корму для тварин реалізовано функцією `pet_store_food_add_view` на основі форми за класом моделі. Представлення змінювання даних корму для тварин реалізовано функцією `pet_store_food_edit_view` з визначенням номеру ідентифікації корму `petfoodId`.

Представлення коментування корму для тварин обробляє коментар, введений користувачем, та за наявності вказаного значення параметра `commentId` прив'язує його як відповідь на існуючий коментар, якщо робота виконується для авторизованого користувача вебзастосунку зоомагазину, який є менеджером. Реалізовано функцією `pet_store_food_comment_add_view`.

Представлення відтворення переліку очікуваних подальших замовлень вологого корму реалізовано функцією `wet_pet_store_orders_view`. Представлення відтворення проданих, запасів та очікуваних обсягів вологих кормів реалізовано функцією `wet_pet_store_food_orders_view`. Формуються переліки з кількості проданих позицій за кожним фасуванням вологого корму, наявних на цю мить запасів та кількості замовлень, які були класифіковані як такі, за якими очікується замовлення вологого корму та за якими не було здійснено наступне замовлення.

Представлення відтворення власних звернень до служби підтримки споживачів зоомагазину реалізовано функцією `customer_support_own_tickets_view`. Представлення відтворення переліку звернень до служби підтримки споживачів зоомагазину реалізовано функцією `customer_support_tickets_view`, де параметр `categoryId` визначає номер

ідентифікації категорії звернення, а параметр `status` визначає стан опрацювання звернення. Представлення звернення до служби підтримки споживачів зоомагазину реалізовано функцією `customer_support_ticket_create_view` на основі класу, що забезпечує роботу з відповідною формою на основі моделі. Представлення опрацювання звернення до служби підтримки споживачів зоомагазину реалізовано функцією `customer_support_ticket_answer_view`, яка має параметр `ticketId` з номером ідентифікації звернення, при цьому авторизований користувач вебзастосунку зоомагазину повинен мати права менеджера (аналогічно для `customer_support_tickets_view`).

Представлення відтворення переліку активних підписок на харчування в зоомагазині реалізовано функцією `pet_food_subscription_view`, що має параметр `subscriptionDate`. Якщо цей параметр не задано, то надається перелік всіх тварин з активною діючою підпискою. Якщо значення задано, то виводиться перелік тварин, для яких є діюча підписка та минув 31 день з останнього замовлення, яке було виконано за підпискою.

Представлення замовлення кормів для харчування тварини реалізовано функцією `pet_store_order_make_view`, яка має параметр `orderComps`, який через кому перелічує пари з номера ідентифікації фасування корму та кількості позиції, розділених літерою. Дане представлення окрім збереження самого замовлення використовує навчену модель для класифікації замовлення стосовно подальшого замовлення вологого корму. Далі реалізується підключення до платіжної системи PayPal для реалізації оплати.

Модель `Petstoreuser` реалізована через наслідування класу `AbstractBaseUser`.

Поле `surnameandname` представляє прізвище та ім'я (або альтернативний підпис) для відображення користувача у вебзастосунку зоомагазину і є додатковим до базової структури класу `AbstractBaseUser`, маючи тип `CharField`.

Поле `registereddate` представляє дату, а також і час, коли було зареєстровано обліковий запис користувача вебзастосунку зоомагазину, маючи тип `DateTimeField`.

Поле `lastvisitdate` представляє дату, а також і час, коли було востаннє авторизовано обліковий запис користувача вебзастосунку зоомагазину, маючи тип `DateTimeField`.

Поле `loginacc` представляє логін користувача вебзастосунку зоомагазину, маючи тип `CharField`.

Поле `passwordacc` представляє пароль користувача вебзастосунку зоомагазину, маючи тип `CharField`.

Поле `phoneforcontact` представляє номер телефону для контактів з користувачем вебзастосунку зоомагазину, маючи тип `CharField`.

Поле `emailforcontact` представляє електронну пошту для контактів з користувачем вебзастосунку зоомагазину, маючи тип `EmailField`.

Поле `storerepresenter` представляє тип користувача вебзастосунку зоомагазину: покупець або представник самого зоомагазину, маючи тип `BooleanField`.

Поле `promotype` представляє тип промоції, за якою було виконано реєстрацію облікового запису користувача зоомагазину (самостійно, за рекомендацією друга, за рекомендацією ветеринара тощо), маючи тип `PositiveSmallIntegerField`.

Поле `discountlevel` представляє відсоток дисконту, який діє для даного облікового запису користувача зоомагазину, маючи тип `DecimalField`.

Для роботи з даним класом було реалізовано клас `CustomUserManager`, що розширює клас `BaseUserManager`, реалізуючи зокрема метод `create_user` для підтримки реєстрації.

Модель типів тварин реалізовано класом `Pettytype` з полем `pettypename`, яке представляє назву типу тварини, маючи тип `CharField`.

Модель порід тварин, з якими потенційно працює зоомагазин, реалізовано класом `Petbreed` зі структурою з наступних полів.

Поле `pettype` представляє тип тварини, породою якої є дана, через зв'язок з моделлю `Pettype` за допомогою використання поля класу `ForeignKey`.

Поле `correctbreed` представляє назву породи тварини, маючи тип `CharField`.

Поле `breedszelevel` представляє категорію розміру тварини, до якої належить ця порода, маючи тип `PositiveSmallIntegerField`.

Моделі власних тварин користувачів вебзастосунку зоомагазину реалізовано класом `Ownpet` зі структурою з наступних полів.

Поле `petname` представляє ім'я цієї тварини користувача вебзастосунку зоомагазину, маючи тип `CharField`.

Поле `petbreed` представляє породу цієї тварини користувача вебзастосунку зоомагазину через зв'язок з моделлю `Petbreed` за допомогою використання поля класу `ForeignKey`.

Поле `petuserowner` представляє власника тварини, визначеного як користувача вебзастосунку зоомагазину через зв'язок з моделлю `Petstoreuser` за допомогою використання поля класу `ForeignKey`.

Поле `gender` представляє стать цієї тварини користувача вебзастосунку зоомагазину, маючи тип `BooleanField`.

Поле `petbirth` представляє приблизну або точну дату народження цієї тварини користувача вебзастосунку зоомагазину, маючи тип `DateTimeField`.

Поле `neutered` представляє позначку про те, чи є ця тварина користувача вебзастосунку зоомагазину кастрованою, чи ні, маючи тип `BooleanField`.

Поле `wetfoodate` представляє позначку про те, чи споживала ця тварина користувача вебзастосунку зоомагазину вологий корм, маючи тип `BooleanField`.

Поле `activesubscription` представляє наявність активно діючої підписки на доставку харчування з зоомагазину для цієї тварини користувача, маючи тип `BooleanField`.

Поле `registerdate` представляє дату, а також і час, коли цей користувач вебзастосунку зоомагазину зареєстрував у програмі свою тварину, маючи тип `DateTimeField`.

Поле `allergens` представляє алергени для конкретної тварини користувача вебзастосунку зоомагазину через зв'язок з моделлю `Foodcomponent` за допомогою використання поля класу `ManyToManyField`.

Поле `flavours` представляє смаки, які до вподоби конкретній тварині, через зв'язок з моделлю `Flavour` за допомогою використання поля класу `ManyToManyField`.

Поле `storefoods` представляє корми для тварин у зоомагазині, які до вподоби конкретній тварині, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ManyToManyField`.

Поле `healthissues` представляє проблеми зі здоров'ям, які були виявлені у цієї тварини, через зв'язок з моделлю `Healthissue` допомогою використання поля класу `ManyToManyField`.

Модель виробників кормів для тварин, з якими працює зоомагазин, реалізовано класом `Petstorefoodbrand` з полем `foodbrandname`, що представляє назву виробника кормів для тварин, з яким працює зоомагазин, маючи тип `CharField`.

Модель типів вологого корму, що є в зоомагазині на продажу, реалізовано класом `Wetfoodtype` з полем `wetfoodname`, що представляє назву типу вологого корму, що є в зоомагазині на продажу, маючи тип `CharField`.

Модель кормів для тварин у зоомагазині реалізовано класом `Petstorefood` зі структурою з наступних полів.

Поле `foodtitle` представляє скорочену назву корму для тварин у зоомагазині, маючи тип `CharField`.

Поле `foodcaption` представляє повну назву, якою підписується цей корм для тварин у зоомагазині, маючи тип `CharField`.

Поле `pettypefood` представляє тип тварин, для яких призначений цей корм, через зв'язок з моделлю `Pettype` за допомогою використання поля класу `ForeignKey`.

Поле `foodtype` представляє загальний тип корму для тварин, до якого належить цей корм, маючи тип `BooleanField`.

Поле `wetfoodtype` представляє тип вологого корму, до якого належить цей корм, через зв'язок з моделлю `Wetfoodtype` за допомогою використання поля класу `ForeignKey`.

Поле `storefoodbrand` представляє виробника цього корму для тварин через зв'язок з моделлю `Petstorefoodbrand` за допомогою використання поля класу `ForeignKey`.

Поле `breedsizellevel` представляє розмір породи, для якого може використовуватися цей корм для тварин, маючи тип `SmallIntegerField`.

Поле `countryproduced` представляє країну, в якій виготовлено цей корм для тварин, маючи тип `CharField`.

Поле `proteincomponent` представляє основне джерело білка через зв'язок з моделлю `Foodcomponent` за допомогою використання поля класу `ForeignKey`.

Поле `mainfavour` представляє основний смак корму через зв'язок з моделлю `Flavour` за допомогою використання поля класу `ForeignKey`.

Поле `onpurpose` представляє основне призначення цього корму для тварин, маючи тип `SmallIntegerField`.

Поле `classtier` представляє клас, до якого належить цей корм для тварин, маючи тип `SmallIntegerField`.

Поле `calories` представляє кількість кілокалорій, яку можна отримати при споживанні 100 г даного корму, маючи тип `DecimalField`.

Поле `detaildescription` представляє детальний опис корму для тварин, маючи тип `CharField`.

Модель проблем зі здоров'ям тварин, за яких рекомендовано або навпаки не рекомендовано використання корму, реалізовано класом `Petstorefoodhealthissue` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин, що є в цьому зоомагазині та для якого співставлено цю проблему зі здоров'ям тварин, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `healthissue` представляє проблему зі здоров'ям, за якої рекомендовано або не рекомендовано використання даного корму, через зв'язок з моделлю `Healthissue` за допомогою використання поля класу `ForeignKey`.

Поле `positive` представляє позначку, яка визначає за правдивого значення, що цей корм рекомендовано використовувати при цій проблемі зі здоров'ям у тварини, а за хибного значення відповідно, що не рекомендовано використовувати цей корм взагалі, маючи тип `BooleanField`.

Модель рівнів зрілості тварин, за яких можливе споживання конкретного корму з зоомагазину, реалізовано класом `Petstorefoodmature` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин з зоомагазину, який можна споживати за даного рівня зрілості тварини, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `maturelevel` представляє рівень зрілості тварини, за якого можна споживати цей корм для тварин, маючи тип `SmallIntegerField`.

Модель можливих фасувань, за якими і продається корм для тварин у зоомагазині, реалізовано класом `Petstorefoodtray` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин, що продається в цьому зоомагазині та для якого визначено це фасування, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `trayweight` представляє вагу даного фасування корму для тварин, маючи тип `DecimalField`.

Поле `baseprice` представляє стандартну вартість даного корму для тварин у цьому фасуванні, маючи тип `DecimalField`.

Поле `currentprice` представляє вартість даного корму для тварин у цьому фасуванні, за якою його можна купити в поточний момент, маючи тип `DecimalField`.

Поле `availablepositions` представляє кількість цього корму в цьому фасуванні, яка доступна для продажу, маючи тип `IntegerField`.

Поле `topdiscount` представляє максимальний дисконт (у відсотках), який може бути застосований при продажу цього корму в цьому фасуванні, маючи тип `DecimalField`.

Модель фотографії корму для тварин реалізовано класом `Petstorefoodphoto` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин, який представлений на фотографії, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `photo` представляє саму фотографію корму для тварин, маючи тип `ImageField`.

Поле `isposter` представляє позначку про те, чи ця фотографія корму для тварин використовується як постер для цього корму, тобто як базова фотографія зокрема в результатах відображення різних кормів, чи це додаткова фотографія, маючи тип `BooleanField`.

Модель коментарів користувачів вебзастосунку зоомагазину про корми для тварин реалізовано класом `Petstorefoodcomment` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин, до якого відноситься коментар, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `petstoreuser` представляє користувача вебзастосунку зоомагазину, який зберіг цей коментар, через зв'язок з моделлю `Petstoreuser` за допомогою використання поля класу `ForeignKey`.

Поле `commentcontent` представляє текст коментаря, який користувач вебзастосунку зоомагазину залишив про цей корм, маючи тип `CharField`.

Поле `reactioncomment` представляє текст відповіді на коментар, який користувач вебзастосунку зоомагазину залишив про цей корм, створений представником зоомагазину, маючи тип `CharField`.

Поле `maincommenttime` представляє дату, а також і час, коли було створено коментар користувачем вебзастосунку зоомагазину про корм для тварин, маючи тип `DateTimeField`.

Поле `maincommenttime` представляє дату, а також і час, коли було створено відповідь на коментар користувачем вебзастосунку зоомагазину про корм для тварин представником зоомагазину, маючи тип `DateTimeField`.

Модель смаків, доступних до вибору під час роботи з кормами зоомагазину, реалізовано класом `Flavour` з полем `flavourname`, що представляє назву смаку, маючи тип `CharField`.

Модель складу корму для тварин реалізовано класом `Petstorefoodcomponent` зі структурою з наступних полів.

Поле `storefood` представляє корм для тварин, вміст речовин до якого визначено, через зв'язок з моделлю `Petstorefood` за допомогою використання поля класу `ForeignKey`.

Поле `foodcomponent` представляє речовину, яка входить до цього корму для тварин, через зв'язок з моделлю `Foodcomponent` за допомогою використання поля класу `ForeignKey`.

Поле `percentfromall` представляє відсоток вмісту даної речовини у складі цього корму для тварин, маючи тип `DecimalField`.

Поле `descriptioncomment` представляє опис особливостей вмісту цієї речовини до складу цього корму для тварин, якщо такі особливості є, маючи тип `CharField`.

Модель речовин, які можуть входити до складу кормів для тварин, реалізовано класом `Foodcomponent` з полем `componentname`, що представляє

назву речовини, яка може входити до складу кормів для тварин, які продаються в цьому зоомагазині, маючи тип CharField.

Модель замовлень кормів з зоомагазину реалізовано класом Petstoreorder зі структурою з наступних полів.

Поле ownpet представляє тварину, для якої її власник як користувач вебзастосунку зоомагазину замовив корми, через зв'язок з моделлю Ownpet за допомогою використання поля класу ForeignKey.

Поле suscriptionorder представляє позначку про те, що це замовлення є або не є реалізацією підписки на забезпечення харчування тварини зоомагазином, маючи тип BooleanField.

Поле typicalsubscription представляє позначку про те, що це замовлення має в подальшому використовуватися як основа (шаблон) для реалізації активної підписки, маючи тип BooleanField.

Поле timeordered представляє дату, а також і час, коли було оформлено це замовлення кормів з зоомагазину для власної тварини, маючи тип DateTimeField.

Поле timefordelivery представляє дату, а також і час, коли має бути доставлено (виконано) це замовлення кормів з зоомагазину для тварини, маючи тип DateTimeField.

Поле delivaddr представляє адресу доставки замовленого корму для тварин, маючи тип CharField.

Поле repeatwaited представляє результат класифікації замовлення щодо подальшого замовлення вологого корму (число інтерпретується як набір 1 та 0, що відповідають результату класифікації за кожним типом вологого корму), маючи тип PositiveIntegerField.

Поле pricerayed представляє вартість, яку було сплачено за замовлення кормів з зоомагазину для тварини, маючи тип DecimalField.

Поле transactionnmbr представляє номер транзакції, через яку було виконано оплату замовлення, маючи тип CharField.

Поле `discountused` представляє дисконт у відсотках, який було використано для сплати за це замовлення, маючи тип `DecimalField`.

Поле `statusorder` представляє стан замовлення кормів з зоомагазину, маючи тип `PositiveSmallIntegerField`.

Модель складу замовлень кормів з зоомагазину реалізовано класом `Petstoreordercomponent` зі структурою з наступних полів.

Поле `petstoreorder` представляє замовлення кормів з зоомагазину, до якого відноситься ця складова, через зв'язок з моделлю `Petstoreorder` за допомогою використання поля класу `ForeignKey`.

Поле `storefoodtray` представляє фасування корму для тварин, яке є складовою цього замовлення кормів з зоомагазину, через зв'язок з моделлю `Petstorefoodtray` за допомогою використання поля класу `ForeignKey`.

Поле `trayquantity` представляє кількість одиниць цього корму в даному фасуванні, яку було замовлено, маючи тип `PositiveIntegerField`.

Поле `discountused` представляє використаний дисконт для визначення вартості одиниці цього фасування корму у відсотках, маючи тип `DecimalField`.

Поле `componentprice` представляє фактичну вартість одиниці цього фасування корму в замовленні (після застосування дисконту), маючи тип `DecimalField`.

Модель можливих проблем зі здоров'ям реалізовано класом `Healthissue` з полем `healthissuetitle`, що представляє назву можливої проблеми зі здоров'ям тварини, маючи тип `CharField`.

Модель категорій, за якими можуть відбуватися звернення до служби підтримки споживачів у зоомагазині, реалізовано класом `Customersupportticketcategory` з полем `csticketcategoryname`, що представляє назву категорії звернення до служби підтримки споживачів у зоомагазині, маючи тип `CharField`.

Модель звернень до служби підтримки споживачів у зоомагазині реалізовано класом `Customersupportticket` зі структурою з наступних полів.

Поле `csticketcategory` представляє категорію, до якої належить звернення до служби підтримки споживачів у зоомагазині, через зв'язок з моделлю `Customersupportticketcategory` за допомогою використання поля класу `ForeignKey`.

Поле `petstoreuser` представляє користувача вебзастосунку зоомагазину, який звернувся до служби підтримки, через зв'язок з моделлю `Petstoreuser` за допомогою використання поля класу `ForeignKey`.

Поле `ticketcontent` представляє суть звернення до служби підтримки споживачів у зоомагазині, маючи тип `CharField`.

Поле `ticketanswercontent` представляє суть відповіді на звернення до служби підтримки споживачів у зоомагазині, маючи тип `CharField`.

Поле `answerpetstoreuser` представляє користувача вебзастосунку зоомагазину, який є представником магазину та який надав відповідь на звернення до служби підтримки, через зв'язок з моделлю `Petstoreuser` за допомогою використання поля класу `ForeignKey`.

Поле `tickettimecreated` представляє дату, а також і час, коли було подано звернення до служби підтримки зоомагазину, маючи тип `DateTimeField`.

Поле `ticketstatus` представляє стан обробки звернення до служби підтримки зоомагазину, маючи тип `PositiveSmallIntegerField`.

Поле `answertimecreated` представляє дату, а також і час, коли було надано відповідь на звернення до служби підтримки зоомагазину, маючи тип `DateTimeField`.

Модель відвідувань користувачем вебзастосунку зоомагазину реалізовано класом `Petstoreuservisit` зі структурою з наступних полів.

Поле `petstoreuser` представляє користувача вебзастосунку зоомагазину, який відвідав вебзастосунок зоомагазину, через зв'язок з моделлю `Petstoreuser` за допомогою використання поля класу `ForeignKey`.

Поле `timevisit` представляє дату, а також і час, коли таке відвідування було зафіксовано, маючи тип `DateTimeField`.

Поле `urlopened` представляє посилання, яке було відкрито користувачем вебзастосунку зоомагазину, маючи тип `URLField`.

Поле `newvisit` представляє відвідування, яке відкриває нову сесію під час роботи з вебзастосунком або яке продовжує роботу через відкриття наступного посилання за тієї ж самої авторизації, маючи тип `BooleanField`.

4.3 Висновки за розділом 4

За результатами успішного виконання даного розділу розроблено ПЗ, що програмно надає всі функції, зазначені у вимогах, прийнято рішення щодо зміни станів замовлень корму для тварин, що визначило порядок звернень до функцій програми, описано створені для реалізації функції скрипти, програмні функції, класи.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Опис програми

ПЗ розроблено для реалізації підтримки роботи зоомагазину, що включає пошук, перегляді інформації, яка стосується кормів для тварин, замовлення цих кормів та керування такими замовленнями, визначення наявної кількості кормів у зоомагазині на основі даних про продані корми, наявні запаси та очікувані замовлення.

5.2 Експлуатація програми

Експлуатація програми для користувача, хоч покупця зоомагазину, хоч менеджера зоомагазину, є інтуїтивно зрозумілою загалом, а першочерговий контроль реалізується через меню, доступне на кожній сторінці вгорі.

Меню містить стандартний пошук, який дозволяє ввести назву корму, а всі інші параметри можна задати зі сторінки пошуку кормів, де можна також встановити різні параметри, включаючи тип тварини, потрібний смак, обираючи відповідні значення, тип корму або детальний варіант вологого корму з пропонованих, розмір породи тварини, країни-виробники, мінімальну та максимальну кількість калорій, проблеми зі здоров'ям, з якими має допомагати корм або за яких корм протипоказаний (окрема галочка дозволяє обрати між цими двома варіантами), рівень зрілості тварини, потрібні речовини у складі корму, алергени тварини, порядок впорядкування результатів тощо, а також можливість обрати одну з власних тварин, використовуючи її як прообраз для пошуку (обираючи її тип, рівень зрілості, алергени тощо).

Будь-який результат з кормів можна відкрити та переглянути всю інформацію (рис. 5.1). Така інформація включає весь набір фотографій,

основні дані, варіанти фасування з кнопками для додавання такого фасування (позначено плюсом) до поточного замовлення, коментарів користувачів.



Рисунок 5.1 – Відтворення даних корму для тварин

Кнопка «У кошик» дозволяє перейти до частини сторінки з фасуваннями. Після додавання певного варіанта фасування біля плюса з'являється кількість доданих позицій та мінус для вилучення. Кнопка «Зберегти» відтворює ту частину сторінки, де можна обрати одну з своїх власних тварин, для якої цей корм зберегти. Якщо тварина одна, то збереження відбувається автоматично, а кнопка змінюється на «Збережено».

Пункт меню «Замовлення» спочатку відкриває сторінку зі всіма замовленнями користувача вебзастосунку зоомагазину (якщо він авторизувався), звідки можна за кнопками «За тваринами» або «Створити» перейти відповідно до відтворення замовлень корму за конкретною твариною користувача (рис. 5.2) або до вмісту кошика.

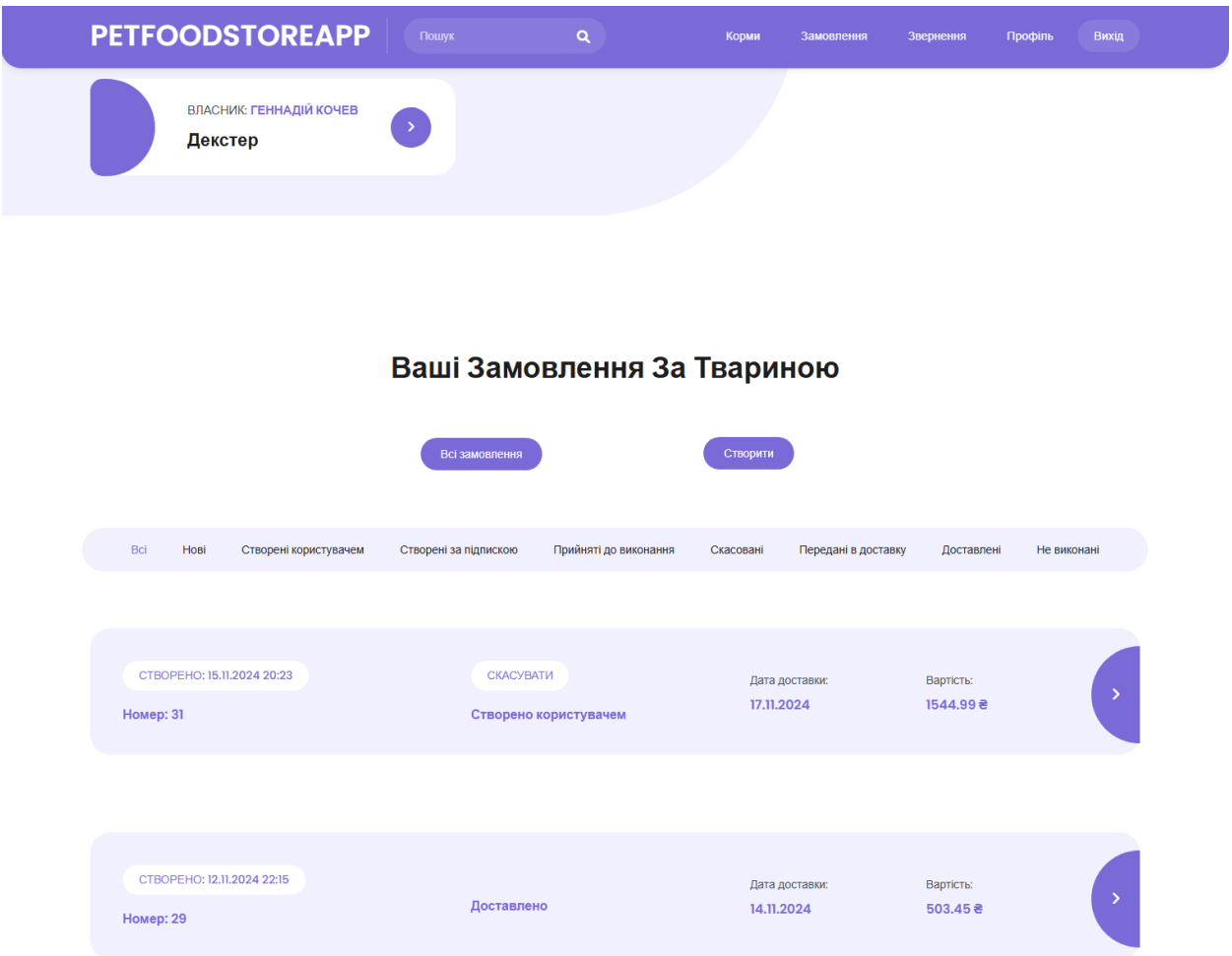


Рисунок 5.2 – Відтворення повного переліку замовлень кормів за твариною

На сторінці з замовленнями за твариною можна згори за допомогою стрілочки перемикатися між тваринами. Окрім того можна обрати всі замовлення або замовлення з потрібним станом. Нові замовлення включають різні стани, але такі, які ще не завершені (успішно або неуспішно). За власним замовленням можна перейти до його скасування, якщо воно було створене самим користувачем та не було оброблено далі менеджером. Стрілка з правої сторони рядка замовлення дозволяє перейти до відтворення повного вмісту замовлення.

Пункт меню «Корми» дозволяє встановити фільтри за кормом, але спочатку потрібно обрати тип тварини, тобто цей варіант роботи передбачає обов'язкове визначення типу тварини (собака, кіт тощо). З цієї ж сторінки доступне посилання для переходу на сторінку зі збереженими кормами, звідки можна відобразити перелік, перемикаючись між власними тваринами.

Пункт меню «Звернення» дозволяє переглядати власні подані звернення до служби підтримки, подавати нові. Пункт меню «Профіль» містить дані користувача, дозволяє визначити активність підписки. Визначення останнього замовлення, за яким ця підписка функціонує, здійснюється зі сторінки перегляду вмісту замовлення. Зі сторінки профілю доступний також перелік тварин, звідки можна переглянути або дані самої тварини, або перейти до замовлень кормів за твариною. Тут також можна перейти до редагування даних та додати за потреби нову тварину.

Для менеджера зоомагазину меню подібне, проте містить також пункт «Планування» замість профілю. Планування дозволяє переглянути корми, обираючи спочатку типи, а тоді відображаючи перелік позицій, за кожною з яких можна обрати фасування (також відредагувати або додати дані нового фасування), а також доступно дані про те, скільки кормів за цією позицією (загалом даного типу та окремо за найменуванням) продано, скільки замовлень за цим типом корму загалом (і які не були реалізовані ще) були класифіковані як такі, що передбачають наступне замовлення вологого корму певного класу, а також кількість позицій цього фасування, що було продано на цю мить.

Сторінка з замовленнями для менеджера зоомагазину дозволяє переходити до очікуваних подальших замовлень (рис. 5.3). Це перелік всіх замовлень корму в зоомагазині, які були програмно на основі запропонованого методу класифіковані як такі, що призведуть до замовлення вологого корму в подальшому. За кожним таким замовленням демонструється за яким саме типом вологого корму очікується замовлення. Якщо таких варіантів декілька, то вони перелічуються.

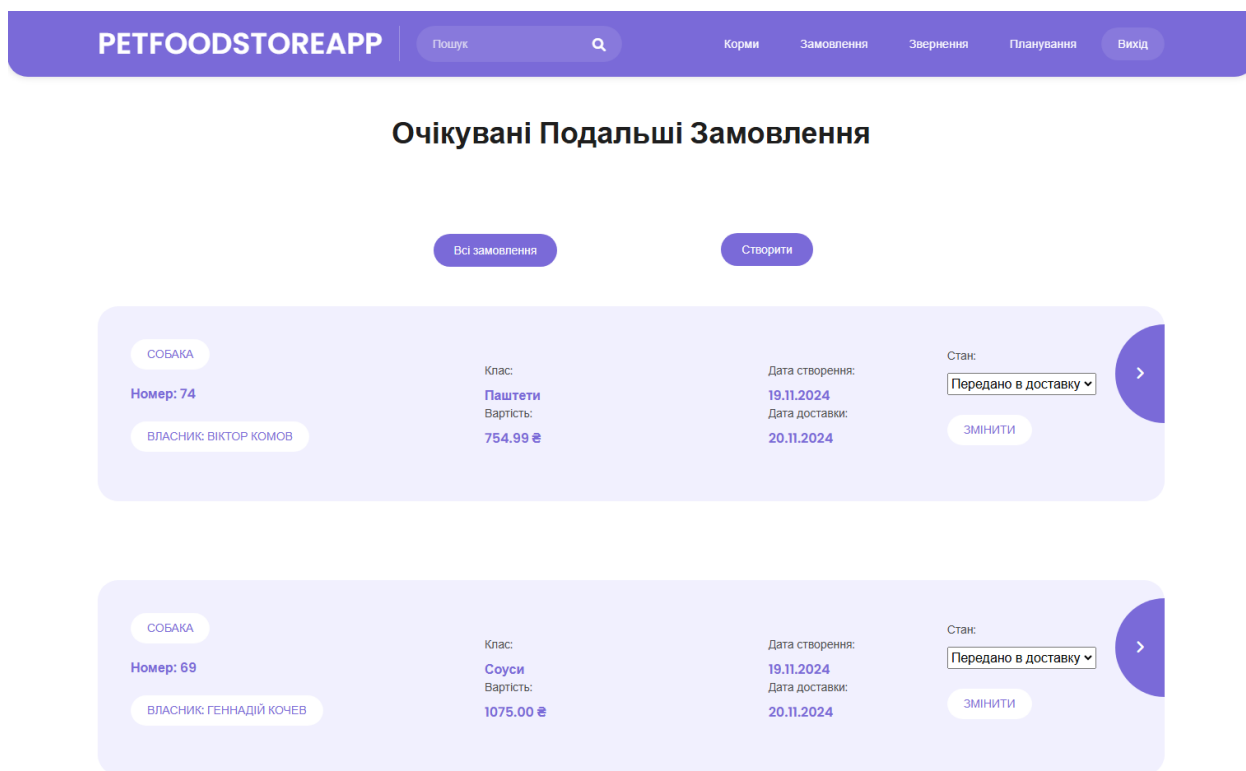


Рисунок 5.3 – Відтворення переліку очікуваних подальших замовлень вологого корму

З цієї сторінки як і зі сторінки всіх замовлень менеджер може обрати стан, на який потрібно змінити стан замовлення, та підтвердити таку зміну.

З пункту меню «Звернення» можна обробляти всі звернення, які було подано до служби підтримки зоомагазину, обираючи потрібні категорії при цьому, стани, керуючим таким чином даним процесом.

5.3 Експериментальне дослідження

Реалізовано експериментальне дослідження на підґрунті вибірки даних [8], яку було детально описано в розділі 2 і яка представлена відповідним CSV-файлом. Після перетворення даної вибірки на набір замовлень за принципом реалізації методу класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи всього було створено близько 45 тисяч записів. Кожен з цих записів відповідав виконаному замовленню зі встановленим цільовим класом для нього. Усі дані розділялись за принципом 60 % на навчання та 40 % на тестування. У випадку, якщо кількість записів одного цільового класу (замовлення, яке в подальшому призводило до замовлення вологого корму, що коротко названо «подальше замовлення») перевищувала кількість записів іншого цільового класу (замовлення, яке в подальшому не призводило до замовлення вологого корму, що коротко названо «подальше замовлення») більше ніж у 20 разів.

Для реалізації вибору ознак було використано оцінювання ознак на основі функції SelectKBest пакету scikit-learn, для чого використовувався критерій Пірсона. Такі результати було отримано для кожного типу вологих кормів, зокрема і для підливних желейних паштетів (табл. 5.1). Шість ознак з найвищими оцінками (унікаючи дублюючих) було виділено для створення моделі. У останньому випадку це були кілокалорії вологого корму, кілокалорії сухого корму, кількість позицій вологого корму в замовленні (кількість лотків для доставки), сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті, порядковий номер замовлення за даною твариною серед всіх її замовлень, в яких було визначено вологий корм додатково до сухого, кількість днів останнього звернення в службу підтримки (замість дати використовувалась кількість днів, що пройшла з часу першого зафіксованого у зоомагазині звернення).

Таблиця 5.1 – Вибір ознак на основі критерію Пірсона для ознак вибірки даних

Ознака	Оцінка
Кілокалорії вологого корму	24423404,66
Кілокалорії сухого корму	3510537,56
Сумарна кількість кілокалорій корму	603225,23
Кількість позицій вологого корму в замовленні	150521,80
Сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті	46041,46
Порядковий номер замовлення за даною твариною серед всіх її замовлень, в яких було визначено вологий корм додатково до сухого	42544,10
Сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті з моменту останнього замовлення	18793,36
Кількість замовлень, які були оформлені після першого замовлення вологого корму цим користувачем для своєї тварини	15680,24
Кількість днів останнього звернення в службу підтримки	13620,04

Для кожного типу вологого корму було створено окрему модель на основі запропонованого методу класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи з використанням методу Random Forest, а отримані результати тестування було оцінено на основі показників точності та влучності, який було розраховано для непродовженого замовлення та для подальшого замовлення (табл. 5.2). У моделі використовувалось 50 дерев.

Таблиця 5.2 – Результати класифікації на основі запропонованого методу класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

Тип вологого корму	Точність	Влучність для непродовженого замовлення	Влучність для подальшого замовлення
Соуси	0,9821478124132828	0,99096122	0,80842912
Соусні желе	0,9864690721649485	0,9899312	0,90361446
Підливні желейні паштети	0,9074778909660808	0,93447885	0,69059656
Підливні паштети	0,983672899804633	0,98880977	0,85964912
Желе	0,9858874060438717	0,99100546	0,87713311
Желейні паштети	0,9915137614678899	0,99281093	0,96256684
Паштети	0,9850850292723724	0,99041673	0,85714286

Окрім того було отримано відповідні результати класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи на основі використання моделі, побудованої за допомогою дерев рішень за алгоритмом Classification and Regression Tree (CART) (табл. 5.3).

Ще один набір результатів було отримано на основі моделі, побудованої за допомогою методу k-найближчих сусідів для виконання класифікації (табл. 5.4).

Отримані результати виявились кращими у запропонованого методу, при цьому точність була підвищена порівняно з CART та методом k-найближчих сусідів на 2,8 % та 3,7 % відповідно. Однак показник влучності дозволяє більш точно виявити зміни. Це пов'язано з тим, що більша частина спостережень належить до класу непродовженого замовлення, тому показник точності та показник влучності для непродовженого замовлення не

дозволяють продемонструвати ефективність класифікації при роботі з подальшими замовленнями.

Таблиця 5.3 – Результати класифікації на основі алгоритму CART

Тип вологого корму	Точність	Влучність для не продовженого замовлення	Влучність для подальшого замовлення
Соуси	0,9768754046804181	0,98738599	0,76237624
Соусні желе	0,9819587628865979	0,99004891	0,81003584
Підливні желейні паштети	0,8774207992835553	0,92608053	0,54727669
Підливні паштети	0,97488138431482	0,98784773	0,71130952
Желе	0,9825126553152324	0,99034127	0,82410423
Желейні паштети	0,9873853211009175	0,9925427	0,8817734
Паштети	0,9789517702815723	0,99107274	0,73607038

Таблиця 5.4 – Результати класифікації на основі методу k-найближчих сусідів

Тип вологого корму	Точність	Влучність для не продовженого замовлення	Влучність для подальшого замовлення
Соуси	0,9522708352603829	0,95984578	0,49717514
Соусні желе	0,9513530927835051	0,959157	0,40229885
Підливні желейні паштети	0,8681853800514945	0,90117561	0,51694915
Підливні паштети	0,9517164387384873	0,95799267	0,33333333
Желе	0,9519865009970855	0,96205112	0,5
Желейні паштети	0,9474770642201835	0,95967175	0,4
Паштети	0,9530248118204628	0,9619142	0,38738739

Відповідно показник влучності для подальшого замовлення продемонстрував для методу k-найближчих сусідів, що така модель фактично є непридатною на практиці, оскільки для певних типів вологого корму значення показника виявилось менше 0,5. Окрім того за усередненими значеннями серед всіх типів вологого корму (табл. 5.5) значення даного показника для методу k-середніх дорівнює лише 0,43. У той же час дерева рішень на основі алгоритму CART дозволили отримати придатні результати для практичного застосування разом зі значенням влучності для подальшого замовлення, що усереднено дорівнює 0,75 та для жодного типу вологого корму не мало значення менше 0,547. При цьому запропонований метод серед всіх типів вологого корму мав найменше значення даного показника у 0,69, а середнє – 0,85, що на 96,2 % краще за відповідне значення для методу k-середніх та на 13 % краще за відповідне значення дерев рішень CART.

Таблиця 5.5 – Середні результати класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

	Точність	Влучність для непродовженого замовлення	Влучність для подальшого замовлення
Запропонований метод	0,974607696	0,982630594	0,851304581
CART	0,9657123	0,980759981	0,753278043
Метод k-середніх	0,939430589	0,951686876	0,433877694

5.4 Висновки за розділом 5

За результатами успішного виконання даного розділу описано процес роботи користувача з ПЗ, виконано експериментальне дослідження запропонованого методу класифікації замовлень в зоомагазині стосовно

подальшого замовлення вологого корму для планування роботи порівняно з деревами рішень на основі алгоритму CART та методу k-середніх. Було виявлено, що доцільним для практичного використання може бути застосування алгоритму CART та запропонованого методу, при цьому запропонований метод дозволив отримати кращі значення точності та влучності, що загалом демонструє його переваги та дозволило впровадити класифікацію в програмі при створенні замовлення саме на основі даного методу.

ВИСНОВКИ

У роботі розроблено ПЗ підтримки роботи зоомагазину, що цілісно дозволило підвищити інформованість прийняття рішень стосовно планування роботи зоомагазину, таким чином досягнувши мету роботи.

У роботі реалізовано вебзастосунок зоомагазину, який призначений для підтримки роботи зоомагазину шляхом пошуку, перегляду інформації, яка стосується кормів для тварин, замовлення цих кормів та керування такими замовленнями, визначення наявної кількості кормів у зоомагазині на основі даних про продані корми, наявні запаси та очікувані замовлення, що реалізовано за допомогою виконання класифікації замовлень на основі навченої моделі з використанням запропонованого методу. Для реалізації використано мову Python разом з набором пакетів для роботи з даними та машинного навчання, фреймворк веброзробки Django разом з системою керування базами даних PostgreSQL.

Наукова новизна роботи полягає в методі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи, який виконує класифікацію замовлень на основі моделі, побудованої методом Random Forest, визначає виконання відбору ознак для побудови даної моделі для конкретного типу вологого корму, підготовку вибірки даних для відповідного навчання моделі. У роботі виконано експериментальне дослідження запропонованого методу класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи порівняно з деревами рішень на основі алгоритму CART та методу k-середніх. Було виявлено, що доцільним для практичного використання може бути застосування алгоритму CART та запропонованого методу, при цьому запропонований метод дозволив отримати кращі значення точності та влучності, що загалом демонструє його переваги та дозволило впровадити класифікацію в програмі при створенні замовлення саме на основі даного методу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Лук'янчук, Д. Як війна вплинула на життя тварин та зоозахист в Україні [Електрон. ресурс] / Д. Лук'янчук. – Режим доступу : <https://life.pravda.com.ua/columns/2023/02/24/253030/>.
2. Тиравський, В. В Україні, внаслідок війни, назавжди зникли 20 видів рослин і тварин [Електрон. ресурс] / В. Тиравський. – Режим доступу : <https://foreignukraines.com/2023/11/22/in-ukraine-as-a-result-of-the-war-20-species-of-plants-and-animals-disappeared-forever/>.
3. Фільм "Ми, наші улюбленці та війна" ("Us, Our Pets and the War") [Електрон. ресурс]. – Режим доступу : <https://megogo.net/ua/view/23777956-mi-nashi-ulyublentsi-ta-viyna.html>.
4. Шутяк, Л. Врятувати своїх [Електрон. ресурс] : як українці вивозять тварин із «гарячих точок» / Л. Шутяк. – Режим доступу : <https://media.zagoriy.foundation/velyka-istoriya/vryatuvaty-svoyih-yak-ukrayinczi-vyvozyat-tvaryn-iz-garyachyh-tochok/>.
5. Вони нас люблять, а ми їх? [Електрон. ресурс] : українці вийшли на марш за права тварин. – Режим доступу : <https://rubryka.com/photo/marsh-zaprava-tvaryn/>.
6. Зоомагазин MasterZoo – товари для домашніх тварин в Україні [Електрон. ресурс]. – Режим доступу : <https://masterzoo.ua/ua/>.
7. Зоомагазин Pethouse.ua Київ – купити зоотовари в Україні [Електрон. ресурс]. – Режим доступу : <https://pethouse.ua/ua/>.
8. Raina, J. Pet Food Customer Orders Online [Electronic resource]. – Access mode : <https://www.kaggle.com/datasets/jahangirraina/pet-food-customer-orders-online>.
9. What Is Random Forest? IBM [Electronic resource]. – Access mode : <https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems>.
10. Django Documentation [Electronic resource]. – Access mode :

<https://docs.djangoproject.com/en/5.1/>.

11. Welcome to Flask – Flask Documentation (3.0.x) [Electronic resource]. – Access mode : <https://flask.palletsprojects.com/en/stable/#user-s-guide>.

12. Free Template 586 Scholar [Electronic resource]. – Access mode : <https://templatemo.com/tm-586-scholar>.

ДОДАТОК А
Технічне завдання

Вступ

Організація та планування роботи зоомагазину потребує врахування багатьох факторів для розуміння власної клієнтської бази:

- з однієї сторони ця клієнтська база визначається тими замовленнями, які вже було виконано;
- з другого боку вона пов'язана з тими замовленнями, які принесуть повторні замовлення в майбутньому;
- з третьої сторони цього замало для планування роботи зоомагазину, адже потрібно розуміти, які корми клієнт в подальшому замовить: розуміння хоча би групи кормів дозволяє планувати очікувану кількість замовлень за групою кормів, а відповідно і замовляти такі корми для продажу в зоомагазині (якщо кількість кормів буде недостатньою, клієнт може не повернутися в цей зоомагазин, а обрати конкурента, проте якщо кількість кормів є надмірною, то це додаткові витрати для зоомагазину).

До того ж всі ці сторони мають підтримуватися вебзастосунком зоомагазину, таким чином підкреслюючи, що забезпечення продажу кормів на даний момент вже недостатньо, зважаючи на конкурентність такого бізнесу. Врахування особливостей тварини, для якої користувач вебзастосунку як її господар збирається придбати корм, є важливим і сприяє зручності його роботи, а значить і задоволеності взаємодії з зоомагазином загалом. Тож цей набір факторів має бути врахований під час розробки вебзастосунку в цій роботі, щоб сприяти задоволеності клієнтів зоомагазину з одного боку та інформованості для реалізації планування роботи зоомагазину з іншого боку.

A.1 Підстави для розробки

На підставі наказу № 398 від 7 жовтня 2024 р., який був виданий у Національному університеті «Запорізька політехніка», де і ведеться розробка в межах даної дипломної кваліфікаційної роботи магістра, визначена тема

роботи «Дослідження та програмна реалізація методів планування роботи зоомагазину», що було деталізовано також виданим завданням на роботу.

A.2 Призначення розробки

ПЗ має бути розроблено для реалізації підтримки роботи зоомагазину, що включає пошук, перегляді інформації, яка стосується кормів для тварин, замовлення цих кормів та керування такими замовленнями, визначення наявної кількості кормів у зоомагазині на основі даних про продані корми, наявні запаси та очікувані замовлення.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для реалізації комплексної роботи вебзастосунку зоомагазину як щодо продажу і замовлення кормів, так і щодо планування роботи зоомагазину необхідно забезпечити такі функціональні можливості:

- виконання авторизації та реєстрації покупців та менеджерів зоомагазину;
- відтворення профілю користувача разом з його власними тваринами;
- внесення даних власної тварини користувачем вебзастосунку зоомагазину, включаючи визначення алергенів, улюблених смаків тварини та наявних проблем зі здоров'ям;
- змінювання даних власної тварини користувачем вебзастосунку зоомагазину;
- відтворення даних тварини користувача вебзастосунку зоомагазину;
- відтворення повного переліку замовлень кормів за твариною;
- відтворення повного переліку замовлень кормів за тваринами користувача вебзастосунку зоомагазину;

– пошук кормів у зоомагазині, забезпечуючи також індивідуалізований пошук кормів для придбання для власної тварини користувача вебзастосунку зоомагазину;

– відтворення наявних кормів у зоомагазині за обраним типом тварин та у відповідності з обраною власною твариною користувача вебзастосунку зоомагазину;

– збереження корму для подальших замовлень за твариною та скасування;

– відтворення вмісту замовлення;

– відтворення переліку кормів, збережених за твариною;

– відтворення даних корму для тварин;

– відтворення наявності фасувань корму для тварин;

– змінювання наявності фасування корму для тварин;

– додавання фасування корму для тварини;

– змінювання даних фасування корму для тварини;

– керування замовленнями;

– додавання корму для тварин;

– змінювання даних корму для тварин;

– коментування корму для тварин;

– оформлення підписки на харчування в зоомагазині;

– скасування підписки на харчування в зоомагазині;

– створення відповіді на коментар покупця менеджером зоомагазину;

– класифікація замовлення стосовно подальшого замовлення вологого корму;

– відтворення переліку очікуваних подальших замовлень вологого корму;

– замовлення кормів для харчування тварини;

– відтворення проданих, запасів та очікуваних обсягів вологих кормів;

– звернення до служби підтримки споживачів зоомагазину;

- відтворення переліку звернень до служби підтримки споживачів зоомагазину;
- опрацювання звернення до служби підтримки споживачів зоомагазину;
- відтворення власних звернень до служби підтримки споживачів зоомагазину;
- відтворення переліку активних підписок на харчування в зоомагазині, в тому числі на заданий день.

A.3.2 Умови експлуатації

Експлуатація ПЗ підтримки роботи зоомагазину має забезпечуватися виконанням ряду вимог до серверної частини:

- забезпечення вебсерверу з підтримкою інтерпретатора мови програмування Python 3;
- налаштування Django-застосунку для виконання на сервері з реалізацією міграції для забезпечення необхідної структури БД за допомогою підтримки системи керування БД PostgreSQL;
- інсталювані Python-пакети Numpy, Pandas, Scikit-learn.

A.3.3 Вимоги до складу та параметрів технічних засобів

ПЗ підтримки роботи зоомагазину має забезпечуватися серверною частиною з такими мінімальними параметрами технічних засобів:

- двоядерний процесор;
- 4 Гб оперативної пам'яті;
- 10 Гб простору на жорсткому диску, що необхідно для збереження даних самого ВС та фотографій кормів, файлів вебзастосунку, БД: в залежності від якості, кількості фотографій можливе подальше збільшення мінімальних вимог.

А.4 Порядок контролю та приймання

Робота має виконуватися у відповідності з затвердженим завданням і контролюється керівником. Отримані результати роботи представляються в пояснювальній записці.

Завершальним етапом виконання та відповідно контролю і приймання дипломної кваліфікаційної роботи магістра є її захист, який виконується перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from django.shortcuts import render
from .models import Ownpet
from .models import Petstoreorder
from django.shortcuts import get_object_or_404
from django.core.paginator import Paginator
from .models import Petstoreuser
from .models import Pettype
from .models import Petbreed
from .models import Petstoreuservisit
from .models import Petstorefoodbrand
from .models import Wetfoodtype
from .models import Petstorefood
from .models import Petstorefoodhealthissue
from .models import Petstorefoodmature
from .models import Petstorefoodtray
from .models import Petstorefoodphoto
from .models import Petstorefoodcomment
from .models import Petstorefoodcomponent
from .models import Foodcomponent
from datetime import datetime
from .models import Petstoreordercomponent
from .models import Healthissue
from .models import Customersupportticketcategory
from .models import Customersupportticket
from django.db.models import Q

def ownpet_view(request, ownpet_id):
    if request.user.is_authenticated:
        pet_store_user = request.user
        requested_own_pets_all = Ownpet
            .objects
            .prefetch_related("allergens", "flavours", "storefoods", "healthissues")
            .select_related("petbreed")
            .filter(petuserowner =
pet_store_user)
        requested_own_pets =
requested_own_pets_all.filter(id = ownpet_id)
        if requested_own_pets:
            requested_own_pet = requested_own_pets.first()
            pet_store_orders = Petstoreorder
                .objects
                .filter(ownpet
= requested_own_pet)
                .order_by("-
timeordered")[:5]
            processed_pet_store_orders = Petstoreorder
                .objects
                .filter(ownpet = requested_own_pet, statusorder__in = [0, 1, 2,
3])

```

```

        .count()

        pet_store_user_visit =
Petstoreuservisit(petstoreuser = pet_store_user,
                    timevisit = datetime.now(),
                    urlopened =
request.build_absolute_uri(),
                    newvisit = False)
        pet_store_user_visit.save()
        return render(request,
"petfoodstoreapp/ownpet.html", {"ownpet": requested_own_pet,
"ownpets": requested_own_pets_all, "orders": pet_store_orders,
"neworders": processed_pet_store_orders, "message": ""})
        else:
            return render(request,
"petfoodstoreapp/ownpet.html", {"ownpet": None, "orders": None,
"neworders": 0, "ownpets": None, "message": "Це не ваш домашній
улюбленець. Дані задано некоректно. Така ситуація могла статися,
якщо ви змінювали посилання самостійно. Будь ласка,
використовуйте тільки програмно сформовані посилання зі сторінки
своїх улюбленців. У випадку, якщо перехід відбувся за
автоматичним посиланням, ми з радістю вирішимо цю ситуацію та
просимо вибачення за неї, але повідомте тоді нам про це,
скориставшись запитом до служби підтримки."})
        else:
            return render(request,
"petfoodstoreapp/ownpet.html", {"ownpet": None, "orders": None,
"neworders": 0, "ownpets": None, "message": "Без авторизації не
можна переглядати цю інформацію, оскільки вона доступна тільки
власникам."})

def ownpet_orders_view(request):
    ownpet_id = request.GET.get("ownpet")
    if not ownpet_id:
        return render(request,
"petfoodstoreapp/ownpetorders.html", {"ownpet": None, "orders":
None, "orders_all": 0, "orders_number": 0, "proc_orders_number":
0, "ownpets": None, "message": "Не задано дані улюбленця.
Неможливо відтворити таку сторінку. Така ситуація могла статися,
якщо ви змінювали посилання самостійно. Будь ласка,
використовуйте тільки програмно сформовані посилання зі сторінки
своїх улюбленців. У випадку, якщо перехід відбувся за
автоматичним посиланням, ми з радістю вирішимо цю ситуацію та
просимо вибачення за неї, але повідомте тоді нам про це,
скориставшись запитом до служби підтримки."})
    else:
        ownpet_id = int(ownpet_id)
        if request.user.is_authenticated:
            pet_store_orders_statuses =
request.GET.get("status")
            if pet_store_orders_statuses:
                if pet_store_orders_statuses == "user":
                    pet_store_orders_statuses = [0]

```

```

        elif pet_store_orders_statuses == "subscr":
            pet_store_orders_statuses = [1]
        elif pet_store_orders_statuses == "proc":
            pet_store_orders_statuses = [2]
        elif pet_store_orders_statuses == "deliv":
            pet_store_orders_statuses = [3]
        elif pet_store_orders_statuses == "done":
            pet_store_orders_statuses = [4]
        elif pet_store_orders_statuses == "lost":
            pet_store_orders_statuses = [5]
        elif pet_store_orders_statuses == "cancel":
            pet_store_orders_statuses = [6]
        elif pet_store_orders_statuses == "new":
            pet_store_orders_statuses = [0, 1, 2, 3]
    pet_store_user = request.user
    requested_own_pets_all = Ownpet
                                .objects
                                .filter(petuserowner =
pet_store_user)
    requested_own_pets = requested_own_pets_all

    .filter(id = ownpet_id)
        if requested_own_pets:
            requested_own_pet = requested_own_pets.first()
            pet_store_orders = Petstoreorder
                                .objects
                                .filter(ownpet
= requested_own_pet)
                if pet_store_orders_statuses:
                    pet_store_all_orders_count =
pet_store_orders.count()
                    pet_store_orders =
pet_store_orders.filter(statusorder__in =
pet_store_orders_statuses)
                else:
                    pet_store_all_orders_count = 0
                    pet_store_orders = pet_store_orders
                                .order_by("-
timeordered")
                    pet_store_orders_count = pet_store_orders

.count()
                    pet_store_orders_paginator =
Paginator(pet_store_orders, 10)
                    pet_store_orders_page = request.GET.get("page")
                    pet_store_orders_page =
pet_store_orders_paginator.get_page(pet_store_orders_page)
                    processed_pet_store_orders = Petstoreorder

.objects

.filter(ownpet = requested_own_pet, statusorder__in = [0, 1, 2,
3])

```

```

        .count()

        pet_store_user_visit =
Petstoreuservisit(petstoreuser = pet_store_user,
                    timevisit = datetime.now(),
                    urlopened =
request.build_absolute_uri(),
                    newvisit = False)
        pet_store_user_visit.save()
        return render(request,
"petfoodstoreapp/ownpetorders.html", {"ownpet":
requested_own_pet, "ownpets": requested_own_pets_all, "orders":
pet_store_orders_page, "orders_all": pet_store_all_orders_count,
"orders_number": pet_store_orders_count, "proc_orders_number":
processed_pet_store_orders, "message": ""})
    else:
        return render(request,
"petfoodstoreapp/ownpetorders.html", {"ownpet": None, "orders":
None, "orders_number": 0, "ownpets": None, "orders_all": 0,
"proc_orders_number": 0, "message": "Це не ваш домашній
улюбленець. Дані задано некоректно. Така ситуація могла статися,
якщо ви змінювали посилання самостійно. Будь ласка,
використовуйте тільки програмно сформовані посилання зі сторінки
своїх улюбленців. У випадку, якщо перехід відбувся за
автоматичним посиланням, ми з радістю вирішимо цю ситуацію та
просимо вибачення за неї, але повідомте тоді нам про це,
скориставшись запитом до служби підтримки."})
    else:
        return render(request,
"petfoodstoreapp/ownpetorders.html", {"ownpet": None, "orders":
None, "orders_number": 0, "ownpets": None, "orders_all": 0,
"proc_orders_number": 0, "message": "Без авторизації не можна
переглядати цю інформацію, оскільки вона доступна тільки
власникам."})

def petstore_orders_view(request):
    if request.user.is_authenticated:
        pet_store_user = request.user
        requested_own_pets_all = Ownpet
                                .objects
                                .filter(petuserowner =
pet_store_user)
        requested_own_pets =
requested_own_pets_all.filter(id = ownpet_id)
        if requested_own_pets:
            pet_store_orders = Petstoreorder
                                .objects
                                .filter(ownpet
__in = requested_own_pets)
            pet_store_orders_statuses =
request.GET.get("status")
            if pet_store_orders_statuses:
                if pet_store_orders_statuses == "user":
                    pet_store_orders_statuses = [0]

```

```

        elif pet_store_orders_statuses == "subscr":
            pet_store_orders_statuses = [1]
        elif pet_store_orders_statuses == "proc":
            pet_store_orders_statuses = [2]
        elif pet_store_orders_statuses == "deliv":
            pet_store_orders_statuses = [3]
        elif pet_store_orders_statuses == "done":
            pet_store_orders_statuses = [4]
        elif pet_store_orders_statuses == "lost":
            pet_store_orders_statuses = [5]
        elif pet_store_orders_statuses == "cancel":
            pet_store_orders_statuses = [6]
        elif pet_store_orders_statuses == "new":
            pet_store_orders_statuses = [0, 1, 2,
3]

        if pet_store_orders_statuses:
            pet_store_all_orders_count =
pet_store_orders.count()
            pet_store_orders = pet_store_orders

        .filter(statusorder__in = pet_store_orders_statuses)
        else:
            pet_store_all_orders_count = 0
            pet_store_orders = pet_store_orders
                                                                .order_by("-
timeordered")

            pet_store_orders_count = pet_store_orders

        .count()

            pet_store_orders_paginator =
Paginator(pet_store_orders, 10)
            pet_store_orders_page = request.GET.get("page")
            pet_store_orders_page =
pet_store_orders_paginator.get_page(pet_store_orders_page)
            processed_pet_store_orders = Petstoreorder

        .objects

        .filter(ownpet = requested_own_pet, statusorder__in = [0, 1, 2,
3])

        .count()

            pet_store_user_visit =
Petstoreuservisit(petstoreuser = pet_store_user,
                                                                timevisit = datetime.now(),
                                                                urlopened =
request.build_absolute_uri(),
                                                                newvisit = False)
            pet_store_user_visit.save()
            return render(request,
"petfoodstoreapp/ownpetorders.html", {"ownpet":
requested_own_pets, "ownpets": requested_own_pets_all, "orders":
pet_store_orders_page, "orders_number": pet_store_orders_count,

```



```

                                                                    .first()
        pet_food_results =
pet_food_results.filter(pettypefood =
requested_own_pet.petbreed.pettype)
        if requested_own_pet.flavours:
            pet_food_results =
pet_food_results.filter(mainfavour__in =
requested_own_pet.flavours)
            pet_food_results =
pet_food_results.filter(breedszelevel =
requested_own_pet.petbreed.breedszelevel)
            if requested_own_pet.healthissues:
                pet_store_food_healthissues =
Petstorefoodhealthissue
                    .objects
                    .filter(healthissue__in =
requested_own_pet.healthissues,
                        positive = False)
                    .values_list('storefood',
                        flat = True)
                    .distinct()
                pet_food_results =
pet_food_results.exclude(id__in =
pet_store_food_for_healthissues)
                petstorefoodmatures_for = Petstorefoodmature
                    .objects.filter(maturelevel = requested_own_pet.maturelevel)
                    .values_list('storefood', flat = True)
                    .distinct()
                pet_food_results = pet_food_results
                    .filter(id__in = petstorefoodmatures_for)
                    if requested_own_pet.allergens:
                        pet_food_components = Petstorefoodcomponent
                            .objects
                            .filter(foodcomponent__in = requested_own_pet.allergens)
                            .values_list('storefood')
                            .distinct()
                        pet_food_results = pet_food_results
                            .exclude(id__in = pet_food_components)
                            if not ownpet_id or add_params:
                                if set_petfood_brands:
                                    set_petfood_brands_ready = []
                                    for set_petfood_brand in
set_petfood_brands.split(","):
                                        set_petfood_brand = int(set_petfood_brand)

```

```

        set_petfood_brand = Petstorefoodbrand

.objects

.get(id = set_petfood_brand)

set_petfood_brands_ready.append(set_petfood_brand)
    pet_food_results = pet_food_results

.filter(storefoodbrand__in = set_petfood_brands_ready)
    food_part_title = request.GET.get("title")
    if food_part_title:
        pet_food_results = pet_food_results

.filter(Q(foodtitle__icontains =
food_part_title)|Q(foodcaption__icontains = food_part_title))
    pet_type = request.GET.get("ptype")
    if pet_type:
        pet_type = Petttype
                                .objects
                                .get(pettyname = pet_type)
        pet_food_results =
pet_food_results.filter(pettytypefood = pet_type)
    flavour = request.GET.get("flav")
    if flavour:
        flavour = Flavour
                                .objects
                                .get(flavourname = flavour)
        pet_food_results =
pet_food_results.filter(mainfavour = flavour)
    pet_food_types = request.GET.get("ftypes")
    if pet_food_types:
        pet_food_types_ready = []
        for pet_food_type in pet_food_types.split(","):
            pet_food_type = int(pet_food_type)
            pet_food_types_ready.append(pet_food_type)
        pet_food_results = pet_food_results

.filter(foodtype__in = pet_food_types_ready)
    wet_pet_food_types = request.GET.get("wtypes")
    if wet_pet_food_types and (not pet_food_types or (1
in pet_food_types)):
        wet_pet_food_types_ready = []
        for wet_pet_food_type in
wet_pet_food_types.split(","):
            wet_pet_food_type = int(wet_pet_food_type)

wet_pet_food_types_ready.append(wet_pet_food_type)
    pet_food_results = pet_food_results

.filter(wetfoodtype__in = wet_pet_food_types_ready)
    breed_size_levels = request.GET.get("breedsize")
    if breed_size_levels:

```

```

        breed_size_levels_ready = []
        for breed_size_level in
breed_size_levels.split(","):
            breed_size_level = int(breed_size_level)

breed_size_levels_ready.append(breed_size_level)
        pet_food_results = pet_food_results

.filter(breedsizel__in = breed_size_levels_ready)
        country_producers = request.GET.get("countries")
        if country_producers:
            country_producers_ready = []
            for country_producer in
country_producers.split(","):

country_producers_ready.append(country_producer)
            pet_food_results = pet_food_results

.filter(countryproduced__in = country_producers_ready)
            main_proteincomponents = request.GET.get("protein")
            if main_proteincomponents:
                main_proteincomponents_ready = []
                for main_proteincomponent in
main_proteincomponents.split(","):
                    main_proteincomponent = Foodcomponent

.objects.get(componentname = main_proteincomponent)

main_proteincomponents_ready.append(main_proteincomponent)
            pet_food_results = pet_food_results

.filter(proteincomponent__in = main_proteincomponents_ready)
            onpurposes = request.GET.get("purposes")
            if onpurposes:
                onpurposes_ready = []
                for onpurpose in onpurposes.split(","):
                    onpurpose = int(onpurpose)
                    onpurposes_ready.append(onpurpose)
                pet_food_results = pet_food_results

.filter(onpurpose__in = onpurposes_ready)
            classtiers = request.GET.get("classtier")
            if classtiers:
                classtiers_ready = []
                for classtier in classtiers.split(","):
                    classtier = int(classtier)
                    classtiers_ready.append(classtier)
                pet_food_results =
pet_food_results.filter(classtier__in = classtiers_ready)
            calories_min = request.GET.get("calorymin")
            if calories_min:
                calories_min = int(calories_min)
                pet_food_results = pet_food_results

```

```

        .filter(calories__gte = calories_min)
        calories_max = request.GET.get("calorymax")
        if calories_max:
            calories_max = int(calories_max)
            pet_food_results = pet_food_results

    .filter(calories__lte = calories_max)
    healthissues = request.GET.get("health")
    healthissues_positive = request.GET.get("positive")
    if healthissues:
        healthissues_ready = []
        for healthissue in healthissues.split(","):
            healthissue =
Healthissue.objects.get(healthissuetitle = healthissue)
            healthissues_ready.append(healthissue)
        if healthissues_positive:
            pet_store_food_healthissues =
Petstorefoodhealthissue
                .objects
                .filter(healthissue__in =
healthissues_ready,
                    positive = True)
        else:
            pet_store_food_healthissues =
Petstorefoodhealthissue
                .objects
                .filter(healthissue__in =
healthissues_ready,
                    positive = False)
            pet_store_food_for_healthissues =
pet_store_food_healthissues
                .values_list('storefood',
                    flat = True)
                .distinct()
            pet_food_results =
pet_food_results.filter(id__in =
pet_store_food_for_healthissues)
            mature_levels = request.GET.get("mature")
            if mature_levels:
                mature_levels_ready = []
                for mature_level in mature_levels.split(","):
                    mature_level = int(mature_level)
                    mature_levels_ready.append(mature_level)
                petstorefoodmatures_for = Petstorefoodmature

.objects.filter(maturelevel__in = mature_levels_ready)

.values_list('storefood', flat = True)

.distinct()
            pet_food_results = pet_food_results

.filter(id__in = petstorefoodmatures_for)

```

```

        food_components_to = request.GET.get("components")
        if food_components_to:
            food_components_to_ready = []
            for food_component_to in
food_components_to.split(","):
                food_component_to = int(food_component_to)

food_components_to_ready.append(food_component_to)
        petstorefoodmatures_for = Petstorefoodcomponent
.objects.filter(foodcomponent__in = food_components_to_ready)
.values_list('storefood', flat = True)

.distinct()
        pet_food_results = pet_food_results

.filter(id__in = petstorefoodmatures_for)
        food_components_to = request.GET.get("allergens")
        if food_components_to:
            food_components_to_ready = []
            for food_component_to in
food_components_to.split(","):
                food_component_to = int(food_component_to)

food_components_to_ready.append(food_component_to)
        petstorefoodmatures_for = Petstorefoodcomponent
.objects.filter(foodcomponent__in = food_components_to_ready)
.values_list('storefood', flat = True)

.distinct()
        pet_food_results = pet_food_results

.exclude(id__in = petstorefoodmatures_for)
        if ownpet_id:
            for pet_food_result in pet_food_results:
                if pet_food_result in
requested_own_pet.storefoods:
                    pet_food_result.forpet = True
                else:
                    pet_food_result.forpet = False
        pet_store_food_trays = Petstorefoodtray
                                .objects

.filter(availablepositions__gt = 0)
        pet_store_tray_foods = pet_store_food_trays

.values_list('storefood', flat = True)
                                .distinct()
        pet_food_results = pet_food_results
        .filter(id__in = pet_store_tray_foods)

```

```

        for pet_food_result in pet_food_results:
            pet_food_result.photo = Petstorefoodphoto
                                                .objects

        .filter(storefood = pet_food_result,
                                                isposter
= True)
            pet_food_result_trays = Petstorefoodtray
                                                .objects

        .filter(storefood = pet_food_result)

        .order_by('currentprice')
            pet_food_result.minprice =
pet_food_result_trays.first()
            pet_food_result.minprice =
pet_food_result.minprice.currentprice
            pet_food_result.maxprice = pet_food_result.last()
            pet_food_result.maxprice =
pet_food_result.maxprice.currentprice
            pet_food_order = request.GET.get("order")
            pet_food_order = int(pet_food_order)
            if pet_food_order == 0:
                pet_food_results = pet_food_results

        .order_by('foodtitle')
            elif pet_food_order == 1:
                pet_food_results = pet_food_results
                                                .order_by('-
foodtitle')
            elif pet_food_order == 2:
                pet_food_results = pet_food_results
                                                .order_by('-
minprice')
            elif pet_food_order == 3:
                pet_food_results = pet_food_results

        .order_by('minprice')
            pet_food_results_count = pet_food_results
                                                .count()

            pet_food_results_paginator =
Paginator(pet_food_results, 10)
            pet_food_results_page = request.GET.get("page")
            pet_food_results_page =
pet_food_results_paginator.get_page(pet_food_results_page)

            pet_store_user_visit = Petstoreuservisit(petstoreuser =
pet_store_user,
                timevisit = datetime.now(),
                urlopened = request.build_absolute_uri(),
                newvisit = False)
            pet_store_user_visit.save()

```

```

        return render(request,
            "petfoodstoreapp/petfoodsearch.html", {"foods":
pet_food_results_page, "food_count": pet_food_results_count,
"brands": petfood_brands, "health_issues": health_issues,
"ownpets": requested_own_pets})

    def pet_type_food_view(request):
        petfood_brands = Petstorefoodbrand
                                .objects
                                .all()

        .order_by("foodbrandname")
        health_issues = Healthissue
                                .objects
                                .all()

        .order_by("healthissuetitle")
        set_petfood_brands = request.GET.get("brands")
        pet_food_results = Petstorefood
                                .objects
                                .all()

        pet_food_components = Foodcomponent
                                .objects
                                .all()

        ownpet_id = request.GET.get("ownpet")
        add_params = request.GET.get("add")
        if ownpet_id:
            ownpet_id = int(ownpet_id)
            if request.user.is_authenticated:
                pet_store_user = request.user
                requested_own_pets = Ownpet
                                .objects

        .prefetch_related("allergens", "flavours", "storefoods",
"healthissues")

        .select_related("petbreed__pettype")

                                .filter(petuserowner =
pet_store_user)
            requested_own_pet = requested_own_pets
                                .filter(id =
ownpet_id)
                                .first()

            pet_food_results =
pet_food_results.filter(pettypefood =
requested_own_pet.petbreed.pettype)
            if requested_own_pet.flavours:
                pet_food_results =
pet_food_results.filter(mainfavour__in =
requested_own_pet.flavours)

```

```

        pet_food_results =
pet_food_results.filter(breedszelevel =
requested_own_pet.petbreed.breedszelevel)
        if requested_own_pet.healthissues:
            pet_store_food_healthissues =
Petstorefoodhealthissue
                .objects
                .filter(healthissue__in =
requested_own_pet.healthissues,
                    positive = False)
                .values_list('storefood',
                    flat = True)
                .distinct()
            pet_food_results =
pet_food_results.exclude(id__in =
pet_store_food_for_healthissues)
            petstorefoodmatures_for = Petstorefoodmature

.objects.filter(maturelevel = requested_own_pet.maturelevel)

.values_list('storefood', flat = True)

.distinct()
        pet_food_results = pet_food_results

.filter(id__in = petstorefoodmatures_for)
        if requested_own_pet.allergens:
            pet_food_components = Petstorefoodcomponent

.objects

.filter(foodcomponent__in = requested_own_pet.allergens)

.values_list('storefood')

.distinct()
        pet_food_results = pet_food_results

.exclude(id__in = pet_food_components)
        if not ownpet_id or add_params:
            if set_petfood_brands:
                set_petfood_brands_ready = []
                for set_petfood_brand in
set_petfood_brands.split(","):
                    set_petfood_brand = int(set_petfood_brand)
                    set_petfood_brand = Petstorefoodbrand

.objects

.get(id = set_petfood_brand)

set_petfood_brands_ready.append(set_petfood_brand)
        pet_food_results = pet_food_results

```

```

        .filter(storefoodbrand__in = set_petfood_brands_ready)
        if not ownpet_id:
            pet_type = request.GET.get("ptype")
            if pet_type:
                pet_type = Pettype
                                .objects
                                .get(pettyname =
pet_type)

                pet_food_results =
pet_food_results.filter(pettyname = pet_type)
                flavour = request.GET.get("flav")
                if flavour:
                    flavour = Flavour
                                .objects
                                .get(flavourname = flavour)

                pet_food_results =
pet_food_results.filter(mainflavour = flavour)
                pet_food_types = request.GET.get("ftypes")
                if pet_food_types:
                    pet_food_types_ready = []
                    for pet_food_type in pet_food_types.split(","):
                        pet_food_type = int(pet_food_type)
                        pet_food_types_ready.append(pet_food_type)
                    pet_food_results = pet_food_results

        .filter(foodtype__in = pet_food_types_ready)
                wet_pet_food_types = request.GET.get("wtypes")
                if wet_pet_food_types and (not pet_food_types or (1
in pet_food_types)):
                    wet_pet_food_types_ready = []
                    for wet_pet_food_type in
wet_pet_food_types.split(","):
                        wet_pet_food_type = int(wet_pet_food_type)

wet_pet_food_types_ready.append(wet_pet_food_type)
                pet_food_results = pet_food_results

        .filter(wetfoodtype__in = wet_pet_food_types_ready)
                if not ownpet_id:
                    breed_size_levels =
request.GET.get("breedsizes")
                    if breed_size_levels:
                        breed_size_levels_ready = []
                        for breed_size_level in
breed_size_levels.split(","):
                            breed_size_level =
int(breed_size_level)

breed_size_levels_ready.append(breed_size_level)
                pet_food_results = pet_food_results

        .filter(breedsizeslevel__in = breed_size_levels_ready)
                country_producers = request.GET.get("countries")

```

```

        if country_producers:
            country_producers_ready = []
            for country_producer in
country_producers.split(","):
country_producers_ready.append(country_producer)
            pet_food_results = pet_food_results

.filter(countryproduced__in = country_producers_ready)
        main_proteincomponents = request.GET.get("protein")
        if main_proteincomponents:
            main_proteincomponents_ready = []
            for main_proteincomponent in
main_proteincomponents.split(","):
                main_proteincomponent = Foodcomponent

.objects.get(componentname = main_proteincomponent)

main_proteincomponents_ready.append(main_proteincomponent)
            pet_food_results = pet_food_results

.filter(proteincomponent__in = main_proteincomponents_ready)
        onpurposes = request.GET.get("purposes")
        if onpurposes:
            onpurposes_ready = []
            for onpurpose in onpurposes.split(","):
                onpurpose = int(onpurpose)
                onpurposes_ready.append(onpurpose)
            pet_food_results = pet_food_results

.filter(onpurpose__in = onpurposes_ready)
        classtiers = request.GET.get("classtier")
        if classtiers:
            classtiers_ready = []
            for classtier in classtiers.split(","):
                classtier = int(classtier)
                classtiers_ready.append(classtier)
            pet_food_results =
pet_food_results.filter(classtier__in = classtiers_ready)
        calories_min = request.GET.get("calorymin")
        if calories_min:
            calories_min = int(calories_min)
            pet_food_results = pet_food_results

.filter(calories__gte = calories_min)
        calories_max = request.GET.get("calorymax")
        if calories_max:
            calories_max = int(calories_max)
            pet_food_results = pet_food_results

.filter(calories__lte = calories_max)
        if not ownpet_id:
            healthissues = request.GET.get("health")

```

```

        healthissues_positive =
request.GET.get("positive")
        if healthissues:
            healthissues_ready = []
            for healthissue in healthissues.split(","):
                healthissue =
Healthissue.objects.get(healthissuetitle = healthissue)
                healthissues_ready.append(healthissue)
            if healthissues_positive:
                pet_store_food_healthissues =
Petstorefoodhealthissue

.objects

.filter(healthissue__in = healthissues_ready,
positive = True)
                else:
                    pet_store_food_healthissues =
Petstorefoodhealthissue

.objects

.filter(healthissue__in = healthissues_ready,
positive = False)
                    pet_store_food_for_healthissues =
pet_store_food_healthissues
                        .values_list('storefood',
                            flat = True)
                        .distinct()
                    pet_food_results =
pet_food_results.filter(id__in =
pet_store_food_for_healthissues)
                    if not ownpet_id:
                        mature_levels = request.GET.get("mature")
                        if mature_levels:
                            mature_levels_ready = []
                            for mature_level in
mature_levels.split(","):
                                mature_level = int(mature_level)

mature_levels_ready.append(mature_level)
                            petstorefoodmatures_for =
Petstorefoodmature

.objects.filter(maturelevel__in = mature_levels_ready)

.values_list('storefood', flat = True)

.distinct()
                                pet_food_results = pet_food_results
                                .filter(id__in = petstorefoodmatures_for)

```

```

        food_components_to = request.GET.get("compoments")
        if food_components_to:
            food_components_to_ready = []
            for food_component_to in
food_components_to.split(","):
                food_component_to = int(food_component_to)

food_components_to_ready.append(food_component_to)
        petstorefoodmatures_for = Petstorefoodcomponent
.objects.filter(foodcomponent__in = food_components_to_ready)
.values_list('storefood', flat = True)

.distinct()
        pet_food_results = pet_food_results

.filter(id__in = petstorefoodmatures_for)
        if not ownpet_id:
            food_components_to =
request.GET.get("allergens")
            if food_components_to:
                food_components_to_ready = []
                for food_component_to in
food_components_to.split(","):
                    food_component_to =
int(food_component_to)

food_components_to_ready.append(food_component_to)
                petstorefoodmatures_for =
Petstorefoodcomponent

.objects.filter(foodcomponent__in = food_components_to_ready)
.values_list('storefood', flat = True)

.distinct()
        pet_food_results = pet_food_results

.exclude(id__in = petstorefoodmatures_for)
        if ownpet_id:
            for pet_food_result in pet_food_results:
                if pet_food_result in
requested_own_pet.storefoods:
                    pet_food_result.forpet = True
                else:
                    pet_food_result.forpet = False
        pet_store_food_trays = Petstorefoodtray
                                .objects

.filter(availablepositions__gt = 0)
        pet_store_tray_foods = pet_store_food_trays
        .values_list('storefood', flat = True)

```

```

pet_food_results = pet_food_results.distinct()

.filter(id__in = pet_store_tray_foods)
    for pet_food_result in pet_food_results:
        pet_food_result.photo = Petstorefoodphoto.objects

.filter(storefood = pet_food_result,
        isposter
= True)
        pet_food_result_trays = Petstorefoodtray.objects

.filter(storefood = pet_food_result)

.order_by('currentprice')
        pet_food_result.minprice =
pet_food_result_trays.first()
        pet_food_result.minprice =
pet_food_result.minprice.currentprice
        pet_food_result.maxprice = pet_food_result.last()
        pet_food_result.maxprice =
pet_food_result.maxprice.currentprice
        pet_food_order = request.GET.get("order")
        pet_food_order = int(pet_food_order)
        if pet_food_order == 0:
            pet_food_results = pet_food_results

.order_by('foodtitle')
        elif pet_food_order == 1:
            pet_food_results = pet_food_results.order_by('-
foodtitle')
        elif pet_food_order == 2:
            pet_food_results = pet_food_results.order_by('-
minprice')
        elif pet_food_order == 3:
            pet_food_results = pet_food_results

.order_by('minprice')
        pet_food_results_count = pet_food_results.count()

        pet_food_results_paginator =
Paginator(pet_food_results, 10)
        pet_food_results_page = request.GET.get("page")
        pet_food_results_page =
pet_food_results_paginator.get_page(pet_food_results_page)
        pet_store_user_visit = Petstoreuservisit(petstoreuser =
pet_store_user,
            timevisit = datetime.now(),
            urlopened = request.build_absolute_uri(),

```

```

        newvisit = False)
    pet_store_user_visit.save()
    return render(request,
"petfoodstoreapp/pettytypefoodsearch.html", {"foods":
pet_food_results_page, "food_count": pet_food_results_count,
"brands": petfood_brands, "health_issues": health_issues,
"ownpets": requested_own_pets})

    def pet_food_view(request):
        petfoodId = request.GET.get("petfood")
        petfoodId = int(petfoodId)
        pet_food = Petstorefood
                                .objects
                                .get(id = petfoodId)
        pet_food_result_trays = Petstorefoodtray
                                .objects

    .filter(storefood = pet_food)

    .order_by('trayweight')
        pet_food.minprice = pet_food_result_trays.first()
        pet_food.minprice = pet_food.minprice.currentprice
        pet_food.maxprice = pet_food_result.last()
        pet_food.maxprice = pet_food.maxprice.currentprice
        matures = Petstorefoodmature
                                .objects
                                .filter(storefood =
pet_food)
        health_issues = Petstorefoodhealthissue
                                .objects

    .select_related("healthissue")

    .filter(storefood = pet_food)
        pet_food_photos = Petstorefoodphoto
                                .objects
                                .filter(storefood =
pet_food)
        pet_food_comments = Petstorefoodcomment
                                .objects

    .select_related("petstoreuser")

    .filter(storefood = pet_food)
                                .order_by("-
maincommenttime")
        pet_food_components = Petstorefoodcomponent
                                .objects

    .select_related("foodcomponent")

    .filter(storefood = pet_food)
        requested_own_pets_all = None

```

```

        requested_own_pet = None
        if request.user.is_authenticated:
            pet_store_user = request.user
            requested_own_pets_all = Ownpet
                                .objects
                                .prefetch_related("allergens", "flavours", "storefoods", "healthissues")
                                .select_related("petbreed")
                                .filter(petuserowner =
pet_store_user)
            if own_pet:
                requested_own_pet =
requested_own_pets_all.filter(id = own_pet)
                pet_store_user_visit =
Petstoreuservisit(petstoreuser = pet_store_user,
                    timevisit = datetime.now(),
                    urlopened = request.build_absolute_uri(),
                    newvisit = False)
                pet_store_user_visit.save()
            return render(request, "petfoodstoreapp/petfood.html",
{"petfood": pet_food, "trays": pet_food_result_trays, "matures":
matures, "health_issues": health_issues, "photos":
pet_food_photos, "comments": pet_food_comments, "components":
pet_food_components, "ownpet": requested_own_pet, "ownpets":
requested_own_pets_all })

```

ДОДАТОК В
Слайди презентації

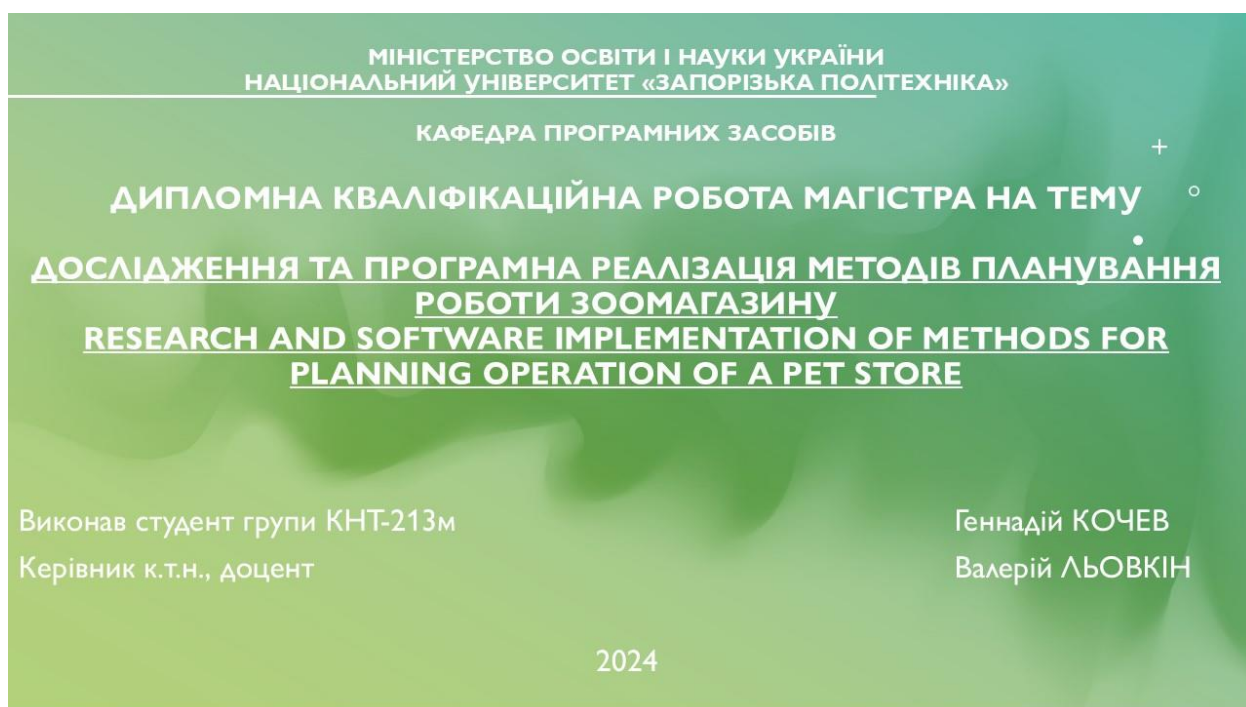


Рисунок В.1 – Початковий слайд

Об'єкт, предмет, мета роботи

- об'єкт роботи – процес роботи зоомагазину;
- предмет роботи – методи планування роботи зоомагазину;
- мета роботи – розробити програмне забезпечення підтримки роботи зоомагазину для підвищення інформованості прийняття рішень стосовно планування роботи зоомагазину.

Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

- виконати аналіз програмного забезпечення підтримки роботи зоомагазину;
- розробити метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи;
- виконати проєктування програмного забезпечення підтримки роботи зоомагазину;
- реалізувати програмне забезпечення підтримки роботи зоомагазину;
- виконати експериментальне дослідження класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи зоомагазину.

3

Рисунок В.3 – Завдання роботи

Порівняння аналогів розробки вебзастосунку підтримки роботи зоомагазину

Показник	Master ZOO	Pethouse	Розробка
Пошук кормів	За назвою	За назвою	За параметрами
Відтворення переліку кормів	За параметрами	За параметрами	За параметрами
Замовлення кормів з оформленням підписки	–	–	+
Визначення власних тварин	–	–	+
Пошук кормів з врахуванням характеристик тварин	–	–	+
Класифікація замовлень стосовно подальшого замовлення вологого корму	–	–	+

4

Рисунок В.4 – Порівняння аналогів розробки вебзастосунку підтримки роботи зоомагазину

Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи



5

Рисунок В.5 – Метод класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

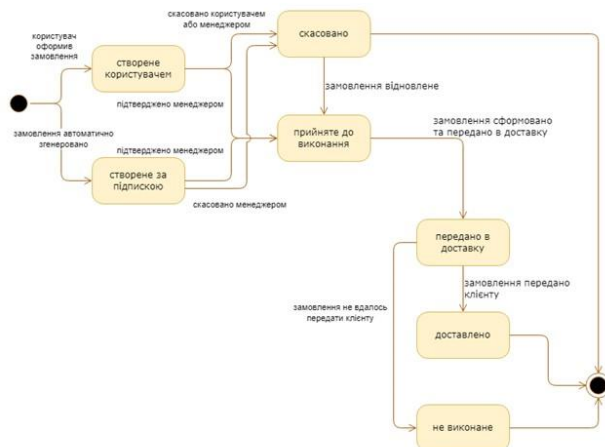
Вибір вебфреймворку для розробки програмного забезпечення підтримки роботи зоомагазину

	Django	Flask
Придатність для реалізації машинного навчання	Розробка ведеться мовою Python, доступні всі Python-пакети для роботи з даними та машинного навчання	
Наявні готові засоби розробки	Авторизація, механізм шаблонізації, адміністративна панель, об'єктно-реляційне відображення	Потребує встановлення і налаштування таких засобів, а відповідно і вибору
Час розробки	Пришвидшений	Порівняно більший

6

Рисунок В.6 – Вибір вебфреймворку для розробки програмного забезпечення підтримки роботи зоомагазину

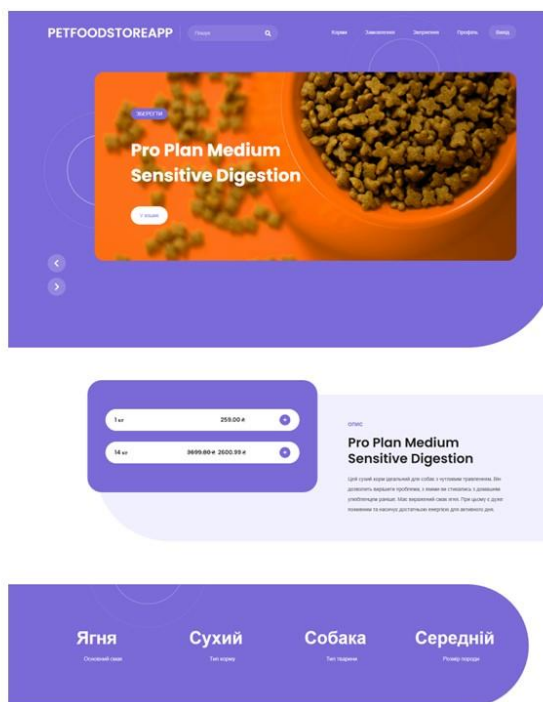
Діаграма станів замовлення корму для тварин



9

Рисунок В.9 – Діаграма станів замовлення корму для тварин

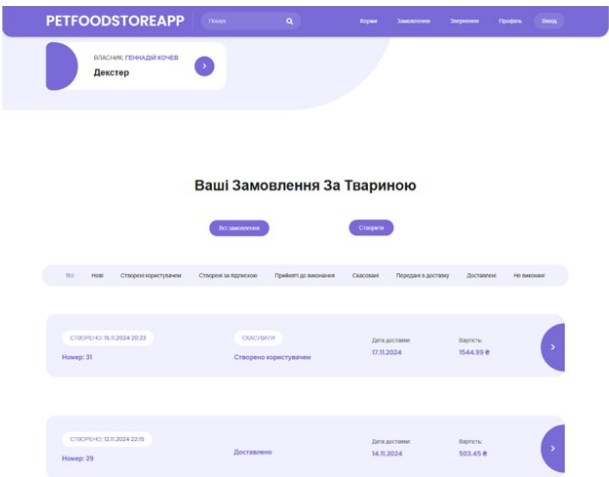
Вебсторінка відтворення даних корму для тварин



10

Рисунок В.10 – Вебсторінка відтворення даних корму для тварин

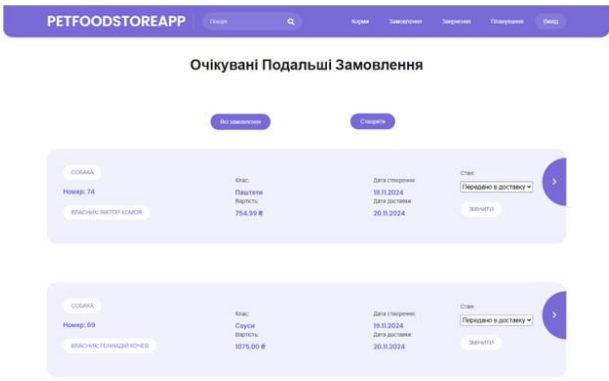
Вебсторінка
відтворення
повного переліку
замовлень кормів
за твариною



11

Рисунок В.11 – Вебсторінка відтворення повного переліку замовлень кормів за твариною

Вебсторінка
відтворення
переліку очікуваних
подальших
замовлень вологого
корму



12

Рисунок В.12 – Вебсторінка відтворення переліку очікуваних подальших замовлень вологого корму

Вибір ознак

Ознака	Оцінка
Кілокалорії вологого корму	24423404,66
Кілокалорії сухого корму	3510537,56
Сумарна кількість кілокалорій корму	603225,23
Кількість позицій вологого корму в замовленні	150521,80
Сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті	46041,46
Порядковий номер замовлення за даною твариною серед всіх її замовлень, в яких було визначено вологий корм додатково до сухого	42544,10
Сумарна кількість хвилин, проведена власником тварини як користувачем на вебсайті з моменту останнього замовлення	18793,36
Кількість замовлень, які були оформлені після першого замовлення вологого корму цим користувачем для своєї тварини	15680,24
Кількість днів останнього звернення в службу підтримки	13620,04

13

Рисунок В.13 – Вибір ознак

Результати класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

	Точність	Влучність для непродовженог о замовлення	Влучність для подальшого замовлення
Запропонований метод	0,974607696	0,982630594	0,851304581
CART	0,9657123	0,980759981	0,753278043
Метод k-середніх	0,939430589	0,951686876	0,433877694

14

Рисунок В.14 – Результати класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи

Висновки

- У роботі розроблено програмне забезпечення підтримки роботи зоомагазину, що цілісно дозволило підвищити інформованість прийняття рішень стосовно планування роботи зоомагазину, таким чином досягнувши мету роботи.
- У роботі реалізовано вебзастосунок зоомагазину, який призначений для підтримки роботи зоомагазину шляхом пошуку, перегляду інформації, яка стосується кормів для тварин, замовлення цих кормів та керування такими замовленнями, визначення наявної кількості кормів у зоомагазині на основі даних про продані корми, наявні запаси та очікувані замовлення, що реалізовано за допомогою виконання класифікації замовлень на основі навченої моделі з використанням запропонованого методу. Для реалізації використано мову Python разом з набором пакетів для роботи з даними та машинного навчання, фреймворк веброзробки Django разом з системою керування базами даних PostgreSQL.
- Наукова новизна роботи полягає в методі класифікації замовлень в зоомагазині стосовно подальшого замовлення вологого корму для планування роботи, який виконує класифікацію замовлень на основі моделі, побудованої методом Random Forest, визначає виконання відбору ознак для побудови даної моделі для конкретного типу вологого корму, підготовку вибірки даних для відповідного навчання моделі.

15

Рисунок В.15 – Висновки