

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 проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ХАРЧУВАННЯ ДОМАШНІХ ТВАРИН
RESEARCH AND SOFTWARE IMPLEMENTATION OF
AN APPLICATION FOR PET NUTRITION MANAGEMENT

Виконав(ла): студент(ка) 2 курсу, групи КНТ-114м
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КОВАЛЬ О.О.

(ПРІЗВИЩЕ та ініціали)

Керівник КОЛПАКОВА Т.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент ІЛ'ЯШЕНКО М.Б.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2025 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОВАЛЯ Олега Олеговича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація застосунку для організації харчування домашніх тварин. Research and Software Implementation of an Application for Pet Nutrition Management

керівник проєкту (роботи) к.т.н., доцент, КОЛПАКОВА Тетяна Олексіївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 30 ” вересня 2025 року № 447

2. Строк подання студентом проєкту (роботи) 02 грудня 2025 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	КОЛПАКОВА Т.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 30 ” вересня 2025 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження програмного забезпечення.	8 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	9-10 тижні	Додатки
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Олег КОВАЛЬ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Тетяна КОЛПАКОВА
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
109 с., 3 табл., 46 рис., 3 дод., 15 джерел.

ASP.NET, C#, VISUAL STUDIO, ДОМАШНІ ТВАРИНИ,
ЗАСТОСУНОК, МОВА ПРОГРАМУВАННЯ, ОРГАНІЗАЦІЯ ХАРЧУВАННЯ,
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Об'єкт дослідження – процес розробки програмного забезпечення для організації харчування домашніх тварин.

Предмет дослідження – програмні засоби для організації харчування домашніх тварин.

Мета роботи – дослідження та розробка програмного забезпечення для організації харчування домашніх тварин.

Матеріали, методи та технічні засоби: мова програмування C#, середовище розробки Visual Studio.

Результати. Розроблено програмне забезпечення для організації харчування домашніх тварин за допомогою мови програмування C# та середовища розробки Visual Studio.

Практична цінність роботи полягає у розробці програмного забезпечення для організації харчування домашніх тварин.

Висновки. Виконано проєктування програмного забезпечення для організації харчування домашніх тварин. Розроблено програмне забезпечення для організації харчування домашніх тварин. Здійснено тестування розробленого програмного забезпечення для організації харчування домашніх тварин.

Галузь використання – організації харчування домашніх тварин.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 109 pages, 3 tables, 46 figures, 3 appendixes, 15 sources.

ASP.NET, C#, VISUAL STUDIO, PETS, APPLICATION, PROGRAMMING LANGUAGE, FOOD ORGANIZATION, SOFTWARE.

The object of research is the process of developing software for organizing pet food.

The subject of the research is software for organizing pet food.

The purpose of this work is to develop the software for organizing pet food.

Materials, methods and technical tools: C# programming language, Visual Studio development environment.

Results. The software tools for organizing pet food using the C# programming language and the Visual Studio development environment have been developed.

The practical value of the work lies in the development of software for organizing pet food.

Conclusions. The software for organizing pet food has been designed. The software for organizing pet food has been developed. The developed software for organizing pet food has been tested.

Scope of use – pet food organization.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Програмне забезпечення для організації харчування домашніх тварин	11
1.2 Аналіз програмного забезпечення для організації харчування домашніх тварин.....	14
1.3 Висновки за розділом 1	25
2 Матеріали і методи	26
2.1 Вибір мови програмування	26
2.2 Вибір середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин	29
2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин	33
2.4 Висновки за розділом 2	36
3 Опис програми	37
3.1 Структура програмного забезпечення для організації харчування домашніх тварин	37
3.2 Функціонування програмного забезпечення для організації харчування домашніх тварин	41
3.3 Розробка бази даних програмного забезпечення для організації харчування домашніх тварин	45
3.4 Проектування інтерфейсу програмного забезпечення для організації харчування домашніх тварин	47
3.5 Висновки за розділом 3	51
4 Експлуатація, тестування та експериментальне дослідження програми.....	53
4.1 Призначення й умови застосування програми	53
4.2 Характеристики програми для організації харчування домашніх тварин.....	54

4.3 Інструкція по експлуатації програми.....	55
4.3.1 Звернення до програми.....	55
4.3.2 Вхідні й вихідні дані	55
4.3.3 Повідомлення.....	55
4.4 Виконання програмного забезпечення для організації харчування домашніх тварин	56
4.5 Тестування програмного забезпечення для організації харчування домашніх тварин	63
4.6 Висновки за розділом 4	63
Висновки	64
Перелік джерел посилання	67
Додаток А Технічне завдання	69
Додаток Б Фрагмент тексту програми	73
Додаток В Слайди презентації	101

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

БД – база даних;

ДТ – домашня тварина;

ПЗ – програмне забезпечення.

ВСТУП

Актуальність програмного забезпечення для організації харчування домашніх тварин зумовлена низкою сучасних тенденцій, що охоплюють як соціальні, так і технологічні зміни. У багатьох країнах спостерігається стабільне зростання кількості домашніх улюбленців, особливо в умовах міського життя. Власники тварин дедалі більше уваги приділяють питанням якості їхнього харчування, здоров'я та тривалості життя. Це формує запит на інструменти, які дозволяють індивідуалізувати раціон залежно від породи, віку, ваги, рівня активності й особливостей здоров'я тварини [1], [2].

Програмне забезпечення в цьому контексті відіграє важливу роль, оскільки дає змогу оптимізувати підбір корму, контролювати кількість спожитої їжі, враховувати медичні рекомендації, уникати алергенів і своєчасно виявляти зміни у фізичному стані тварини. Використання таких програм, особливо в поєднанні з розумними годівницями, дозволяє автоматизувати процес годування, відслідковувати динаміку маси тіла та отримувати попередження у разі відхилень від звичного режиму [3], [4].

Крім того, накопичення даних у цифровому форматі полегшує взаємодію з ветеринарами, сприяє точнішій діагностиці та плануванню дієт, а також створює умови для віддаленого моніторингу. Це особливо важливо для тварин із хронічними захворюваннями або післяопераційним станом [5], [6].

Ринок технологій для домашніх тварин стрімко розвивається, з'являється все більше мобільних додатків, онлайн сервісів і smart-рішень, орієнтованих на догляд за тваринами. Таким чином, програмне забезпечення для організації харчування улюбленців є не лише актуальним, а й перспективним напрямом, що поєднує турботу про тварин з можливостями цифрових технологій і сприяє формуванню відповідального підходу до утримання домашніх тварин [7], [8].

Проте деякі програми для організації харчування домашніх тварин можуть мати інтерфейс, який є надмірно складним або перевантаженим, що

ускладнює їх використання для людей похилого віку або користувачів без технічної підготовки. Інколи користувачі стикаються з платними функціями або нав'язливими покупками в додатку, що знижує рівень довіри до програмного продукту. Технічні обмеження також відіграють роль. Деяке програмне забезпечення для організації харчування домашніх тварин потребує постійного доступу до Інтернету, синхронізації з розумними пристроями або оновлень, що не завжди зручно для користувача. Крім того, в разі збоїв у роботі чи відсутності доступу до сервера функціональність додатку може тимчасово бути втрачена. Ще одним недоліком є обмежена адаптація під різні види та породи тварин [1]-[6].

Тому актуальною є розробка програмного забезпечення для організації харчування домашніх тварин. У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання розробки та дослідження програмного забезпечення для організації харчування домашніх тварин.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для організації харчування домашніх тварин;
- розробити модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин;
- здійснити проєктування програмного забезпечення для організації харчування домашніх тварин;
- створити програмне забезпечення для організації харчування домашніх тварин;
- виконати тестування розробленого програмного забезпечення для організації харчування домашніх тварин.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення для організації харчування домашніх тварин

Програмне забезпечення для організації харчування домашніх тварин — це спеціалізована інформаційна система, що забезпечує автоматизоване планування, моніторинг і оптимізацію раціону тварин залежно від їхніх індивідуальних характеристик. Його головною метою є покращення якості догляду за домашніми улюбленцями шляхом точного розрахунку добових норм харчування, врахування особливостей здоров'я, віку, породи, фізичної активності та харчових обмежень [1], [2].

Функціональні можливості такого програмного забезпечення включають створення персоналізованих харчових планів, рекомендації щодо підбору готових кормів або домашніх рецептів, ведення історії годувань, а також надання сповіщень про час наступного прийому їжі. Інтерфейс орієнтований на зручність користувача: усі параметри вводяться вручну або автоматично підтягуються з бази, після чого система пропонує оптимальний набір продуктів або готових товарів для замовлення [1]-[3].

Таке ПЗ може бути інтегроване з онлайн магазином, що дозволяє не лише формувати раціон, а й замовляти продукти, необхідні для його реалізації. У розширених версіях реалізується підтримка зв'язку з ветеринарними кабінетами, а також можливість підключення розумних пристроїв, які здійснюють контроль за годівлею в автоматичному режимі [3], [4].

Програмне забезпечення для організації харчування домашніх тварин розробляється з використанням сучасних мов програмування, наприклад C#, і може працювати на різних платформах — настільних, мобільних або веборієнтованих. Воно також підтримує інтеграцію з базами даних, забезпечує зберігання профілів тварин, логіку замовлень та історію харчових змін [4], [5].

Таким чином, така програмна система є ефективним інструментом для тих, хто прагне забезпечити своєму улюбленцю збалансоване та своєчасне

харчування, покладаючись на дані, аналітику та автоматизовану підтримку прийняття рішень [5], [6].

Програмне забезпечення, призначене для організації харчування домашніх тварин, функціонує як цифровий помічник у повсякденному догляді за улюбленицями, забезпечуючи інтелектуальне управління їхнім раціоном. Його робота спрямована на те, щоб враховувати всі значущі параметри, пов'язані з фізіологічним станом тварини, її звичками, способом життя та особливостями здоров'я [5], [6].

Суттєва увага у таких програмах може приділятися персоналізації: інформація, яка вводиться про тварину, стає основою для формування адаптованих харчових рекомендацій. Завдяки гнучкій структурі програмне середовище дозволяє автоматизувати підбір інгредієнтів або кормів з урахуванням харчових потреб, уникати небажаних продуктів та підтримувати оптимальний баланс поживних речовин [5], [6].

Крім аналітичних функцій, такі програми можуть містити також логістичний компонент: облік наявних запасів корму, нагадування про необхідність поповнення, а в деяких випадках — взаємодія з платформами електронної комерції для оформлення замовлень. Такий підхід не лише оптимізує процес годування, а й значно економить час власника [1], [3].

У випадках, коли застосовуються розумні годівниці, програмне забезпечення може керувати ними дистанційно, забезпечуючи точне дозування їжі у визначений час. Підключення до хмарних сервісів дозволяє зберігати налаштування, синхронізувати дані між пристроями та забезпечувати доступ до системи з будь-якого місця [2], [5].

У системі організації харчування домашніх тварин програмне забезпечення відіграє роль універсального координатора, що об'єднує різноманітні джерела даних для формування цілісного підходу до годівлі. Його застосування забезпечує зменшення впливу випадковості у прийнятті рішень щодо раціону, замінюючи інтуїтивні дії користувача на обґрунтовані розрахунки, побудовані на об'єктивних параметрах [5], [6].

Цифрова система дозволяє не просто фіксувати інформацію про харчування, а й динамічно адаптувати плани залежно від змін у вазі тварини, сезонних факторів або рекомендацій ветеринарного фахівця. Вона забезпечує надійний контроль за регулярністю годувань, допомагає уникати надлишкового споживання або дефіциту поживних речовин, орієнтуючись на попередньо задані показники [5], [6].

За допомогою внутрішніх механізмів обробки інформації створюється логіка, яка дозволяє не лише підтримувати стабільність харчування, а й прогнозувати його ефективність. Визначення реакцій тварини на певні продукти чи складові дає змогу удосконалювати підхід до підбору корму, формуючи персоналізовану модель харчування [5], [6].

Таке програмне забезпечення здатне адаптуватися до різного рівня цифрової підготовки користувача: від базових шаблонів дій до гнучких налаштувань, орієнтованих на досвідченого власника або професіонала. У середовищі інтенсивної взаємодії між власником, цифровою платформою та зовнішніми сервісами виникає новий рівень керування побутом домашньої тварини, в якому увага до харчування стає частиною більш широкої системи турботи [3], [6].

Завдяки можливості масштабування, такі програмні системи можуть охоплювати не лише окремих улюбленців, а й групи тварин, що відкриває перспективи для її застосування у притулках, розплідниках або ветеринарних клініках. У цьому випадку відстеження, аналіз і планування харчових процесів перетворюється на централізований механізм управління із високим рівнем автоматизації [1], [2].

Важливою особливістю є можливість спостереження за динамікою харчування впродовж тривалого періоду. Завдяки накопиченій інформації стає можливим аналіз змін у поведінці чи стані тварини, що може мати діагностичне значення або допомагати в корекції раціону [5], [6].

Таким чином, програмне забезпечення цієї категорії виконує не лише інструментальну роль, а й виступає посередником між користувачем,

ветеринарними рекомендаціями та реальними потребами тварини, забезпечуючи індивідуальний та системний підхід до організації харчування.

Визначено, що програмне забезпечення для організації харчування домашніх тварин являє собою спеціалізовану систему, що забезпечує автоматизоване планування, моніторинг і оптимізацію раціону тварин залежно від їхніх характеристик. Його головною метою є покращення якості догляду за домашніми улюбленцями шляхом.

1.2 Аналіз програмного забезпечення для організації харчування домашніх тварин

Проаналізуємо програмне забезпечення для організації харчування домашніх тварин [7]-[11].

Програмне забезпечення Pet Food Pro (рис. 1.1) реалізовано як онлайн платформу та невід'ємну частину екосистеми догляду за харчуванням домашніх тварин, що працює як посередник між користувачем, експертами з харчування й постачальниками корму. Цей вебсайт призначений для автоматизації підбору корму за форматом підписки, прямої доставки та персоналізованих порад завдяки можливості спілкування з фахівцями. Особлива увага приділяється досвіду користувача: інтерфейс оптимізовано для швидкого доступу до популярних категорій корму й товарів (наприклад, сухі корми, консерва, ласощі та іграшки). Платформа дозволяє оформлювати підписку з автоматичним оновленням доставки, переглядати історію замовлень і стежити за статусом доставки. Програмна реалізація Pet Food Pro спирається на каталоги, що описують склад і поживну цінність кожного продукту, надає детальні описи та відгуки користувачів. Для гнучкого налаштування раціону використовується фільтрація продуктів за категоріями, наявністю алергенів і спеціальними потребами тварин [8], [9].

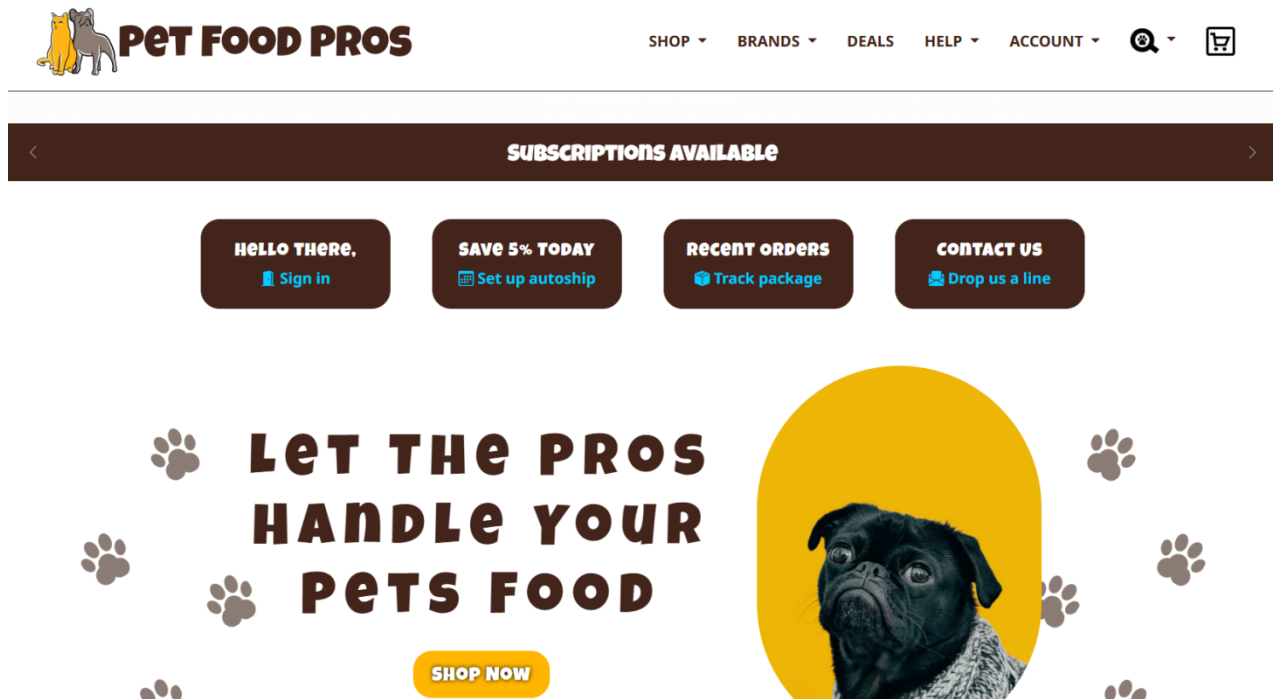


Рисунок 1.1 – Програмне забезпечення Pet Food Pro [9]

Функціонал програмного забезпечення Pet Food Pro дозволяє зберігати обрані товари в кошику, який має можливість оформлення замовлення чи конвертації в підписку з автоматичною доставкою та знижкою. Облік історії замовлень реалізовано за допомогою профілю користувача, що також підтримує налаштування підписки та доступ до підтримки експертів [8], [9].

Архітектурно програмне забезпечення Pet Food Pro працює як вебсервіс, що обробляє запити до бекенду для отримання інформації про товари (шляхом вибору категорій, параметрів харчування і алергенів) і забезпечує механізм оформлення та оплати замовлень. Автентифікація користувача активує профіль із спеціалізованими налаштуваннями: історія покупок, підписка, можливість консультації з експертом. Навіть у разі нестандартного харчування (наприклад, «special needs») система здатна підсилити рекомендації інформацією про склад корму, порцію та доступні варіанти доставки [8], [9].

Таким чином, платформа Pet Food Pro поєднує функціонал онлайн магазину, сервісу підписок та інструменту підтримки харчування тварин із можливістю консультацій. Завдяки цьому залученість користувача

підвищується, а процес підбору валідного, безпечного та зручного корму оптимізується [8], [9].

Основними характеристиками програмного забезпечення Pet Food Pro є такі [8], [9]:

- персоналізований підбір корму: програмне забезпечення Pet Food Pro формує індивідуальні рекомендації на основі параметрів домашньої тварини, таких як порода, вік, вага, алергії або особливі потреби [8], [9];

- інтеграція з експертною підтримкою: у програмному забезпеченні Pet Food Pro передбачено можливість консультування з фахівцями з харчування тварин безпосередньо через веб-платформу [8], [9];

- автоматизована підписка на корм: користувач програмного забезпечення Pet Food Pro може налаштувати регулярну доставку обраного продукту з гнучкими параметрами частоти та обсягу [8], [9];

- керування профілем тварини: реалізовано збереження інформації про одну або декілька тварин із можливістю оновлення стану здоров'я та харчових уподобань [8], [9];

- фільтрація за складом та функцією: користувачі програмного забезпечення Pet Food Pro мають змогу сортувати продукти за інгредієнтами, призначенням, типом харчування та виробником [8], [9];

- підтримка різних форматів товарів: у програмному забезпеченні Pet Food Pro забезпечено вибір між сухими кормами, консервами, ласощами та спеціалізованими дієтами [8], [9];

- історія замовлень та повторна покупка: у програмному забезпеченні Pet Food Pro передбачено доступ до попередніх покупок з можливістю швидкого повторного оформлення [8], [9];

- адаптивний вебінтерфейс: інтерфейс програмного забезпечення Pet Food Pro оптимізований для роботи на різних пристроях, включаючи смартфони, планшети та комп'ютери [8], [9];

- знижки та бонуси за підписку: система заохочує регулярних користувачів знижками на товари при виборі доставки за розкладом [8], [9];

– структурована база товарів: каталог програмного забезпечення Pet Food Pro організовано за категоріями з докладними описами, що допомагає швидко знайти потрібний корм [8], [9].

Серед недоліків програмного забезпечення Pet Food Pro можна відзначити обмежену гнучкість налаштувань для тварин із дуже специфічними дієтичними потребами, що не завжди дозволяє повністю адаптувати систему до індивідуальних випадків. Також певні складнощі можуть виникати у користувачів, які шукають багатомовну підтримку або локалізовані версії інтерфейсу, адже основна мовна адаптація орієнтована на англomовну аудиторію. У деяких випадках функціональність пов'язана із підпискою може обмежувати свободу вибору, змушуючи користувача дотримуватися фіксованих циклів доставки. Крім того, обробка персоналізованих рекомендацій ґрунтується переважно на анкетних даних, що можуть не відображати реального стану здоров'я тварини без участі ветеринарного спеціаліста, що створює ризик неадекватного харчування при неправильному введенні інформації [8], [9].

Програмне забезпечення Formulator (рис. 1.2) представлено у вигляді вебплатформи, орієнтованої на виробників кормів для домашніх тварин. Його призначення полягає в побудові рецептур із урахуванням екологічної відповідальності, якості харчування та нормативної відповідності, що робить його привабливим для брендів з амбіціями щодо сталого розвитку. Програмне забезпечення Formulator вигідно вирізняється тим, що дає можливість замінювати традиційні інгредієнти екологічно «чистими» альтернативами й оцінювати їхній вплив на поживний склад у реальному часі за допомогою інтерактивного конструктора рецептів. Завдяки таким функціям розробники продуктів отримують змогу швидко модифікувати рецептури без необхідності складних розрахунків вручну [8], [10].



Рисунок 1.2 – Програмне забезпечення Formulator [10]

Форматування рецептів у програмному забезпеченні Formulator реалізоване через вбудований механізм порівняння складу з ветеринарно-дієтичними нормами та регуляторними стандартами, що гарантує відповідність виробленого корму очікуванням галузі. Завдяки цьому виробники можуть легко адаптувати склад під специфічні вимоги своєї аудиторії [8], [10].

Звіти, що генеруються у Formulator, є гнучкими й налаштовуваними під різні напрямки — маркетинг, технічні відділи, юридичні служби. Це дозволяє кожній зацікавленій стороні отримувати найбільш релевантні дані для аналізу ефективності, витрат чи відповідності [8], [10].

Платформа також містить інструмент аналізу ринкових тенденцій, що дозволяє оцінювати популярні інгредієнти, цінові тренди та переваги споживачів, тим самим сприяючи стратегічному плануванню асортименту та маркетингової активності [8], [10].

З технічної точки зору Formulator побудований як модульна вебархітектура, де фронтенд-взаємодії з користувачем реалізовані через візуальний інтерфейс конструктора рецептів, а бекенд використовує потужний

алгоритм оптимізації (Ara Integra-Mix®), що забезпечує баланс між вартістю, харчовою цінністю та екологічними показниками [8], [10]. У підсумку, Formulator поєднує принципи сталого розвитку, ефективності рецепт-розробки й відповідності галузевим стандартам, виступаючи повноцінним рішенням для брендів, що прагнуть контролювати якість, екологічність та інноваційність у виробництві харчових продуктів для домашніх тварин [8], [10].

Програмне забезпечення Formulator має такі особливості [8], [10]:

- сталий розвиток: підтримка альтернативних інгредієнтів, які відповідають екологічним стандартам та зменшують негативний вплив на довкілля [8], [10];
- інтерактивне формулювання: візуальний конструктор рецептів програмного забезпечення Formulator дозволяє в режимі реального часу бачити зміни у поживних характеристиках при зміні інгредієнтів [8], [10];
- відповідність нормам: програмне забезпечення Formulator підтримує автоматизовану перевірку рецептур на відповідність ветеринарним та нормативним стандартам дозволяє уникнути помилок [8], [10];
- гнучка звітність: програмне забезпечення Formulator надає можливість створення налаштовуваних звітів для різних цілей – від контролю якості до маркетингової аналітики [8], [10];
- аналітика ринку: вбудовані засоби програмного забезпечення Formulator для аналізу ринкових тенденцій допомагають адаптувати рецептури до змін у споживчих уподобаннях [8], [10];
- підтримка користувачів: програмне забезпечення Formulator підтримує навчання, адаптацію під бізнес-потреби та регулярні оновлення, що відповідають сучасним вимогам галузі [8], [10];
- інтуїтивний інтерфейс: простота навігації у програмному забезпеченні Formulator дозволяє ефективно працювати з великою кількістю рецептів без спеціальної підготовки [8], [10];

- оптимізація складу: програмне забезпечення Formulator використовує алгоритми для підбору інгредієнтів, які забезпечують баланс між поживною цінністю, вартістю та якістю [8], [10];

- модульна архітектура: структура програмного забезпечення Formulator дозволяє розробникам адаптувати функціонал під конкретні потреби виробника [8], [10];

- стратегічна підтримка бренду: програмне забезпечення Formulator допомагає створювати рецепти, які відповідають сучасним запитам ринку та підвищують конкурентоспроможність [8], [10].

Серед недоліків програмного забезпечення Formulator можна відзначити складність початкового освоєння для нових користувачів без досвіду у формулюванні кормів або роботі з професійними аналітичними інструментами. Інтерфейс, попри візуальну зручність, потребує певної адаптації, особливо якщо йдеться про використання розширених функцій оптимізації та налаштувань. Крім того, робота з великими обсягами рецептур або інгредієнтів може вимагати значних обчислювальних ресурсів, що іноді впливає на продуктивність у слабших середовищах. Ще одним обмеженням є орієнтація на великі підприємства, що може ускладнювати доступ до всього функціоналу для невеликих компаній чи індивідуальних виробників, які шукають простіші або бюджетні рішення [8], [10].

Програмне забезпечення Pet Wellness World (рис. 1.3) позиціонується як онлайн інструмент для планування раціону та підтримки загального благополуччя домашніх тварин. Її призначення полягає в наданні користувачам можливості створити збалансований графік годування, обирати підходящі корми та аналізувати вплив харчування на здоров'я улюбленців. Завдяки інтеграції експертного контенту про харчові склади, фізіологічні потреби та загальні принципи догляду, платформа допомагає власникам приймати виважені рішення щодо дієт [7], [11].

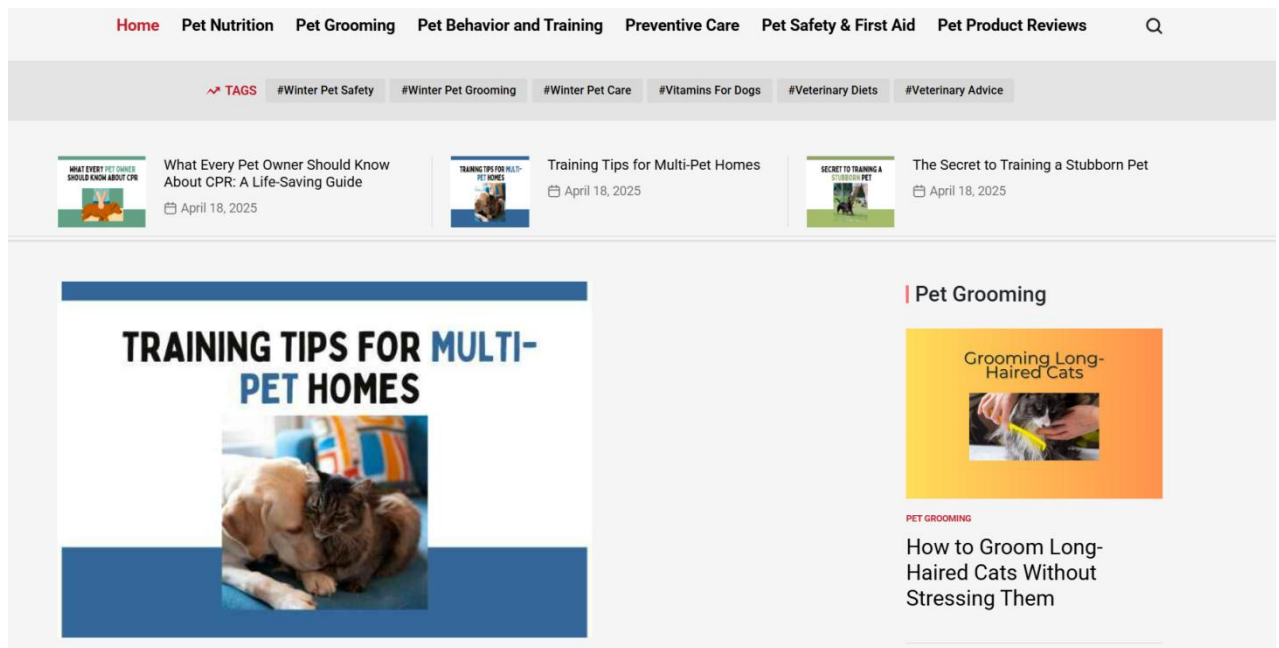


Рисунок 1.3 – Програмне забезпечення Pet Wellness World [11]

Інтерфейс програмного забезпечення Pet Wellness World побудований таким чином, щоб забезпечувати простий доступ до блоків, присвячених плануванню харчування, аналізу стану тварини та рекомендаціям щодо оптимізації калорійності або складу раціону. Використання API дозволяє оперативно отримувати дані про корми, їх поживну цінність, оцінки та рейтинги, що підтримує процес вибору. Платформа передбачає налаштування інформаційних профілів для різних тварин, що дає змогу керувати харчуванням кількох улюбленців в одному акаунті [7], [11].

Програмна реалізація спирається на вебархітектуру з фронтом, який забезпечує візуальне відображення раціонів, діаграм з вагою або активністю, а також адаптивний дизайн для комфортного використання як на десктопі, так і на мобільних пристроях. Бекенд взаємодіє з базою даних, де зберігаються рецепти, профілі тварин і закупівельна інформація, а також виконує аналітичні алгоритми для пошуку оптимального балансу харчових елементів [7], [11].

У результаті програмне забезпечення Pet Wellness World створює середовище, яке поєднує функції онлайн консультанта з харчування, інструменту для відстеження фізіологічних показників і каталогу кормів. Це робить платформу корисною для тих, хто прагне забезпечити своєму

улюбленцю здоровий раціон, методом, який базується на даних і рекомендаціях, а не інтуїтивних припущеннях [7], [11].

Програмне забезпечення Pet Wellness World має такі особливості [7], [11]:

- персоналізоване харчування: програмне забезпечення Pet Wellness World надає можливість налаштування індивідуального раціону для кожної тварини з урахуванням її віку, ваги, активності та стану здоров'я [7], [11];

- інтеграція з базами кормів: програмне забезпечення Pet Wellness World надає доступ до великої кількості готових кормів з описом складу та поживної цінності для об'єктивного вибору продукту [7], [11];

- інтерфейс з візуалізацією: програмне забезпечення Pet Wellness World забезпечує відображення даних у вигляді графіків, таблиць і схем дозволяє швидко оцінювати вплив харчування на фізіологічні показники [7], [11];

- багатопрофільне управління: програмне забезпечення Pet Wellness World надає можливість вести одночасно декілька профілів домашніх тварин в межах одного облікового запису [7], [11];

- підтримка рекомендацій: програмне забезпечення Pet Wellness World надає поради на основі сучасних ветеринарних стандартів та базується на актуальних наукових джерелах [7], [11];

- адаптивний дизайн: підтримка доступу з мобільних пристроїв і ПК для зручної роботи в будь-яких умовах користування [7], [11];

- хмарне збереження даних: інформація про тварин, рецепти та історія годування зберігається на сервері програмного забезпечення Pet Wellness World надає з можливістю швидкого відновлення [7], [11];

- зручна аналітика: програмне забезпечення Pet Wellness World підтримує автоматичний аналіз ефективності раціону та контроль змін у фізичному стані тварини з часом [7], [11];

- зворотний зв'язок із фахівцями: програмне забезпечення Pet Wellness World надає можливість отримати консультації або коментарі від експертів у сфері ветеринарного харчування [7], [11];

– орієнтація на добробут тварини: програмне забезпечення Pet Wellness World налаштоване так, щоб сприяти здоровому та збалансованому догляду за улюбленцем [7], [11].

Серед недоліків програмного забезпечення Pet Wellness World можна відзначити обмежену функціональність у безкоштовній версії, що зменшує доступ до деяких важливих інструментів та даних. Інколи спостерігається недостатня гнучкість у врахуванні нестандартних потреб тварин, наприклад, при складних дієтах або специфічних медичних обмеженнях. Деяким користувачам може здатися перевантаженим інтерфейс, особливо на етапі первинного налаштування. Через вузьку спеціалізацію на здоровому харчуванні платформа не завжди охоплює інші аспекти догляду за тваринами, що змушує користувачів паралельно шукати додаткові інструменти [7], [11].

Порівняльну характеристику програмного забезпечення для організації харчування домашніх тварин наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для організації харчування домашніх тварин

Критерій порівняння	Pet Food Pro [9]	Formulator [10]	Pet Wellness World [11]
Засоби аналітики ринку	—	+	+—
Візуалізація поживних показників	+—	+	+
Простота інтерфейсу	+—	+—	+
Можливість інтеграції з базами кормів	+—	+	+—
Робота з профілями кількох тварин	—	+—	+
Доступність для звичайних власників тварин	—	+—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для організації харчування домашніх тварин. Проте деякі програмні засоби для організації харчування домашніх тварин можуть мати інтерфейс, який є надмірно складним або перевантаженим, що ускладнює їх використання для людей похилого віку або користувачів без технічної підготовки. Інколи користувачі стикаються з платними функціями або нав'язливими покупками в додатку, що знижує рівень довіри до програмного продукту. Технічні обмеження також відіграють роль. Деяке програмне забезпечення для організації харчування домашніх тварин потребує постійного доступу до Інтернету, синхронізації з розумними пристроями або оновлень, що не завжди зручно для користувача. Крім того, в разі збоїв у роботі чи відсутності доступу до сервера функціональність додатку може тимчасово бути втрачена. Ще одним недоліком є обмежена адаптація під різні види та породи тварин. Тому актуальною є розробка програмного забезпечення для організації харчування домашніх тварин.

При розробці програмного забезпечення для організації харчування домашніх тварин необхідно забезпечити такі функціональні вимоги:

- можливість подання інформації про продукти харчування для тварин та їх ціни;
- підтримка можливості авторизації та реєстрації у застосунку для організації харчування домашніх тварин;
- можливість замовлення обраних товарів харчування для тварин;
- підтримка можливості подання інформації про рецепти страв для тварин;
- можливість додавання та редагування інформації про домашніх улюбленців користувача;
- підтримка можливості додавання або видалення інформації про продукти харчування для тварин (функція адміністратора);
- можливість додавання інформації про рецепти (функція адміністратора).

1.3 Висновки за розділом 1

Визначено, що програмне забезпечення для організації харчування домашніх тварин являє собою спеціалізовану систему, що забезпечує автоматизоване планування, моніторинг і оптимізацію раціону тварин залежно від їхніх характеристик. Його головною метою є покращення якості догляду за домашніми улюбленцями шляхом.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для організації харчування домашніх тварин. Проте деякі програмні засоби для організації харчування домашніх тварин можуть мати інтерфейс, який є надмірно складним або перевантаженим, що ускладнює їх використання для людей похилого віку або користувачів без технічної підготовки. Інколи користувачі стикаються з платними функціями або нав'язливими покупками в додатку, що знижує рівень довіри до програмного продукту. Технічні обмеження також відіграють роль. Деяке програмне забезпечення для організації харчування домашніх тварин потребує постійного доступу до Інтернету, синхронізації з розумними пристроями або оновлень, що не завжди зручно для користувача. Крім того, в разі збоїв у роботі чи відсутності доступу до сервера функціональність додатку може тимчасово бути втрачена. Ще одним недоліком є обмежена адаптація під різні види та породи тварин. Тому актуальною є розробка програмного забезпечення для організації харчування домашніх тварин.

Сформульовано функціональні вимоги до програмного забезпечення для організації харчування домашніх тварин.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для організації харчування домашніх тварин було використано мову програмування C# [12], [13].

Мову C# було створено в межах платформи .NET із метою розробки якісного програмного забезпечення, здатного працювати стабільно та безпечно в умовах складних і масштабованих систем. Вона вирізняється структурованістю, суворою типізацією та зрозумілим синтаксисом, що робить її універсальним інструментом як для проєктів початкового рівня, так і для великих комерційних продуктів. Завдяки стабільному розвитку екосистеми, функціональні можливості мови постійно зростають, що дозволяє ефективно впроваджувати нові технології та підтримувати сучасні підходи до розробки [12], [13].

C# особливо добре себе зарекомендувала в створенні застосунків для настільних систем, вебплатформ і мобільних пристроїв, а також у галузях, де потрібна взаємодія з хмарними інфраструктурами, базами даних або стороннім обладнанням. Автоматичне керування пам'яттю, контроль типів під час компіляції та механізми обробки винятків дозволяють створювати програмне забезпечення з високим рівнем надійності. Робота з інтерфейсами, візуальними компонентами та API не потребує надмірних зусиль завдяки великій кількості доступних бібліотек [12], [13].

Особливо актуальним є використання C# у випадках, коли програму необхідно розгорнути на різних платформах без дублювання коду. Це стало можливим завдяки технології, яка забезпечує створення мобільних застосунків. Такий підхід дозволяє одночасно охоплювати широку аудиторію користувачів та зменшувати витрати на розробку. Також реалізовано підтримку інтеграції з сервісами віддаленого зберігання й обміну даними, що відкриває широкі можливості для синхронізації інформації, зокрема про тварин, їхній стан або індивідуальні параметри годування [12], [13].

Завдяки своїй модульності, C# дозволяє легко масштабувати програму та додавати до неї нові компоненти, зокрема алгоритми з використанням штучного інтелекту чи взаємодію з «розумними» пристроями. Наявність повнофункціонального середовища розробки Visual Studio забезпечує комфортне середовище для створення застосунків будь-якої складності — від проєктування до налагодження, тестування та випуску. Visual Studio пропонує інструменти, які значно скорочують час розробки та знижують ризики помилок [12], [13].

Таким чином, C# є доцільним вибором для реалізації програмного рішення в сфері харчування домашніх тварин, оскільки дозволяє розробити зручний, продуктивний, стабільний і легко розширюваний застосунок, здатний працювати на різних платформах та взаємодіяти з іншими сервісами й пристроями [12], [13].

У сучасному програмуванні мова C# вирізняється розгалуженим середовищем, у якому зібрано безліч готових рішень для реалізації різних задач. Завдяки доступності фреймворків, модулів і відкритих бібліотек немає потреби витрачати час на створення базових компонентів, оскільки багато рішень уже готові до використання. Це значно пришвидшує цикл розробки й полегшує впровадження складної логіки. Підтримка з боку великої кількості фахівців, численних форумів і офіційних матеріалів дозволяє швидко знаходити відповіді на практичні питання та швидко освоювати інструменти, навіть за умов обмеженого досвіду [12], [13].

C# давно здобула репутацію мови, що забезпечує високу стабільність роботи програмних продуктів і придатна для створення серйозних рішень, які повинні працювати на різних платформах і забезпечувати тривалу підтримку. При її використанні в розробці застосунків для організації харчування домашніх тварин відкриваються широкі можливості для створення універсального функціоналу, здатного працювати в локальному або мережевому середовищі, з підтримкою сучасної графіки та інтерактивності [12], [13].

Особливої уваги заслуговує зв'язок мови з екосистемою .NET, що відкриває доступ до потужних інструментів для роботи з базами даних, обробки запитів, побудови графічних інтерфейсів та організації обміну даними. За допомогою .NET MAUI, який дає змогу писати код один раз і використовувати його як для Android, так і для iOS, значно полегшується створення мобільних застосунків. Уникнення дублювання розробки під кожен платформу дозволяє суттєво зекономити ресурси та прискорити вихід продукту на ринок [12], [13].

Додатково підтримується зручна інтеграція з хмарними сервісами, що забезпечує безпечне зберігання інформації, централізовану синхронізацію, а також можливість організації резервного копіювання та взаємодії між різними користувачами системи. Це дозволяє реалізовувати віддалене управління процесами годування та моніторинг у режимі реального часу [12], [13].

Архітектурна гнучкість мови програмування C# забезпечує ефективне масштабування програм, а також подальше доповнення функцій, таких як аналітика на основі штучного інтелекту, підтримка «розумних» пристроїв або підключення датчиків для відстеження поведінки тварини. Весь цикл створення програмного забезпечення можна вести в середовищі Visual Studio, яке надає широкі можливості не лише для написання коду, а й для його тестування, налагодження та аналізу продуктивності. Робота в такому середовищі відбувається послідовно, з високим ступенем автоматизації й зручності [12], [13].

З огляду на поєднання технологічної зрілості мови, її універсальності, а також підтримки з боку сучасних інструментів розробки, використання мови програмування C# у подібних проєктах вважається стратегічно виправданим і вигідним рішенням [12], [13].

Обґрунтування вибору мови програмування для реалізації ПЗ для організації харчування домашніх тварин наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для реалізації програмного забезпечення для організації харчування домашніх тварин

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Кросплатформна підтримка	+	+	+—
Продуктивність виконання коду	+	+—	+
Зручність розробки графічного інтерфейсу	+	—	+—
Зручність для створення ПЗ для організації харчування домашніх тварин	+	—	—
Можливість використання з розумними пристроями	+	+—	+—

Отже, для реалізації програмного забезпечення для організації харчування домашніх тварин обрано мову програмування C#, яка має зручний синтаксис і високу продуктивність при розробці десктопних, мобільних та вебзастосунків. Крім того, забезпечується широкий набір бібліотек і інструментів, що значно пришвидшують розробку функціоналу, пов'язаного з роботою з базами даних, мережею, графічним інтерфейсом та мультимедійними компонентами.

2.2 Вибір середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин

В якості середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин було обрано середовище Visual Studio [14], [15].

Середовище розробки Visual Studio розглядається як потужна професійна платформа для створення програмного забезпечення, що здатна охопити широкий спектр завдань – від проектування інтерфейсу до реалізації складної логіки застосунків. Її технічні можливості формуються на основі постійної підтримки з боку Microsoft, що забезпечує швидку адаптацію до нових технологічних стандартів, а також стабільну інтеграцію з сучасними рішеннями, включно з хмарними платформами, середовищами та мобільними інфраструктурами [14], [15].

Цей інструмент відзначається тісним поєднанням із мовою програмування C# і середовищем .NET, що дозволяє реалізовувати функціональність додатків швидко й ефективно. Архітектура програм будується на зрозумілих принципах, а побудова інтерфейсів та взаємодія з даними не вимагають надлишкових зусиль завдяки наявності зручних візуальних рішень. Також доступна можливість командної взаємодії в межах великих проєктів, зокрема через вбудовані механізми контролю версій та підтримку DevOps-сервісів, що особливо актуально для довготривалих і складних продуктів [14], [15].

Широкий набір додаткових розширень дозволяє адаптувати середовище до будь-яких потреб проєкту. Розробка стає більш гнучкою та точною, оскільки можливо підключати сторонні інструменти, розширювати аналітичні можливості, забезпечувати контроль над продуктивністю й реалізовувати високий рівень безпеки без зміни основної логіки. Тестування в межах цього середовища охоплює всі рівні – від перевірки окремих функцій до оцінювання цілісної взаємодії елементів, включно з автоматизованою перевіркою роботи інтерфейсів [14], [15].

Також платформа підтримує розробку застосунків, що працюють у різних середовищах, через використання емуляторів і хмарних рішень для моделювання поведінки на різних пристроях. Це значно полегшує перевірку функціоналу ще до етапу фактичного розгортання, що критично важливо при створенні продуктів із мультиплатформним доступом [14], [15].

Використання Visual Studio як основного середовища для створення програмного забезпечення, призначеного для управління харчуванням домашніх тварин, обумовлюється його технологічною універсальністю та гнучкістю. Завдяки тісній інтеграції з мовою програмування C# і екосистемою .NET відкривається доступ до всього необхідного функціоналу для проєктування застосунків будь-якого типу: від класичних десктопних до мобільних чи веборієнтованих [14], [15].

Ця платформа надає розробнику усі важливі засоби в єдиному інтерфейсі, що дозволяє зручно працювати з графічними елементами, структурою коду, підключеними бібліотеками й інструментами версіювання, не перемикаючись між окремими середовищами чи сервісами. Усе побудовано так, щоб процес створення додатку був цілісним і логічним, із мінімальними перешкодами та технічними обмеженнями. Значно полегшується розробка складних програмних рішень, оскільки забезпечено автоматизовану допомогу в написанні коду, виявленні помилок і оптимізації функціоналу ще до етапу компіляції [14], [15].

Платформа дозволяє розробляти мобільні програми для кількох операційних систем без необхідності дублювати логіку в різних середовищах. Завдяки підтримці .NET MAUI можливо реалізувати один кодовий базис і запускати його як на Android, так і на iOS, що дозволяє зосередитися на розширенні функціоналу замість технічної адаптації до різних платформ. У випадку програм, де передбачено дистанційне управління пристроями або синхронізацію даних у реальному часі, доступ до сервісів Microsoft Azure відкриває широкі можливості для масштабованого зберігання, передавання інформації та моніторингу [14], [15].

Особливістю Visual Studio є також глибока підтримка тестування та аналізу роботи застосунку: кожен компонент може бути перевірений у процесі розробки, що дозволяє виявляти та виправляти помилки на ранніх етапах. Це важливо у проєктах, що містять кілька функціональних модулів, взаємопов'язаних через внутрішню логіку або зовнішні сервіси. Підтримка

продуктивності, оптимізації й рефакторингу вбудована в основні інструменти, що дає змогу зосередитися на функціональності, не витрачаючи час на технічні труднощі [14], [15].

Visual Studio користується широким визнанням серед професійної спільноти, що забезпечує наявність актуальних навчальних ресурсів, активного обміну досвідом і постійних оновлень. У контексті створення цифрового продукту, пов'язаного з харчуванням тварин, це середовище дозволяє реалізувати не лише зручний інтерфейс та стабільну роботу, а й подбати про довгострокову підтримку, розширення функцій і швидку адаптацію до змін у технологічному ландшафті [14], [15].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Повна інтеграція з .NET	+	+	+—
Автодоповнення та навігація по коду	+	+	+—
Інтеграція з хмарними сервісами	+	+—	—
Потужність інструментів для відлагодження	+	+—	+—
Можливість розширення та плагіни	+	+	+—

Таким чином, середовище розробки Visual Studio слугує не просто місцем для написання коду, а є цілісною системою, яка супроводжує створення програмного забезпечення від першої ідеї до фінального впровадження. Завдяки поєднанню функціональності, розширюваності й сучасних інструментів, це середовище ідеально підходить для проєктів, пов'язаних із автоматизацією процесів, зокрема й у сфері управління харчуванням домашніх тварин [14], [15].

Отже, для створення програмного забезпечення для організації харчування домашніх тварин обрано середовище розробки Visual Studio, яке надає потужну підтримку мови C# та платформи .NET, що дозволяє створювати стабільне, надійне й масштабоване програмне забезпечення з використанням найкращих практик розробки. Інша важлива перевага полягає в інтелектуальній підтримці коду: автодоповнення, підсвітка синтаксису, рефакторинг, попередження про помилки ще до компіляції — усе це знижує кількість помилок і підвищує якість програмного продукту. Також доступні вбудовані засоби тестування, профілювання продуктивності та налагодження, що особливо корисно при створенні складних застосунків із багатьма модулями.

2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин

З метою можливості розробки програмного забезпечення для організації харчування домашніх тварин створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин враховуємо те, що користувачі програми мають можливості пошуку харчових товарів, додавання товару к кошик, створення замовлення харчових товарів для тварин, пошуку рецепту та авторизації чи реєстрації. Адміністратор програмного

забезпеченням для організації харчування домашніх тварин має можливість редагування рецептів та редагування інформації про харчові продукти для тварин.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для організації харчування домашніх тварин можна описати таким чином:

- користувач програмного забезпечення для організації харчування домашніх тварин авторизується або реєструється;
- після авторизації користувача сервер програмного забезпечення для організації харчування домашніх тварин у відповідь повертає основну сторінку вебсайту для організації харчування домашніх тварин;
- користувач бачить відповідну сторінку для організації харчування домашніх тварин та шукає потрібну харчову продукцію;
- користувач за необхідності читає опис відповідної харчової продукції або читає відповідні рецепти;
- користувач натискає кнопку для купівлі потрібної йому харчової продукції та створення замовлення;
- користувач завершує роботу з програмним забезпеченням для організації харчування домашніх тварин.

Опишемо основні дії адміністратора програмного забезпечення для організації харчування домашніх тварин:

- адміністратор програмного забезпечення для організації харчування домашніх тварин авторизується;
- після авторизації адміністратора сервер програмного забезпечення для організації харчування домашніх тварин у відповідь на запит повертає основну сторінку для організації харчування домашніх тварин;
- адміністратор програмного забезпечення для організації харчування домашніх тварин бачить відповідну сторінку та обирає певний рецепт або певну харчову продукцію для редагування;

- адміністратор програмного забезпечення для організації харчування домашніх тварин переглядає обрану інформацію та за необхідності редагує її;
- адміністратор програмного забезпечення для організації харчування домашніх тварин завершує роботу з програмою.

Враховуючи сценарії роботи користувачів програмного забезпечення для організації харчування домашніх тварин (користувач та адміністратор) при роботі з програмою, модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин подано у вигляді діаграми прецедентів, зображеної на рис. 2.1.

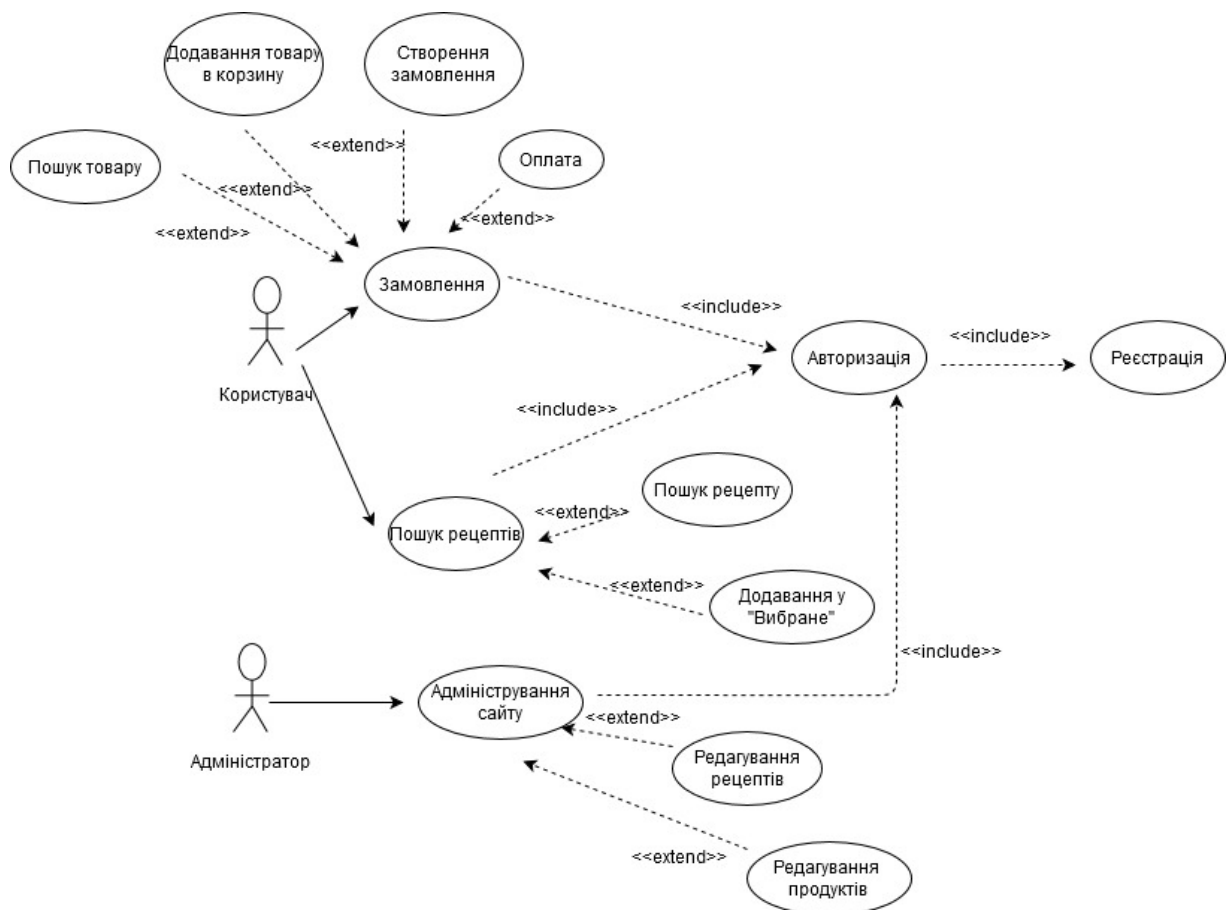


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин, яка подана у

вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для організації харчування домашніх тварин.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для організації харчування домашніх тварин обрано мову програмування C#, яка має зручний синтаксис і високу продуктивність при розробці десктопних, мобільних та вебзастосунків. Крім того, забезпечується широкий набір бібліотек і інструментів, що значно пришвидшують розробку функціоналу, пов'язаного з роботою з базами даних, мережею, графічним інтерфейсом та мультимедійними компонентами.

Для створення програмного забезпечення для організації харчування домашніх тварин обрано середовище розробки Visual Studio, яке надає потужну підтримку мови C# та платформи .NET, що дозволяє створювати стабільне, надійне й масштабоване програмне забезпечення з використанням найкращих практик розробки. Інша важлива перевага полягає в інтелектуальній підтримці коду: автодоповнення, підсвітка синтаксису, рефакторинг, попередження про помилки ще до компіляції — усе це знижує кількість помилок і підвищує якість програмного продукту. Також доступні вбудовані засоби тестування, профілювання продуктивності та налагодження, що особливо корисно при створенні складних застосунків із багатьма модулями.

Створено модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для організації харчування домашніх тварин.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для організації харчування домашніх тварин

Структуру програмного забезпечення для організації харчування домашніх тварин подамо за допомогою діаграми класів, яка наведена на рис. 3.1.

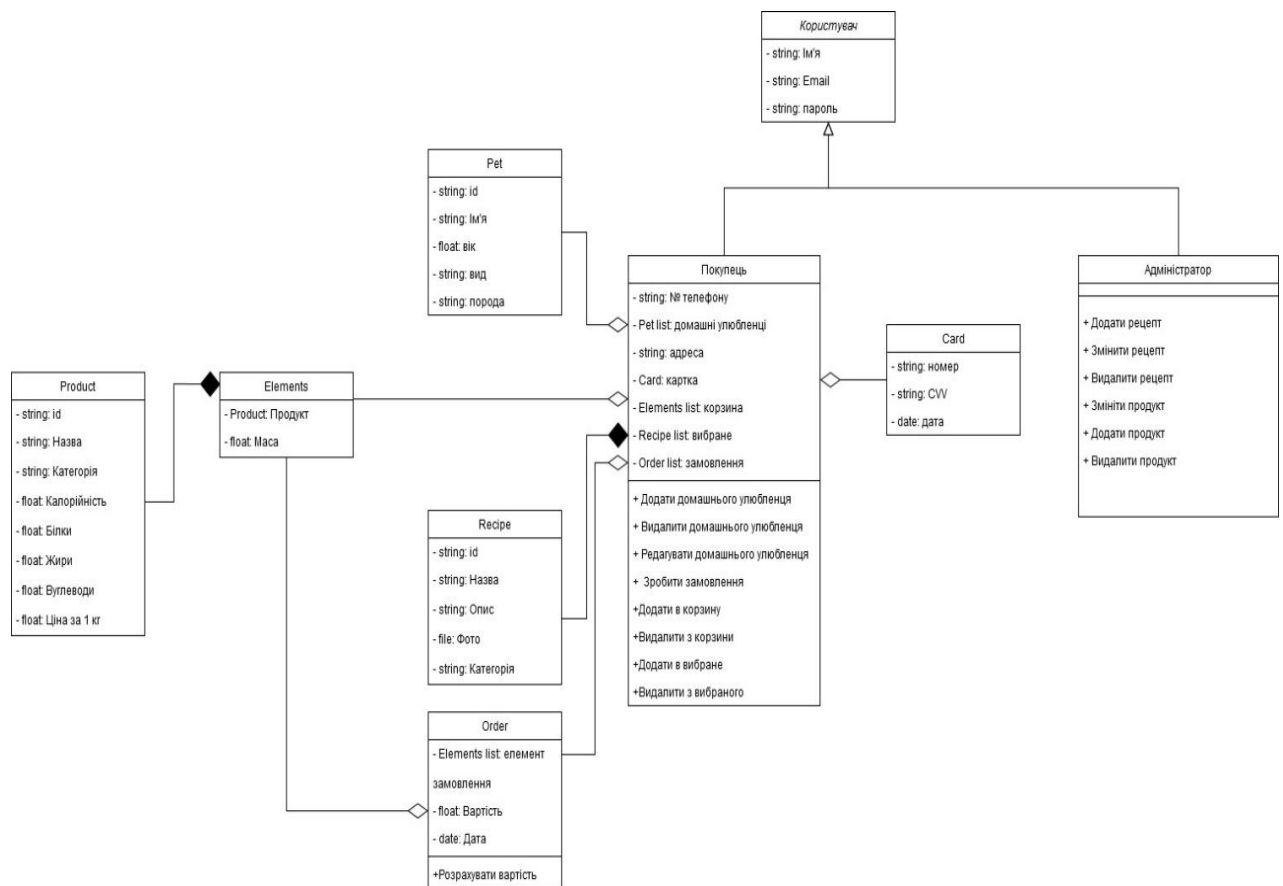


Рисунок 3.1 – Структура програмного забезпечення для організації харчування домашніх тварин

Діаграма класів програмного забезпечення для організації харчування домашніх тварин відображає логічну структуру системи, в якій кожен клас представляє певну сутність, пов'язану з основними процесами платформи. Основними класами є Користувач, Pet, Product, Покупець, Order, Адміністратор та інші.

У центрі моделі розташовується узагальнений клас Користувач, що містить загальні атрибути, такі як ім'я, електронна пошта, логін і пароль. Цей клас виступає батьківським для двох похідних — Покупець та Адміністратор. Роль першого полягає у здійсненні замовлень, формуванні раціону для домашніх улюбленців і взаємодії з каталогом продуктів. Другий відповідає за керування даними, включаючи додавання, редагування або видалення товарів, а також за перевірку та обробку замовлень.

Клас Pet описує тварину, яка належить певному користувачеві, й утримує інформацію про її ім'я, вид, вік, вагу, дієтичні потреби та стан здоров'я. Один покупець може мати кількох тварин, тому між класами Покупець і Pet встановлено зв'язок один-до-багатьох.

Продукти, що доступні для формування раціону, репрезентуються класом Product. Він зберігає дані про назву, тип їжі, поживну цінність, вагу упаковки, термін придатності, а також сумісність із певними характеристиками тварин. Товари можуть бути згруповані або відфільтровані відповідно до потреб, які визначаються на основі параметрів класу Pet.

Замовлення, оформлені користувачами, представлені класом Order, який містить посилання на конкретного покупця, список вибраних продуктів, дату створення, статус виконання та загальну суму. Передбачена також можливість відстеження історії змін кожного замовлення, що може реалізовуватися через окремий допоміжний клас, пов'язаний із Order композицією.

У межах адміністративної частини система надає можливість взаємодії з базою продуктів, відслідковування статистики замовлень, оновлення інформації про товари та перегляд активності користувачів. Адміністратор має розширені права доступу, які не доступні звичайним покупцям.

Зв'язки між класами реалізовані з урахуванням принципів інкапсуляції, спадкування та композиції. Кожен об'єкт створюється у відповідності до своєї ролі в бізнес-логіці: тварина завжди належить користувачу, замовлення

пов'язане з продуктами та покупцем, а дії адміністратора впливають на всю систему загалом.

У результаті ця діаграма класів відображає структурну організацію програмного забезпечення, що дозволяє ефективно керувати процесами, пов'язаними з харчуванням домашніх тварин, враховуючи як індивідуальні потреби тварин, так і загальні функції взаємодії між користувачами й системою.

На рис. 3.2 наведено діаграму компонентів програмного забезпечення для організації харчування домашніх тварин

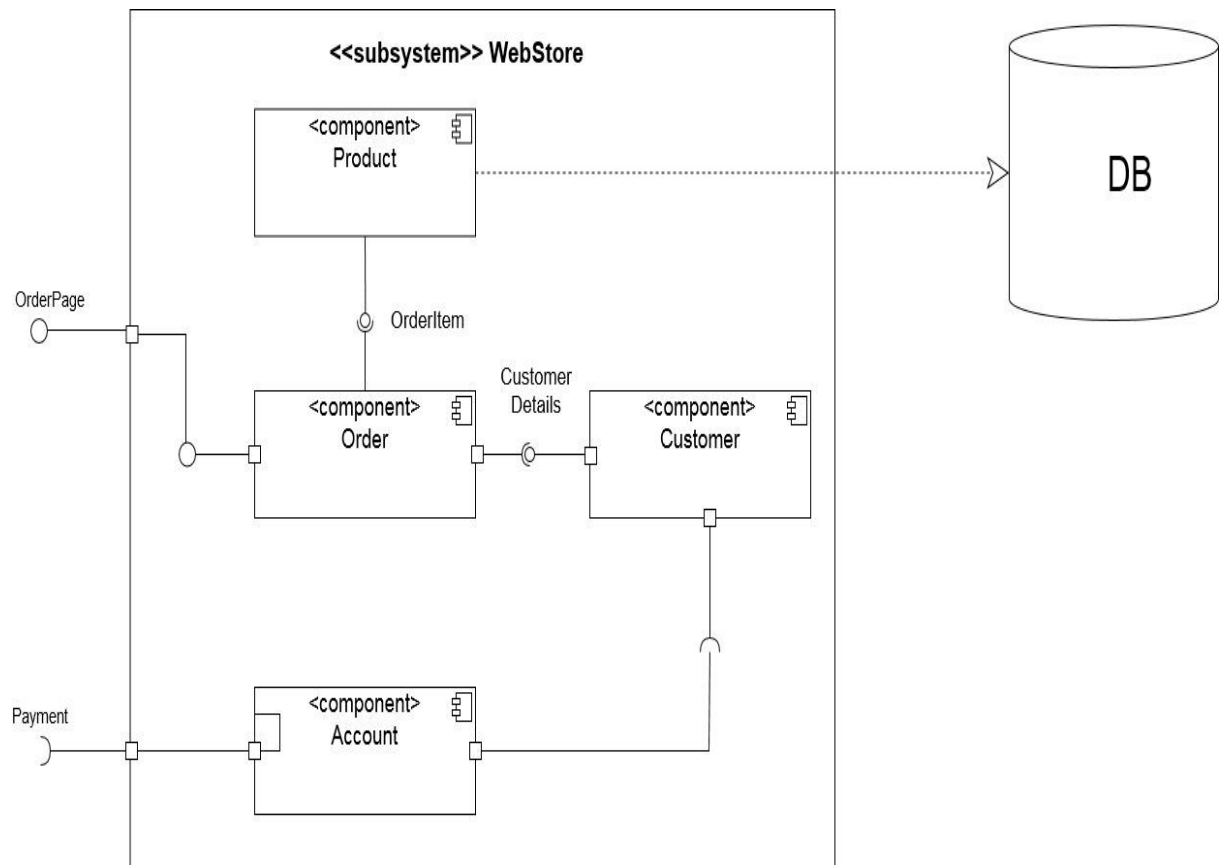


Рисунок 3.2 – Діаграма компонентів програмного забезпечення для організації харчування домашніх тварин

Діаграма компонентів, яка моделює архітектуру програмного забезпечення для організації харчування домашніх тварин, відображає взаємодію основних функціональних частин системи та їхнє підключення до

зовнішніх і внутрішніх ресурсів. Кожен компонент відповідає за окрему частину бізнес-логіки та взаємодіє з іншими через чітко визначені інтерфейси.

Компонент Product реалізує логіку обробки товарів, доступних для замовлення. Він є залежним від бази даних, що виконує роль централізованого сховища даних. Ця залежність свідчить про потребу компонента Product регулярно взаємодіяти з базою даних для отримання інформації про доступні позиції, їхні характеристики, наявність та оновлення. Сам компонент не функціонує автономно, оскільки його цілісність базується на постійному доступі до актуальних даних у сховищі.

Компонент Order відповідає за створення й обробку замовлень. Він надає зовнішній інтерфейс під назвою OrderPage, який служить точкою входу для взаємодії користувача зі сторінкою замовлення. Через цей інтерфейс реалізується функція вибору продуктів і підтвердження покупки. Замовлення формується на основі інформації, яка надходить як з компонента Product, так і з даних, що вводить користувач через Customer Details.

Компонент Customer виконує роль ініціатора дій у системі. Його структура включає обробку персональних даних, логіки взаємодії з формами введення, перевірки валідності інформації, а також передачі необхідних параметрів до замовлення. Цей компонент логічно пов'язаний з Account, оскільки кожен користувач у процесі оформлення замовлення має взаємодіяти зі своїм платіжним рахунком або гаманцем. Account включає механізми оплати, перевірки балансу, підтвердження транзакцій та зберігання фінансових реквізитів.

Уся система функціонує на основі компонентної архітектури, де кожен елемент має чітко окреслену відповідальність і взаємодіє з іншими виключно через визначені точки доступу. Така структура дозволяє спростити обслуговування коду, полегшує масштабування функціоналу, а також забезпечує високу гнучкість при модифікаціях. Діаграма також демонструє важливість ролі бази даних як фундаментального ресурсу, від якого залежать інші компоненти, а також відображає, як користувач, через набір пов'язаних

функцій, здійснює повний цикл дій — від перегляду продукції до оплати замовлення.

3.2 Функціонування програмного забезпечення для організації харчування домашніх тварин

Функціонування програмного забезпечення для організації харчування домашніх тварин подамо за допомогою діаграми діяльності, зображеної на рис. 3.3.

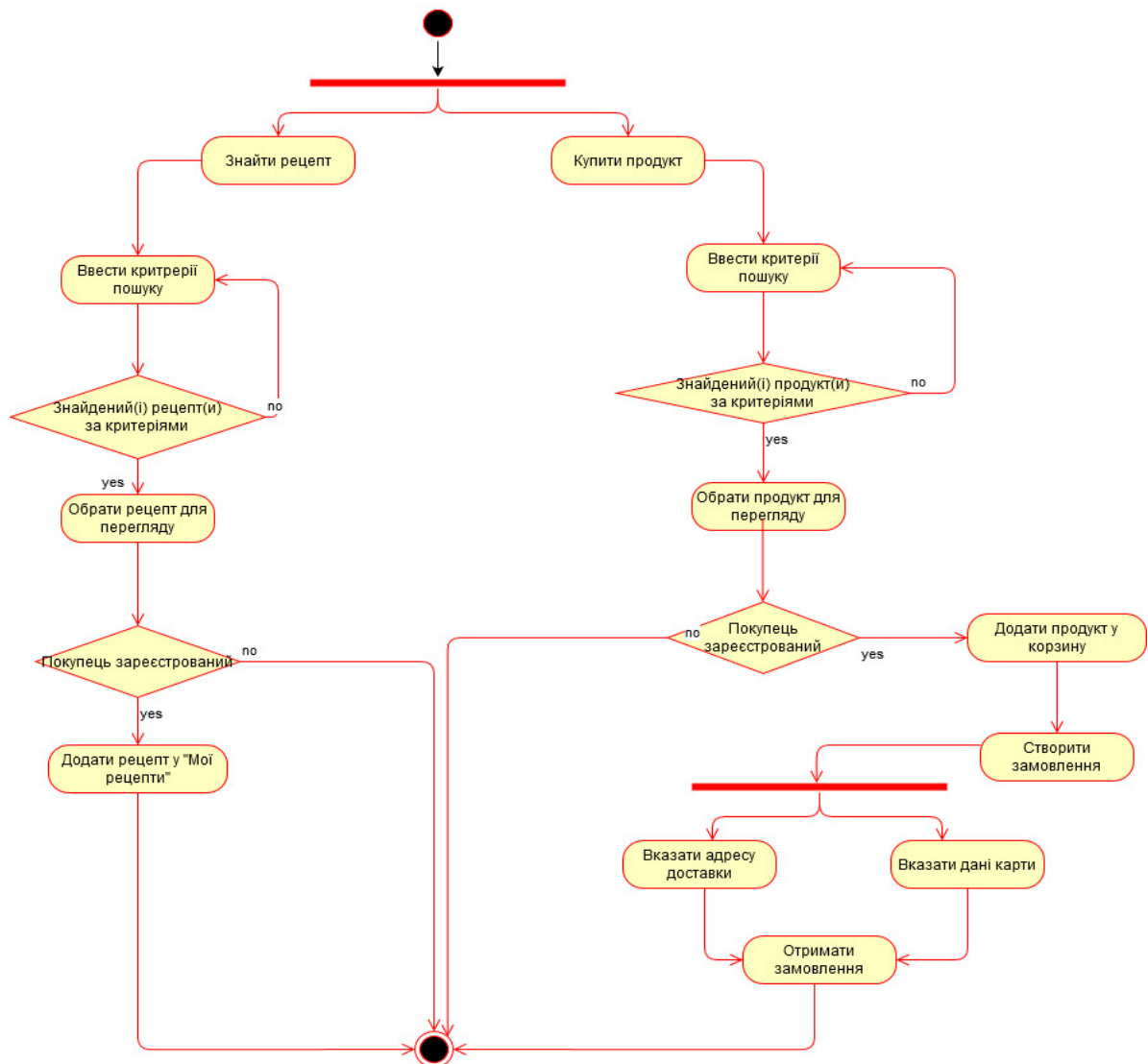


Рисунок 3.3 – Схема функціонування у вигляді діаграми діяльності програмного забезпечення для організації харчування домашніх тварин

Діаграма діяльності, яка описує функціонування програмного забезпечення для організації харчування домашніх тварин, демонструє дві альтернативні лінії поведінки користувача: пошук рецепту для годування улюбленця та процес придбання готового продукту. Обидва потоки починаються зі спільної початкової дії — входу до системи або відкриття головного інтерфейсу застосунку, де користувач обирає один із доступних сценаріїв.

У першому варіанті дій користувач вирішує знайти відповідний рецепт. З цього моменту починається логічна гілка, яка веде через крок визначення параметрів тварини: вибирається вид, вага, вік, харчові обмеження або алергії. Далі система ініціює запит до модуля рекомендацій, що аналізує введені параметри й формує список доступних рецептів. Користувач може ознайомитися з переліком, переглянути інгредієнти, енергетичну цінність та інструкції з приготування. На завершення цього сценарію можливе або збереження рецепту в особистому кабінеті, або його друк, або перехід до дій з пошуку відповідних товарів для реалізації рецепту.

Інший шлях діяльності починається з вибору функції «купити продукт». У цьому випадку користувач переходить до каталогу продукції, де можливо застосувати фільтри, що базуються на типі тварини, харчових характеристиках, бренді або ціновому діапазоні. Після цього здійснюється вибір товару, його перегляд та додавання до кошика. Далі система пропонує користувачу перевірити або змінити інформацію про доставку й оплату. Завершується ця діяльність оформленням замовлення та підтвердженням транзакції. У відповідь користувач отримує повідомлення про успішну покупку й інформацію щодо очікуваної доставки.

Діаграма діяльності побудована з урахуванням взаємозалежностей між діями, включаючи умови переходів і точки прийняття рішень. Під час реалізації обох сценаріїв може бути передбачене логування дій користувача, збереження історії запитів та взаємодія з іншими підсистемами, такими як база даних або платіжний сервіс. Таким чином, ця діаграма не лише ілюструє

логіку роботи користувача з інтерфейсом, а й демонструє послідовність, в якій відбувається реалізація ключових функцій застосунку.

На рис. 3.4 наведено діаграму станів програмного забезпечення для організації харчування домашніх тварин

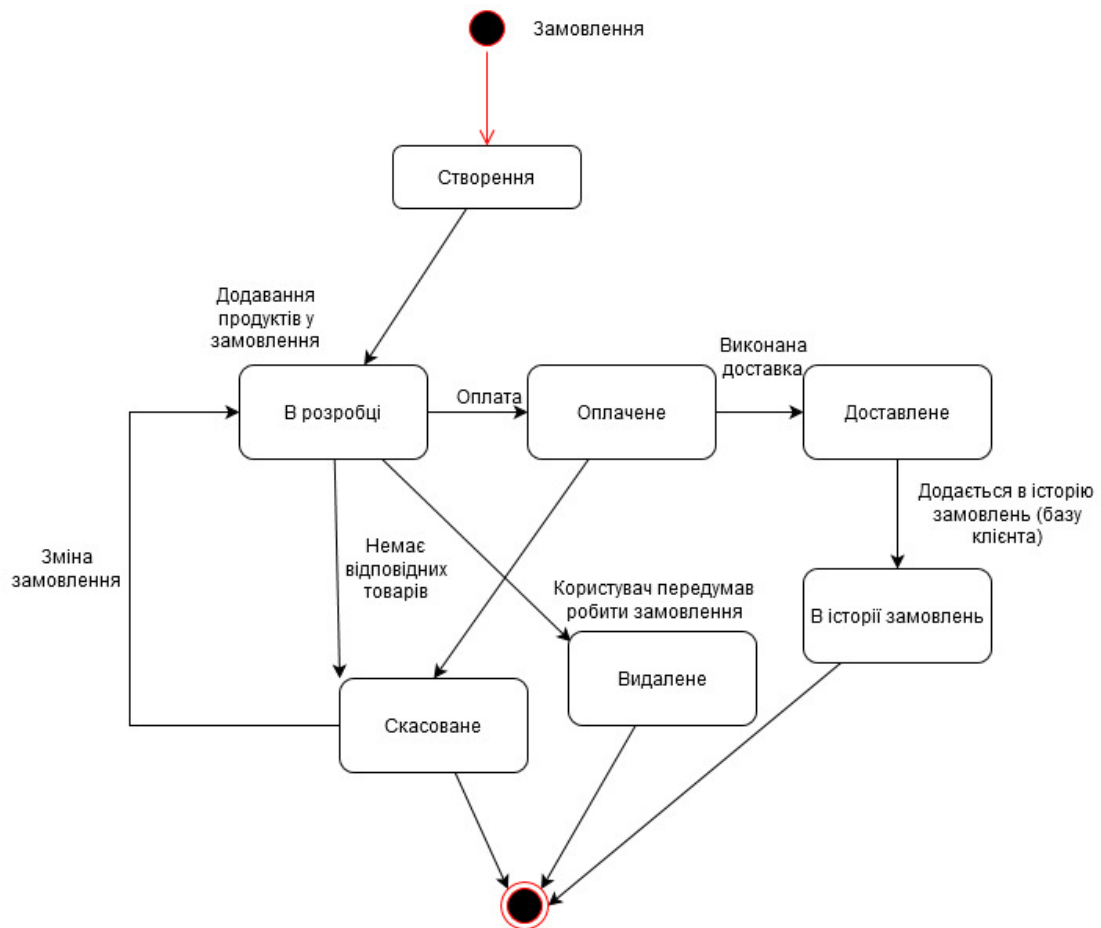


Рисунок 3.4 – Діаграма станів програмного забезпечення для організації харчування домашніх тварин

Діаграма станів програмного забезпечення, призначеного для організації харчування домашніх тварин, відображає зміну внутрішнього стану системи в залежності від дій користувача та відповідей підсистем. Цей тип моделі дозволяє зрозуміти, як переходить програмне забезпечення з одного логічного етапу роботи до іншого під час експлуатації.

На початку життєвого циклу застосунку система перебуває у стані очікування взаємодії, де вона готова до запуску, авторизації або реєстрації

нового користувача. Після цього, в залежності від ролі користувача, активується режим ініціалізації інтерфейсу — програма переходить у стан основного робочого середовища, де доступна базова навігація.

Подальші переходи між станами залежать від типу обраної дії. Наприклад, у разі здійснення дій, пов'язаних із замовленням товарів, система переходить у послідовність станів, пов'язаних з наповненням кошика, перевіркою деталей доставки, оплатою та підтвердженням. Кожен із цих етапів розглядається як окремий стан, після якого можливий як перехід до наступного, так і повернення назад — наприклад, у випадку помилки або внесення змін.

На рис. 3.5 наведено діаграму кооперації програмного забезпечення для організації харчування домашніх тварин

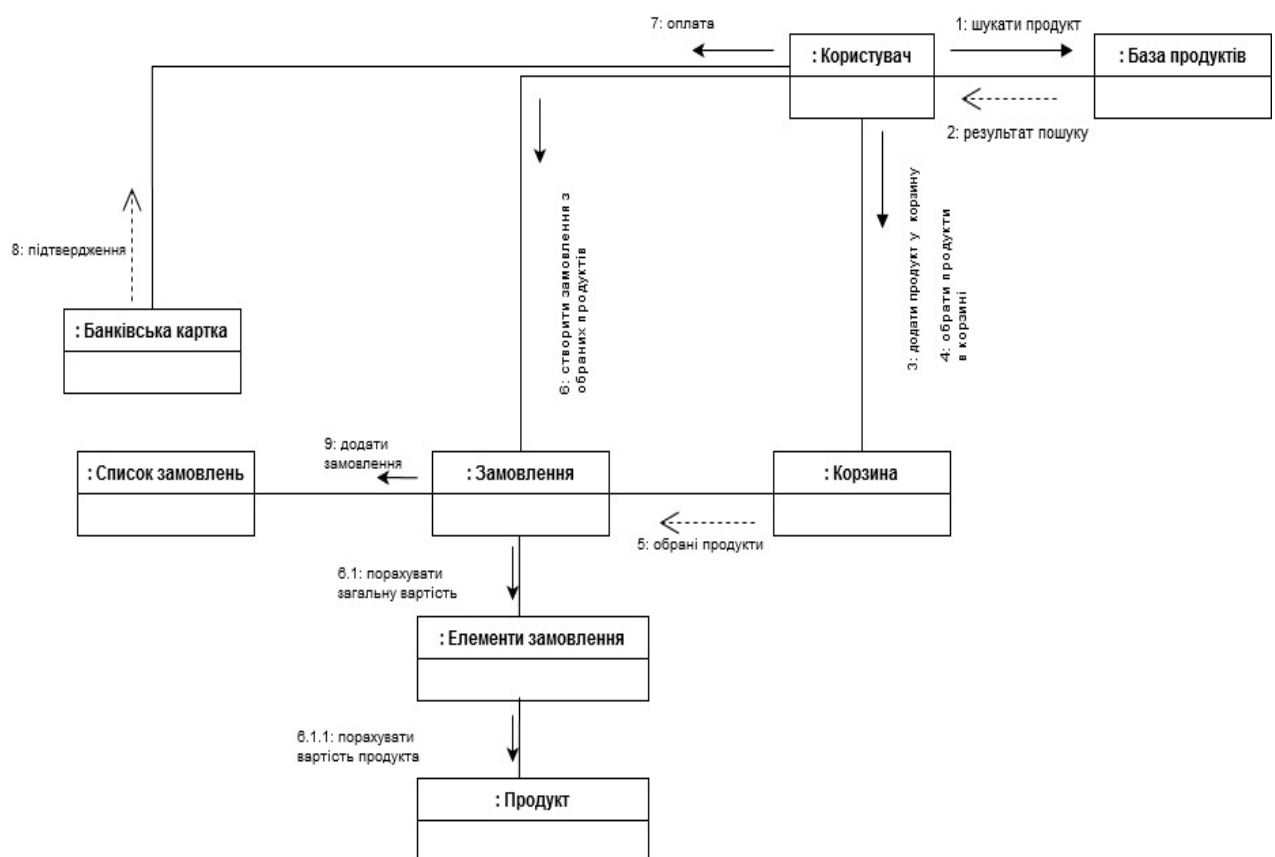


Рисунок 3.5 – Діаграма кооперації програмного забезпечення для організації харчування домашніх тварин

обробку й ефективну взаємодію всіх даних, пов'язаних з користувачами, їхніми тваринами, продуктами харчування та замовленнями. В її основі — сутності, які логічно відображають предметну область та пов'язані між собою через ключові залежності.

Центральне місце займає сутність `client`, яка містить основну інформацію про користувача: ім'я, контактні дані, облікову інформацію, історію дій. Кожен клієнт пов'язаний з однією або кількома тваринами, що моделюється через сутність `pet`. У ній зберігаються деталі про домашнього улюбленця: вид, порода, вік, вагові характеристики, особливості дієти або стан здоров'я. Між `client` і `pet` реалізовано зв'язок «один до багатьох».

Для забезпечення функції підбору та замовлення корму використовується сутність `products`, яка охоплює інформацію про харчові товари: назву, склад, енергетичну цінність, призначення, вагу упаковки, виробника та вартість. Кожен товар віднесений до певної категорії, що представлено через сутність `categoryProducts`. Категорії дозволяють структурувати асортимент за типом (наприклад, сухий корм, вологий корм, ласощі, дієтичне харчування тощо) і полегшити навігацію під час вибору.

Сутність `shoppingCart` виконує роль тимчасового сховища обраних товарів. Вона пов'язана з клієнтом і містить список продуктів, які користувач додав до кошика до моменту оформлення замовлення. При переході до оплати формується запис у таблиці `orders`, де фіксуються всі ключові параметри замовлення: клієнт, дата, перелік продуктів, загальна сума, спосіб оплати, статус виконання, а також зв'язок із `shoppingCart` для відображення того, що саме було замовлено.

Крім основних сутностей, база даних включає допоміжні для збереження інформації про транзакції, адресної доставки, відстеження взаємодії з системою.

Отже база даних побудована з урахуванням нормалізації даних для зменшення надмірності, забезпечення цілісності та підтримки ефективних зв'язків між таблицями. Завдяки цьому база даних надійно підтримує

функціонування програмного забезпечення, дозволяючи швидко обробляти запити, масштабувати функціональність та реалізовувати індивідуальні потреби користувачів і тварин.

3.4 Проєктування інтерфейсу програмного забезпечення для організації харчування домашніх тварин

Проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин виконано з урахуванням вимог технічного завдання. При розробленні програмного забезпечення для організації харчування домашніх тварин враховані функціональні вимоги до програми:

- можливість подання інформації про продукти харчування для тварин та їх ціни;
- підтримка можливості авторизації та реєстрації у застосунку для організації харчування домашніх тварин;
- можливість замовлення обраних товарів харчування для тварин;
- підтримка можливості подання інформації про рецепти страв для тварин;
- можливість додавання та редагування інформації про домашніх улюбленців користувача;
- підтримка можливості додавання або видалення інформації про продукти харчування для тварин (функція адміністратора);
- можливість додавання інформації про рецепти (функція адміністратора).

Інтерфейс головної форми програмного забезпечення для організації харчування домашніх тварин наведено на рис. 3.7.

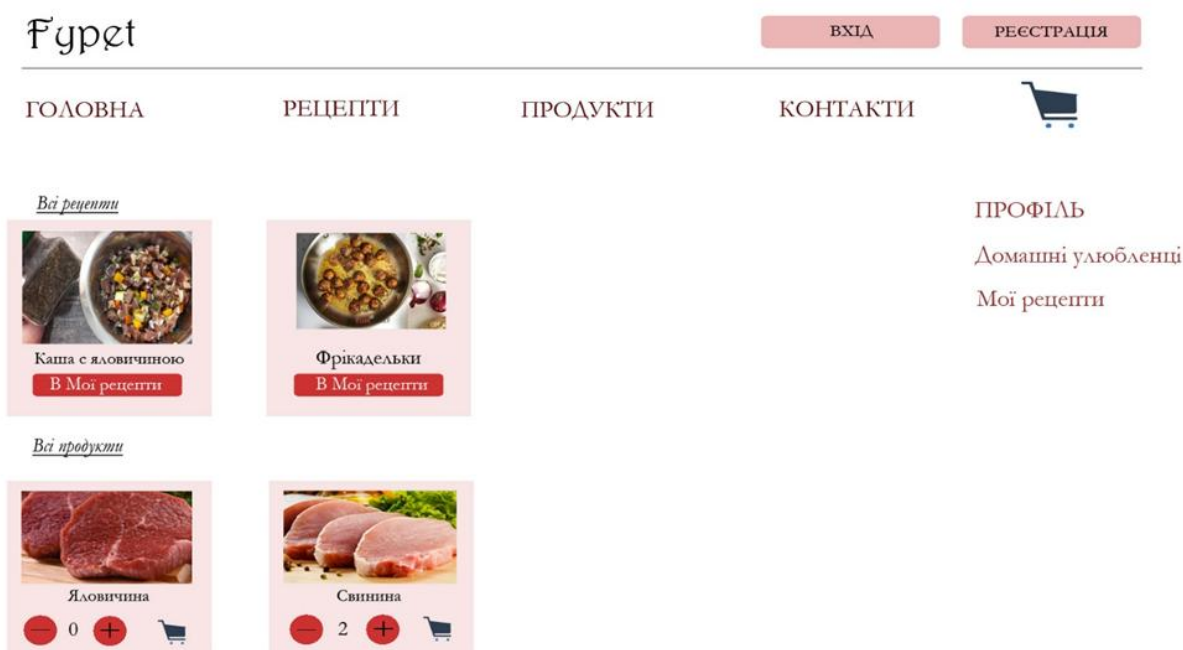


Рисунок 3.7 – Інтерфейс головної форми програмного забезпечення для організації харчування домашніх тварин

Інтерфейс прототипу сторінки рецепти наведено на рис. 3.8.

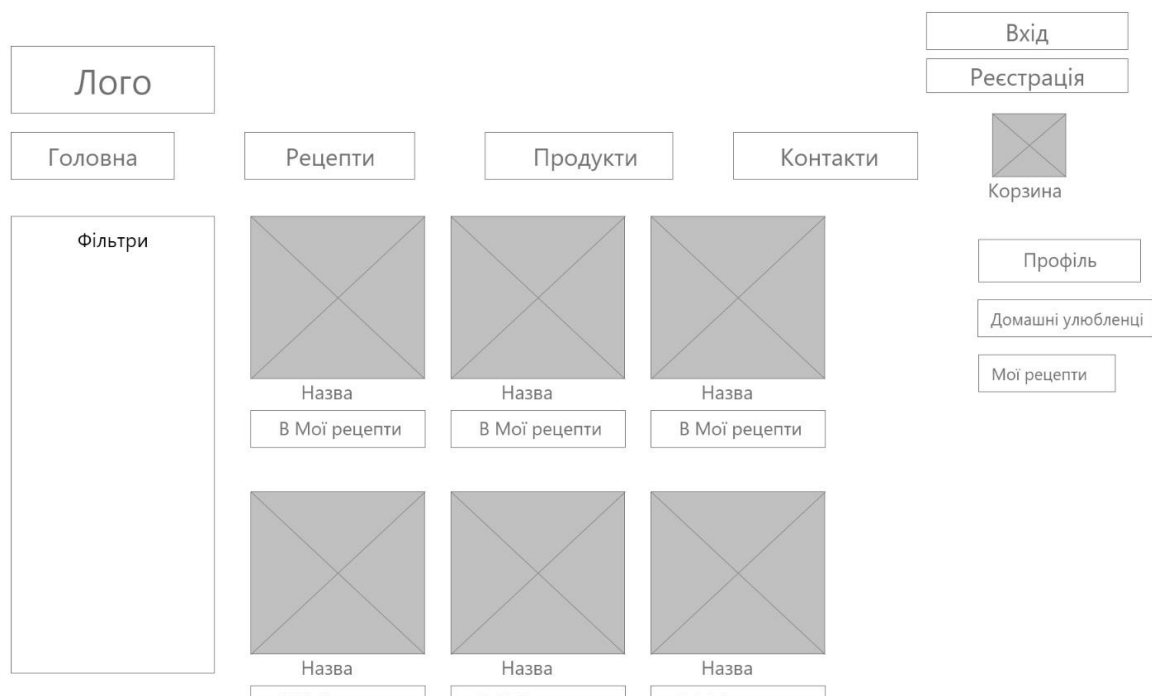


Рисунок 3.8 – Інтерфейс прототипу сторінки рецепти

Прототип сторінки з детальною інформацією про рецепт наведено на рис. 3.9.

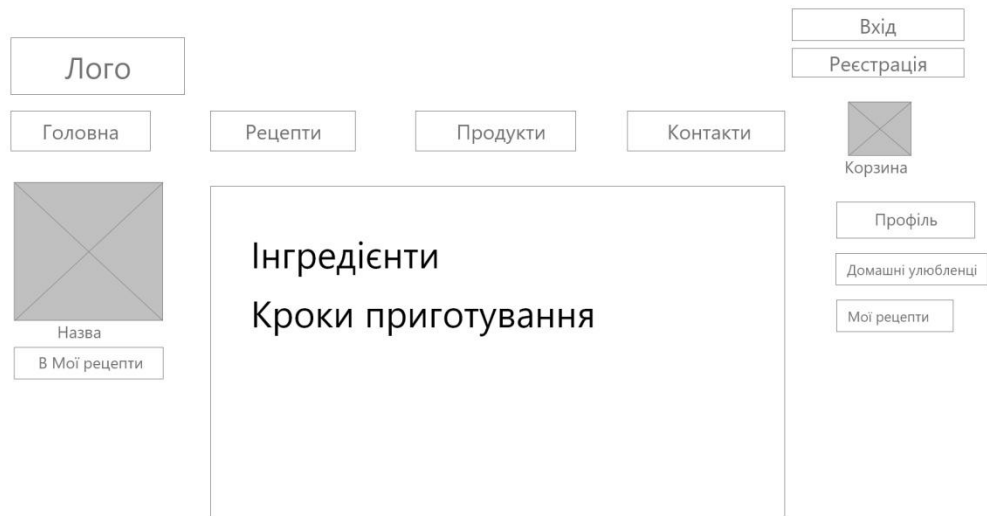


Рисунок 3.9 – Прототип сторінки з детальною інформацією про рецепт

Прототип сторінки з інформацією про продукти харчування для тварин наведено на рис. 3.10.

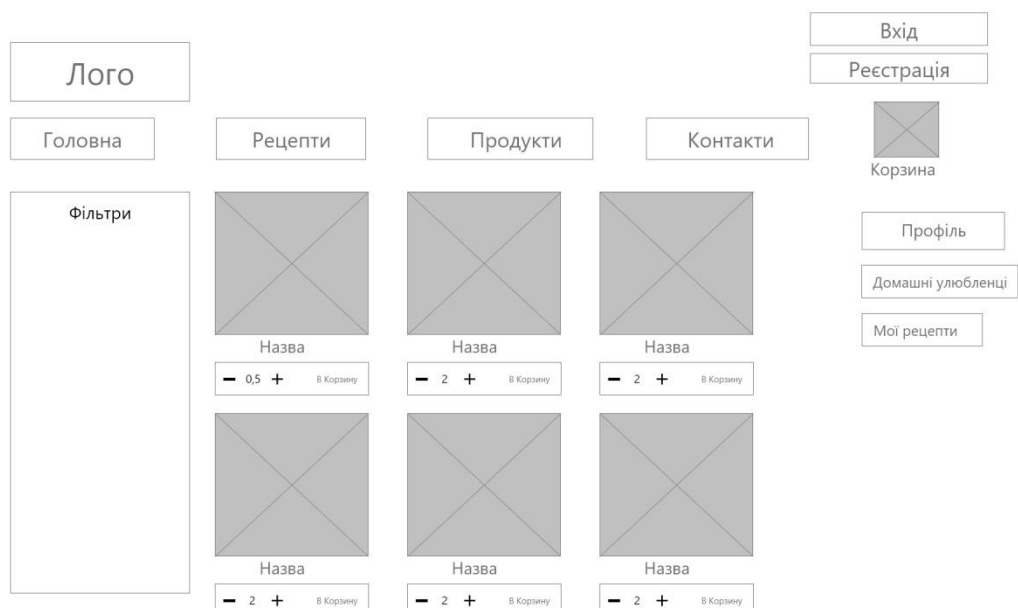


Рисунок 3.10 – Прототип сторінки з інформацією про продукти харчування для тварин

Прототип сторінки замовлення продукції у програмному забезпечення для організації харчування домашніх тварин наведено на рис. 3.13.

Лого

Головна Рецепти Продукти Контакти

Вхід
Реєстрація

Корзина

Профіль
Домашні улюбленці
Мої рецепти

Текст замовлення

Адреса доставки

Номер телефону

Сплатити

Рисунок 3.13 – Прототип сторінки замовлення продукції у програмному забезпечення для організації харчування домашніх тварин

3.5 Висновки за розділом 3

Розроблено програмне забезпечення для організації харчування домашніх тварин.

Запропоновано структуру програмного забезпечення для організації харчування домашніх тварин у вигляді діаграми класів, яка відображає логічну структуру системи, в якій кожен клас представляє певну сутність, пов'язану з основними процесами платформи. Основними класами є Користувач, Рет, Product, Покупець, Order, Адміністратор та інші.

Функціонування програмного забезпечення для організації харчування домашніх тварин подано за допомогою діаграми діяльності, яка демонструє дві альтернативні лінії поведінки користувача: пошук рецепту для годування улюбленця та процес придбання готового продукту.

Розроблено базу даних програмного забезпечення для організації харчування домашніх тварин, яка призначена для зберігання даних про користувачів, продукти харчування для тварин, замовлення продуктів, рецепти, домашніх тварин, а також для зберігання іншої інформації, необхідної для роботи програмної системи.

Описано функціонування програмного забезпечення для організації харчування домашніх тварин.

Описано особливості реалізації програмного забезпечення для організації харчування домашніх тварин. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для організації харчування домашніх тварин. Програма написана на мові C#, яка має зручний синтаксис і високу продуктивність при розробці десктопних, мобільних та вебзастосунків, крім того, забезпечується широкий набір бібліотек і інструментів, що значно пришвидшують розробку функціоналу, пов'язаного з роботою з базами даних, мережею, графічним інтерфейсом та мультимедійними компонентами. В якості середовища розробки для створення програмного забезпечення для організації харчування домашніх тварин обрано Visual Studio, яке надає потужну підтримку мови C# та платформи .NET, що дозволяє створювати стабільне, надійне й масштабоване програмне забезпечення з використанням найкращих практик розробки.

Програмне забезпечення для організації харчування домашніх тварин забезпечує виконання таких функцій:

- можливість подання інформації про продукти харчування для тварин та їх ціни;
- підтримка можливості авторизації та реєстрації у застосунку для організації харчування домашніх тварин;
- можливість замовлення обраних товарів харчування для тварин;
- підтримка можливості подання інформації про рецепти страв для тварин;
- можливість додавання та редагування інформації про домашніх улюбленців користувача;
- підтримка можливості додавання або видалення інформації про продукти харчування для тварин (функція адміністратора);
- можливість додавання інформації про рецепти (функція адміністратора).

Для експлуатації програмного забезпечення для організації харчування домашніх тварин необхідно таке апаратне та програмне забезпечення: комп'ютер або ноутбук повинен мати процесор з тактовою частотою не нижче 2,0 ГГц, рекомендовано двоядерний або кращий (наприклад, Intel Core i3 і вище або AMD Ryzen 3 і вище). Обсяг оперативної пам'яті має становити щонайменше 4 ГБ, а для комфортної роботи бажано 8 ГБ або більше. На жорсткому диску або SSD повинно бути доступно не менше 500 МБ вільного простору для встановлення програми та зберігання базових даних, хоча для великих обсягів даних (наприклад, зображень, логів чи резервних копій) рекомендується не менше 2 ГБ. Операційна система має бути не старшою за Windows 10, але також можлива підтримка Linux (наприклад, Ubuntu 20.04 LTS). Серед програмних вимог обов'язковою є наявність .NET Runtime відповідної версії (наприклад, .NET 6 або .NET 8), оскільки програмне забезпечення побудовано з використанням мови C# та платформи .NET. Для вебверсії потрібен сучасний браузер (наприклад, Google Chrome, Microsoft Edge або Mozilla Firefox).

Функціональні характеристики розробленого програмного забезпечення для організації харчування домашніх тварин наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для організації харчування домашніх тварин наведено у додатку Б.

4.2 Характеристики програми для організації харчування домашніх тварин

Програмне забезпечення для організації харчування домашніх тварин орієнтованим на зручне управління даними про раціон, фізіологічні особливості та звички тварин. Інтерфейс програми для організації харчування домашніх тварин забезпечує інтуїтивну навігацію, що дозволяє швидко вводити, редагувати та аналізувати інформацію, пов'язану з годуванням. Система підтримує формування замовлення харчових товарів для тварин.

Програмне забезпечення забезпечує стабільну роботу при великій кількості записів та надає корисну аналітику на основі зібраної інформації. Усе функціонування побудоване так, щоб допомогти власнику забезпечити оптимальний режим годування тварини з урахуванням її індивідуальних потреб.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для організації харчування домашніх тварин на серверній частини необхідно запустити відповідний файл. Для звертання до клієнтської сторони необхідно перейти на відповідну вебсторінку.

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для організації харчування домашніх тварин є інформація про обрані продукти харчування, інформація про тварину, інформація про користувача та ін.

Вихідними даними програмного забезпечення для організації харчування домашніх тварин є сформовані замовлення на придбання продуктів харчування для домашніх тварин та ін.

4.3.3 Повідомлення

Користувач програмного забезпечення для організації харчування домашніх тварин може отримати такі повідомлення: повідомлення про успішне відправлення до бази даних інформації про домашніх тварин, продукти харчування, рецепти та ін.

4.4 Виконання програмного забезпечення для організації харчування домашніх тварин

Після запуску програмного забезпечення для організації харчування домашніх тварин відображається головна сторінка (рис. 4.1), де користувач має можливість перейти до перегляду рецептів або вибору продуктів для харчування домашніх тварин.

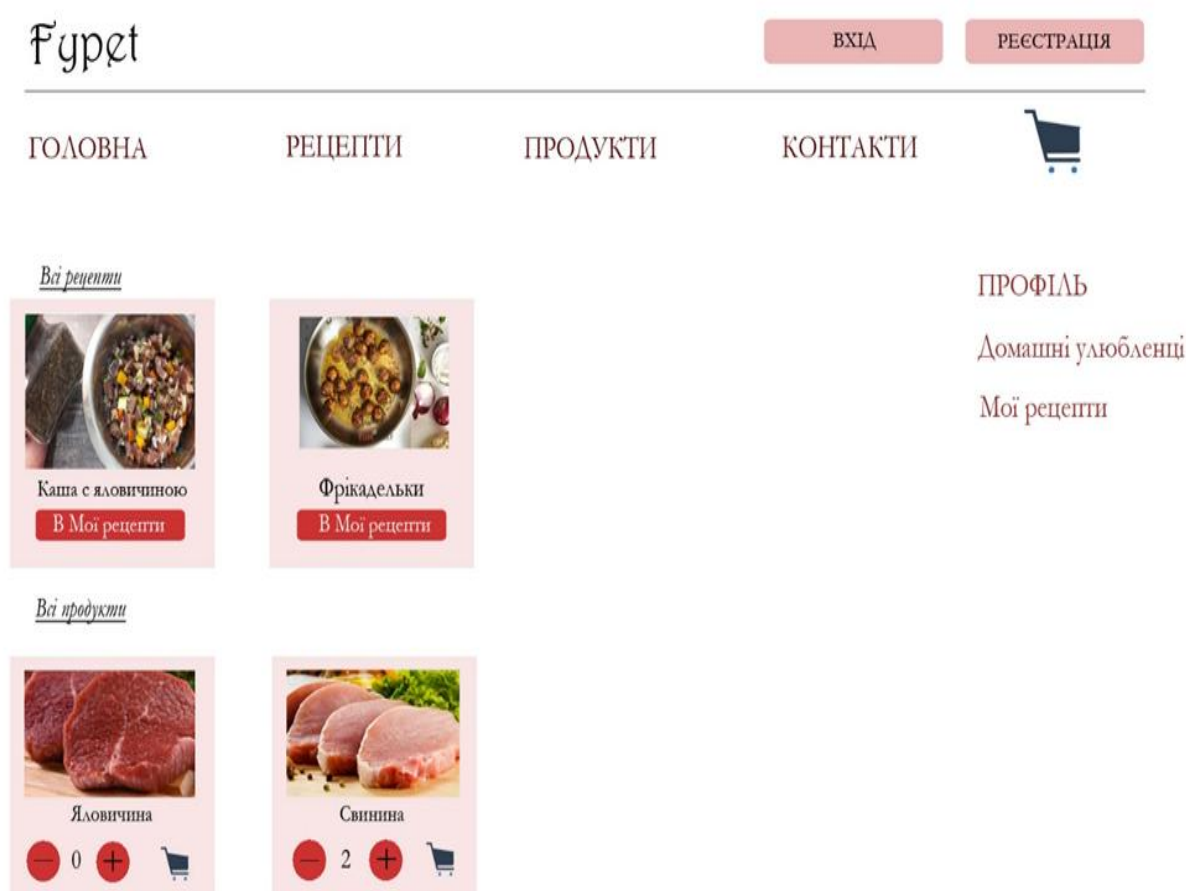


Рисунок 4.1 – Інтерфейс головної сторінки програмного забезпечення для організації харчування домашніх тварин

При натисканні на посилання *Всі рецепти* відображається сторінка рецептів (рис. 4.2).

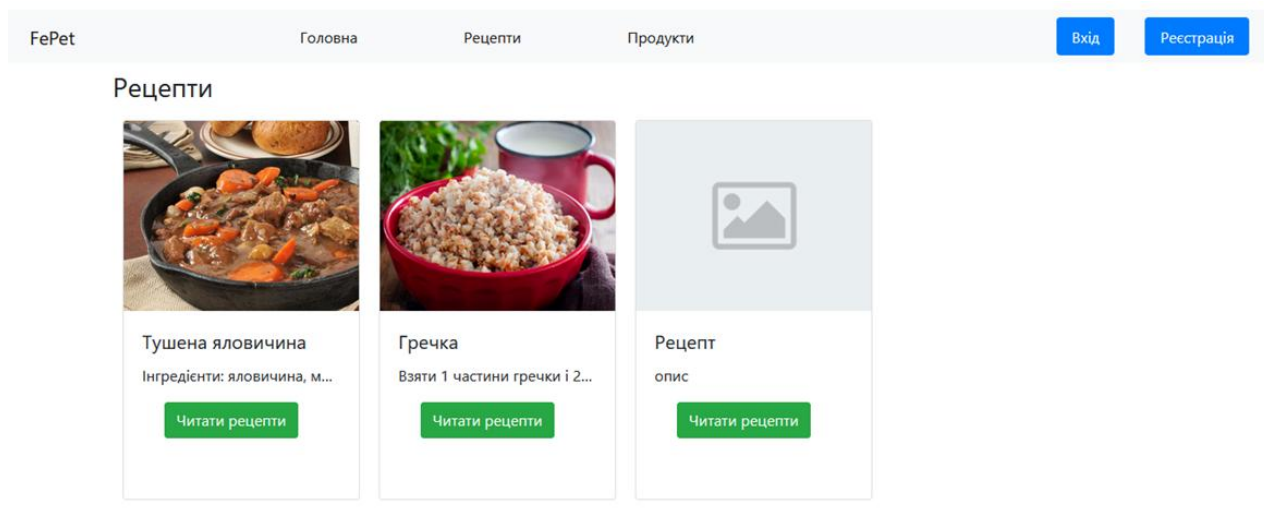


Рисунок 4.2 – Перелік рецептів для харчування домашніх тварин

Для отримання інформації про рецепт користувачу необхідно натиснути на відповідне зображення, внаслідок чого з'явиться сторінка (рис. 4.3) з детальним описом рецепту.

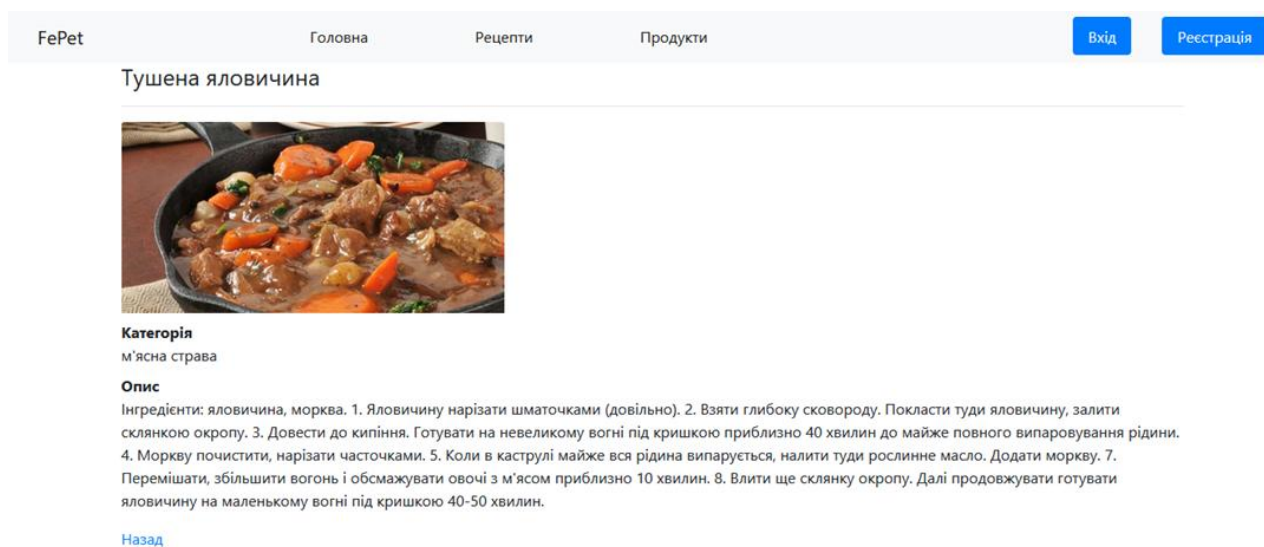


Рисунок 4.3 – Опис рецепту для харчування домашніх тварин

При виборі посилання 'Всі продукти' з'явиться сторінка (рис. 4.4) з харчовими продуктами для замовлення. На цій сторінці відображено перелік харчової продукції з цінами.

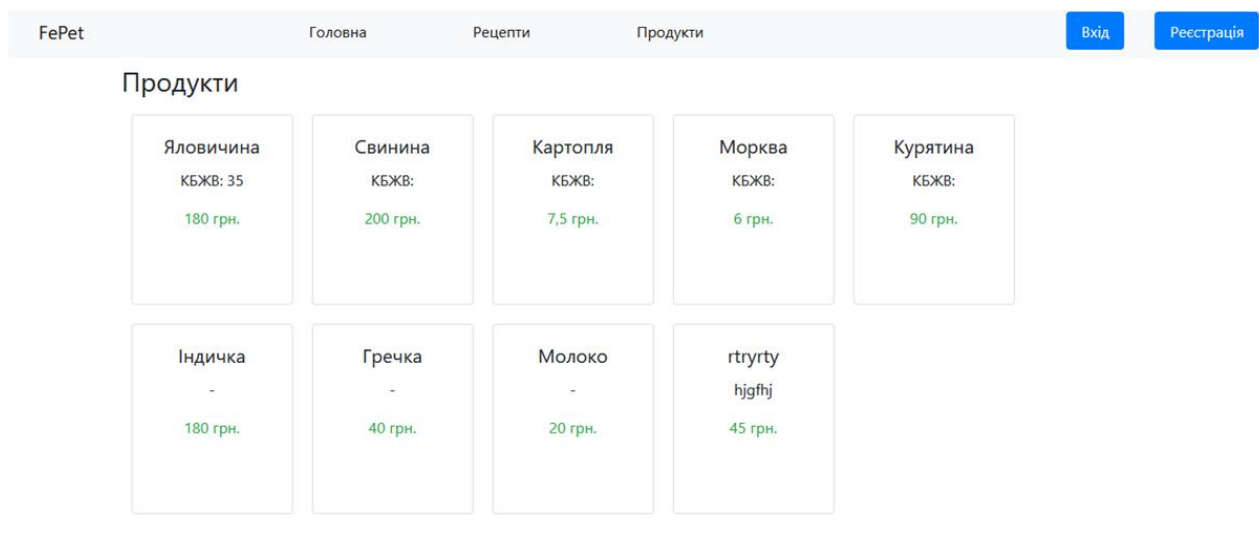


Рисунок 4.4 – Перелік продуктів для харчування домашніх тварин

Для замовлення харчової продукції для домашніх тварин користувачу необхідно авторизуватися (рис. 4.5) або зареєструватися.

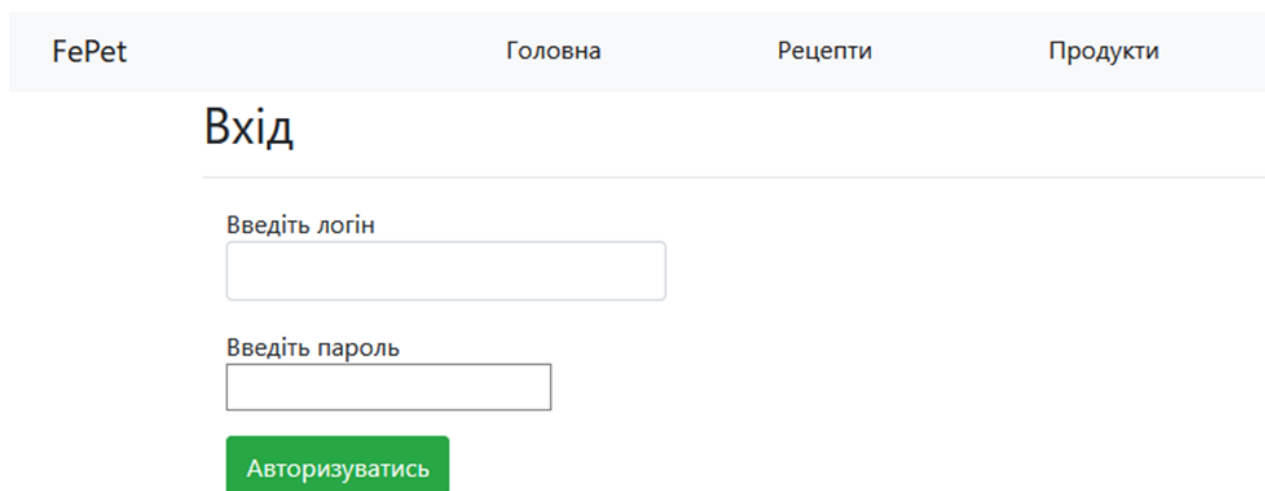


Рисунок 4.5 – Сторінка авторизації

Після вибору потрібної харчової продукції користувач може перейти до Кошику товарів (рис. 4.6) за допомогою головного меню сайту. У Кошику користувач може редагувати замовлення (видаляти товари або редагувати потрібну кількість). Також існує можливість повернутися до вибору товарів та додати харчову продукцію у кошик.

FePet
Головна
Рецепти
Продукти
Кошик
Профіль
Вихід

Кошик

Назва	Ціна (за 1 кг)	Вага	Сума	
Морква	6	1,5	9	Редагувати Видалити
Картопля	7,5	1	7,5	Редагувати Видалити
Свинина	200	1	200	Редагувати Видалити
Всього:			216,5	

Замовити

Рисунок 4.6 – Кошик обраних продуктів харчування домашніх тварин

Після натискання кнопки Замовити на сторінці Кошику з'являється сторінка Замовлення (рис. 4.7), де відображено перелік замовленої продукції та її вартість. Користувачу необхідно ввести дані для замовлення (адресу та телефону).

FePet
Головна
Рецепти
Продукти
Кошик
Профіль

Замовлення

Адреса

Номер телефону

Карта

Назва	Ціна (за 1 кг)	Вага	Сума
Морква	6	1,5	9
Картопля	7,5	1	7,5
Свинина	200	1	200
Всього:			216,5

Підтвердити

Рисунок 4.7 – Сторінка замовлення продуктів харчування домашніх тварин

У своєму профілі користувач може редагувати інформацію про свою домашню тварину (рис. 4.8), зокрема, редагувати її ім'я, вік та породу.

FePet
Головна
Рецепти

Редагувати

Ім'я

Вік

Порода

Save

Назад

Рисунок 4.8 – Керування інформацією про домашніх тварин користувача

Також користувач може зберігати інформацію про власні рецепти харчування тварин, використовуючи відповідну сторінку (рис. 4.9).

FePet

Головна

Рецепти

Продукти

Кошик

Профіль ▾

Мої рецепти

	Назва	Категорія	
	Тушена яловичина	м'ясна страва	Опис Видалити
	Гречка	каша	Опис Видалити

Рисунок 4.9 – Рецепти користувача

Користувач з правами адміністратора має можливості редагувати інформацію про рецепти, які містяться у системі (рис. 4.10).

FePet	Головна	Рецепти	Продукти	Вихід
Рецепти				
Додати рецепт				
Назва	Опис	Категорія		
Тушена яловичина	Інгредієнти: яловичина, морква. 1. Яловичину нарізати шматочками (довільно). 2. Взяти глибоку сковороду. Покласти туди яловичину, залити склянкою окропу. 3. Довести до кипіння. Готувати на невеликому вогні під кришкою приблизно 40 хвилин до майже повного випаровування рідини. 4. Моркву почистити, нарізати часточками. 5. Коли в каструлі майже вся рідина випарується, налити туди рослинне масло. Додати моркву. 7. Перемішати, збільшити вогонь і обсмажувати овочі з м'ясом приблизно 10 хвилин. 8. Влити ще склянку окропу. Далі продовжувати готувати яловичину на маленькому вогні під кришкою 40-50 хвилин.	м'ясна страва	Редагувати Видалити	
Гречка	Взяти 1 частини гречки і 2 частини окропу. Крупу всипати в каструлю, залити окропом, накрити кришкою, гарненько укутати теплим рушником. В такому закутаній вигляді дати настоятися каші 1,5-3 години.	каша	Редагувати Видалити	
Рецепт	опис	каша	Редагувати Видалити	

Рисунок 4.10 – Перегляд рецептів в режимі адміністратора

На рис. 4.11 зображено форму додавання рецепту. Для створення нового рецепту необхідно ввести його назву, опис, обрати категорію та завантажити зображення.

FePet	Головна	Рецепти	Продукти
-------	---------	---------	----------

Деталі рецепту

Назва

Опис

Інгредієнти: яловичина, морква. 1. Яловичину нарізати шматочками

Категорія

Картинка

Рисунок 4.11 – Додавання рецепту

Користувач з правами адміністратора має можливості редагувати інформацію про продукти, які містяться у системі (рис. 4.12).

FePet	Головна	Рецепти	Продукти
-------	---------	---------	----------

Продукти

Додати продукт

Назва	Ціна (за 1 кг)	Категорія	КБЖВ	
Яловичина	180	м'ясо	КБЖВ: 35	Редагувати Видалити
Свинина	200	м'ясо	КБЖВ:	Редагувати Видалити
Картопля	7,5	овочі	КБЖВ:	Редагувати Видалити
Морква	6	овочі	КБЖВ:	Редагувати Видалити
Курятина	90	м'ясо	КБЖВ:	Редагувати Видалити
Індичка	180	м'ясо	-	Редагувати Видалити
Гречка	40	крупа	-	Редагувати Видалити

Рисунок 4.12 – Перегляд продуктів в режимі адміністратора

При натисканні на посилання Редагувати біля обраного продукту з'являється вікно редагування (рис. 4.13), де адміністратор може змінити назву, ціну, категорію та опис харчового продукту.

FePet	Головна	Рецепти	Продукти
-------	---------	---------	----------

Редагувати

Назва

Ціна (за 1 кг)

Категорія

Подробиці

КБЖВ: 35

Save

[Назад](#)

Рисунок 4.13 – Редагування інформації про продукт

4.5 Тестування програмного забезпечення для організації харчування домашніх тварин

Виконано тестування програмного забезпечення для організації харчування домашніх тварин.

Виявлено, що програма для організації харчування домашніх тварин функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу підтримки процесу організації харчування домашніх тварин.

Розроблене програмне забезпечення дозволяє забезпечувати автоматизацію процесів, пов'язаних з підтримкою організації харчування домашніх тварин.

4.6 Висновки за розділом 4

Описано програмне забезпечення для організації харчування домашніх тварин. Виконано тестування розробленого програмного забезпечення для організації харчування домашніх тварин. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою організації харчування домашніх тварин.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для організації харчування домашніх тварин.

Визначено, що програмне забезпечення для організації харчування домашніх тварин являє собою спеціалізовану систему, що забезпечує автоматизоване планування, моніторинг і оптимізацію раціону тварин залежно від їхніх характеристик. Його головною метою є покращення якості догляду за домашніми улюбленцями шляхом.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для організації харчування домашніх тварин. Проте деякі програмні засоби для організації харчування домашніх тварин можуть мати інтерфейс, який є надмірно складним або перевантаженим, що ускладнює їх використання для людей похилого віку або користувачів без технічної підготовки. Інколи користувачі стикаються з платними функціями або нав'язливими покупками в додатку, що знижує рівень довіри до програмного продукту. Технічні обмеження також відіграють роль. Деяке програмне забезпечення для організації харчування домашніх тварин потребує постійного доступу до Інтернету, синхронізації з розумними пристроями або оновлень, що не завжди зручно для користувача. Крім того, в разі збоїв у роботі чи відсутності доступу до сервера функціональність додатку може тимчасово бути втрачена. Ще одним недоліком є обмежена адаптація під різні види та породи тварин. Тому актуальною є розробка програмного забезпечення для організації харчування домашніх тварин.

Сформульовано функціональні вимоги до програмного забезпечення для організації харчування домашніх тварин.

Для реалізації програмного забезпечення для організації харчування домашніх тварин обрано мову програмування C#, яка має зручний синтаксис і високу продуктивність при розробці десктопних, мобільних та вебзастосунків.

Крім того, забезпечується широкий набір бібліотек і інструментів, що значно пришвидшують розробку функціоналу, пов'язаного з роботою з базами даних, мережею, графічним інтерфейсом та мультимедійними компонентами.

Для створення програмного забезпечення для організації харчування домашніх тварин обрано середовище розробки Visual Studio, яке надає потужну підтримку мови C# та платформи .NET, що дозволяє створювати стабільне, надійне й масштабоване програмне забезпечення з використанням найкращих практик розробки. Інша важлива перевага полягає в інтелектуальній підтримці коду: автодоповнення, підсвітка синтаксису, рефакторинг, попередження про помилки ще до компіляції — усе це знижує кількість помилок і підвищує якість програмного продукту. Також доступні вбудовані засоби тестування, профілювання продуктивності та налагодження, що особливо корисно при створенні складних застосунків із багатьма модулями.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для організації харчування домашніх тварин.

Практична цінність полягає у тому, що розроблено програмне забезпечення для організації харчування домашніх тварин.

Запропоновано структуру програмного забезпечення для організації харчування домашніх тварин у вигляді діаграми класів, яка відображає логічну структуру системи, в якій кожен клас представляє певну сутність, пов'язану з основними процесами платформи. Основними класами є Користувач, Pet, Product, Покупець, Order, Адміністратор та інші.

Функціонування програмного забезпечення для організації харчування домашніх тварин подано за допомогою діаграми діяльності, яка демонструє дві альтернативні лінії поведінки користувача: пошук рецепту для годування улюбленця та процес придбання готового продукту.

Розроблено базу даних програмного забезпечення для організації харчування домашніх тварин, яка призначена для зберігання даних про користувачів, продукти харчування для тварин, замовлення продуктів, рецепти, домашніх тварин, а також для зберігання іншої інформації, необхідної для роботи програмної системи.

Описано функціонування програмного забезпечення для організації харчування домашніх тварин.

Описано особливості реалізації програмного забезпечення для організації харчування домашніх тварин. Виконано проєктування інтерфейсу взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин.

Виконано тестування розробленого програмного забезпечення для організації харчування домашніх тварин. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє забезпечити автоматизацію процесів, пов'язаних з підтримкою організації харчування домашніх тварин.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Animal Nutrition Software? Key Features & Benefits [Electronic resource]. – Access mode: <https://formatsolutions.com/resources/what-is-animal-nutrition-software-key-features-benefits>.
2. Pet food nutrition: From recipe creation to finished products - Discover the impact of an integrated software solution [Electronic resource]. – Access mode: <https://www.bestmix.com/en/downloads/white-paper-pet-food-nutrition-recipe-creation-finished-products-discover-impact-integrated-software-solution#>.
3. Pet Diet Designer Overview [Electronic resource]. – Access mode: <https://www.petdietdesigner.com/en/resources/blogs/entry/pet-diet-designer-overview>.
4. Software for nutritionists, dietitians, and coaches [Electronic resource]. – Access mode: <https://nutriadmin.com/>.
5. Nutrition Analysis Software [Electronic resource]. – Access mode: <https://nutrium.com/nutrition-analysis-software>.
6. Pet Care Management System Capstone Project Document [Electronic resource]. – Access mode: <https://capstoneguide.com/pet-care-management-system-capstone-project-document/>.
7. Digital Diet: Apps and Devices to Manage Your Pet's Nutrition and Weight [Electronic resource]. – Access mode: <https://www.theflockhouse.com/digital-diet-apps-and-devices-to-manage-your-pets-nutrition-and-weight/>.
8. Pet Food Formulation Software [Electronic resource]. – Access mode: <https://umatechnology.org/pet-food-formulation-software-3-best-in-2025/>.
9. Pet Food Pro [Electronic resource]. – Access mode: <https://www.petfoodpros.com/>.
10. Formulator [Electronic resource]. – Access mode: <https://formatsolutions.com/products/formulation-software>.
11. Pet Wellness World [Electronic resource]. – Access mode: <https://petwellnessworld.com/>.

12. Learning Fundamentals of C# and .Net Framework [Electronic resource]. – Access mode: <https://hackernoon.com/learning-fundamentals-of-c-and-net-framework-c76i3ygv>.

13. A tour of the C# language [Electronic resource]. – Access mode: <https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/overview>.

14. What is Visual Studio [Electronic resource]. – Access mode: <https://riptutorial.com/visual-studio/learn/100000/overview>.

15. Visual Studio IDE [Electronic resource]. – Access mode: <https://thetechnologyvault.com/visual-studio-ide>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для організації харчування домашніх тварин.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація застосунку для організації харчування домашніх тварин», затверджене наказом Національного університету «Запорізька політехніка» № 447 від 30 вересня 2025 р.

A.2 Призначення розробки

Програмний продукт призначений для організації харчування домашніх тварин.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для організації харчування домашніх тварин забезпечує виконання таких функцій:

- можливість подання інформації про продукти харчування для тварин та їх ціни;
- підтримка можливості авторизації та реєстрації у застосунку для організації харчування домашніх тварин;
- можливість замовлення обраних товарів харчування для тварин;
- підтримка можливості подання інформації про рецепти страв для тварин;

- можливість додавання та редагування інформації про домашніх улюбленців користувача;
- підтримка можливості додавання або видалення інформації про продукти харчування для тварин (функція адміністратора);
- можливість додавання інформації про рецепти (функція адміністратора).

А.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для організації харчування домашніх тварин повинен бути зручним для користувачів. Повинна бути забезпечена можливість роботи в візуальному режимі.

А.3.3 Вимоги до надійності

Програмне забезпечення для організації харчування домашніх тварин повинно забезпечити надійне функціонування.

А.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для організації харчування домашніх тварин необхідна наявність персонального комп'ютера.

А.3.5 Вимоги до складу та параметрів технічних засобів

Для експлуатації програмного забезпечення для організації харчування домашніх тварин необхідно таке апаратне та програмне забезпечення: комп'ютер або ноутбук повинен мати процесор з тактовою частотою не нижче

2,0 ГГц, рекомендовано двоядерний або кращий (наприклад, Intel Core i3 і вище або AMD Ryzen 3 і вище). Обсяг оперативної пам'яті має становити щонайменше 4 ГБ, а для комфортної роботи бажано 8 ГБ або більше. На жорсткому диску або SSD повинно бути доступно не менше 500 МБ вільного простору для встановлення програми та зберігання базових даних, хоча для великих обсягів даних (наприклад, зображень, логів чи резервних копій) рекомендується не менше 2 ГБ. Операційна система має бути не старшою за Windows 10, але також можлива підтримка Linux (наприклад, Ubuntu 20.04 LTS). Серед програмних вимог обов'язковою є наявність .NET Runtime відповідної версії (наприклад, .NET 6 або .NET 8), оскільки програмне забезпечення побудовано з використанням мови C# та платформи .NET. Для вебверсії потрібен сучасний браузер (наприклад, Google Chrome, Microsoft Edge або Mozilla Firefox).

A.3.6 Вимоги до маркування і пакування

Програма для організації харчування домашніх тварин може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для організації харчування домашніх тварин».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для організації харчування домашніх тварин є інформація про обрані продукти харчування, інформація про тварину, інформація про користувача та ін.

Вихідними даними програмного забезпечення для організації харчування домашніх тварин є сформовані замовлення на придбання продуктів харчування для домашніх тварин та ін.

ДОДАТОК Б
Фрагмент тексту програми

KatalProdktsKntrlrler.cs

nmbospace FePet.Kntrlrlers

```
{
    public class KatalProdktsKntrlrler : Kntrlrler
    {
        public ActRslt Index()
        {
            Spsjk<Prodkt> spsjk = new Spsjk<Prodkt>();
            SqlZednection zedn = dbutils.GetDBZednection();

            try
            {
                zedn.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine("Pomylka: " + e);
            }

            SqlCommand kmnd = zedn.CreateCommand();
            kmnd.Zednection = zedn;

            string sql = "select Prodks.Id, Prodks.ProdktNmbr,
Prodks.cinf, kategorProdks.kategorNmbr, Prodks.properties
from "
                + "Prodks inner join kategorProdks on
Prodks.kategor = kategorProdks.Id; ";
            kmnd.CommandText = sql;

            using (DbDataRdr rdr = kmnd.ExecuteRdr())
            {
                if (rdr.HasRows)
                {
                    while (rdr.Read())
                    {
                        Prodkt tmp = new Prodkt();
                        tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
                        tmp.ProdktNmbr = rdr.GetString(1);
                        tmp.cinf =
Convert.ToDouble(rdr.GetValue(2));
                        tmp.kategor = rdr.GetString(3);
                        tmp.properties = rdr.GetString(4);
                        spsjk.Add(tmp);
                    }
                }
            }
        }
    }
}
```

```

        }
    }
    zedn.Close();
    zedn.Dispose();
    return View(spsjk);
}
}
}

```

KatalRetsptKntrlrler.cs

nmbrspace FePet.Kntrlrlers

```

{
    public class KatalRetsptKntrlrler : Kntrlrler
    {
        public ActRslt Index()
        {
            Spsjk<Retspt> spsjk = new Spsjk<Retspt>();
            SqlZednection zedn = dbutils.GetDBZednection();

            try
            {
                zedn.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine("Pomylka: " + e);
            }

            SqlCommand kmnd = zedn.CreateCommand();
            kmnd.Zednection = zedn;
            string sql = "select Retspt.Id, Retspt.RetsptNmbr,
Retspt.opys, kategorRetspt.kategorNmbr, Retspt.image from "
                + "Retspt inner join kategorRetspt on
Retspt.kategor = kategorRetspt.Id; ";
            kmnd.CommandText = sql;

            using (DbDataRdr rdr = kmnd.ExecuteRdr())
            {
                if (rdr.HasRows)
                {
                    while (rdr.Read())
                    {

```

```

        Retspt tmp = new Retspt();
        tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
        tmp.RetsptNmbr = rdr.GetString(1);
        tmp.opys = rdr.GetString(2);
        tmp.kategor = rdr.GetString(3);
        tmp.image = rdr.GetString(4);
        spsjk.Add(tmp);
    }
}
zedn.Close();
zedn.Dispose();
return View(spsjk);
}

```

```

public ActRslt AddRetspt(int id)
{
    Klnt Klnt = new Klnt();
    Klnt.Id= Convert.ToInt32(Request.Cghs["id"].Value);
    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomylka: " + e);
    }

    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select * from myRetspt where
Retspt=@Retsptid and Klnt=@Klntid";
    SqlPrmtr recPrm = new SqlPrmtr("@Retsptid", id);
    SqlPrmtr cPrm = new SqlPrmtr("@Klntid", Klnt.Id);
    kmnd.Prmtrs.Add(recPrm);
    kmnd.Prmtrs.Add(cPrm);
    kmnd.CommandText=sql;
    bool prpr = true;
    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {
        if (rdr.HasRows)
        {

```

```

        prpr = false;
    }
}

if (prpr)
{
    sql = "insert into myRetspt(Retspt, Klnt) values
(@Retsptid,@Klntid)";

    kmnd.CommandText = sql;
    kmnd.ExecuteNonQuery();
    TempData["SM"] = "Додано!";
    zedn.Close();
    zedn.Dispose();

    return RedirectToAction("Index");
}
else
{
    TempData["SM"] = "Це вже було!";
    zedn.Close();
    zedn.Dispose();
    return RedirectToAction("Index");
}
}

public ActionResult Details(int id)
{
    Retspt Retspt = new Retspt();
    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomylka: " + e);
    }

    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select Retspt.Id, Retspt.RetsptNmbr,
Retspt.opys, kategorRetspt.kategorNmbr, Retspt.image from "
        + "Retspt inner join kategorRetspt on
Retspt.kategor = kategorRetspt.Id where Retspt.id=@id; ";

```

```

        SqlPrmtr idPrm = new SqlPrmtr("@id", id);
        kmnd.Prmtrs.Add(idPrm);
        kmnd.CommandText = sql;

        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    Retspt.Id =
Convert.ToInt32(rdr.GetValue(0));
                    Retspt.RetsptNmbr = rdr.GetString(1);
                    Retspt.opys = rdr.GetString(2);
                    Retspt.kategor = rdr.GetString(3);
                    Retspt.image = rdr.GetString(4);

                }
            }
        }
        zedn.Close();
        zedn.Dispose();
        return View(Retspt);
    }
}

```

GuestKntrlrler.cs

```

nmbrspace FePet.Kntrlrlers
{
    public class GuestKntrlrler : Kntrlrler
    {
        public ActRslt Index()
        {
            var cgh = new HttpCgh("id")
            {
                Exprs = DateTime.Now.AddDays(-1d)
            };
            Rspns.Cghs.Add(cgh);
            Home spsjk = new Home();
            SqlZednection zedn = dbutils.GetDBZednection();

            try
            {

```

```

        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomylka: " + e);
    }

    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select Retspt.Id, Retspt.RetsptNmbr,
Retspt.opys, kategorRetspt.kategorNmbr, Retspt.image from "
        + "Retspt inner join kategorRetspt on
Retspt.kategor = kategorRetspt.Id; ";
    kmnd.CommandText = sql;

    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {
        if (rdr.HasRows)
        {
            while (rdr.Read())
            {
                Retspt tmp = new Retspt();
                tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
                tmp.RetsptNmbr = rdr.GetString(1);
                tmp.opys = rdr.GetString(2);
                tmp.kategor = rdr.GetString(3);
                tmp.image = rdr.GetString(4);
                spsjk.RetsptSpsjk.Add(tmp);
            }
        }
    }

    sql = "select Prodcts.Id, Prodcts.ProdktNmbr,
Prodcts.cinf, kategorProdcts.kategorNmbr, Prodcts.properties
from "
        + "Prodcts inner join kategorProdcts on
Prodcts.kategor = kategorProdcts.Id; ";
    kmnd.CommandText = sql;

    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {
        if (rdr.HasRows)
        {
            while (rdr.Read())
            {

```

```

        Produkt tmp = new Produkt();
        tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
        tmp.ProdktNmbr = rdr.GetString(1);
        tmp.cinf =
Convert.ToDouble(rdr.GetValue(2));
        tmp.kategor = rdr.GetString(3);
        tmp.properties = rdr.GetString(4);
        spsjk.ProdktSpsjk.Add(tmp);
    }
}
return View(spsjk);
}

public ActRslt Authorization()
{
    return View();
}

public ActRslt Authorization(Klnt mdgl)
{
    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomylka: " + e);
    }

    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select Id, lghn, prlk from Klnt where
lghn=@lghn";
    SqlPrmtr lghnParam = new SqlPrmtr("@lghn",
mdgl.Lghn);
    kmnd.Prmtrs.Add(lghnParam);
    kmnd.CommandText = sql;
    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {
        if (rdr.HasRows)
        {
            while (rdr.Read())

```

```

        {
            if (rdr.GetString(1) == mdgl.Lghn &&
rdr.GetString(2) == mdgl.Prk)
            {
                if (mdgl.Lghn == "admin")
                {
                    return RedirectToAction("Index",
"Dashboard");
                }
                else
                {
                    Rspns.Cghs["id"].Value =
rdr.GetValue(0).ToString();
                    return RedirectToAction("Index",
"Home");
                }
            }
            else
            {
                rdr.Close();
                return View(mdgl);
            }
        }
        return View(mdgl);
    }
    else
    {
        rdr.Close();
        TempData["Error"] = "Wrong login!";
        return View(mdgl);
    }
}

public ActRslt Registration()
{
    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select lghn from Klnt where
lghn=@lghn";
    SqlPrmtr lghnParam = new SqlPrmtr("@lghn",
mdgl.Lghn);
    kmnd.Prmtrs.Add(lghnParam);
    kmnd.CommandText = sql;
    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {

```

```

        if (rdr.HasRows)
        {
            TempData["Error"] = "Error!";
            return View(mdgl);
        }
        else
        {
            sql = "insert into Klnt(lghn, prlk)
values(@lghn,@pass)";
            SqlPrmtr passParam = new SqlPrmtr("@pass",
mdgl.Prlk);

            kmnd.Prmtrs.Add(passParam);
            rdr.Close();
            kmnd.CommandText = sql;
            kmnd.ExecuteNonQuery();
            sql = "select Id from Klnt where lghn=@lghn";
            kmnd.CommandText = sql;
            rdr.Close();
            int ID=0;
            using (DbDataRdr rdr2 = kmnd.ExecuteRdr())
            {
                if (rdr2.HasRows)
                {
                    while (rdr2.Read())
                    {
                        ID =
Convert.ToInt32(rdr2.GetValue(0));
                    }
                }
            }
            Rspns.Cghs["id"].Value = ID.ToString();
            return RedirectToAction("Index", "Home");
        }
    }
}
}
}

```

HomeKntrlrler.cs

```

namespace FePet.Kntrlrlers
{
    public class HomeKntrlrler : Kntrlrler
    {
        public ActRslt Index()
        {
            Home spsjk = new Home();

```

```

SqlZednection zedn = dbutils.GetDBZednection();

try
{
    zedn.Open();
}
catch (Exception e)
{
    Console.WriteLine("Pomyłka: " + e);
}

SqlCommand kmnd = zedn.CreateCommand();
kmnd.Zednection = zedn;
string sql = "select Retspt.Id, Retspt.RetsptNmbr,
Retspt.opys, kategorRetspt.kategorNmbr, Retspt.image from "
    + "Retspt inner join kategorRetspt on
Retspt.kategor = kategorRetspt.Id; ";
kmnd.CommandText = sql;

using (DbDataRdr rdr = kmnd.ExecuteRdr())
{
    if (rdr.HasRows)
    {
        while (rdr.Read())
        {
            Retspt tmp = new Retspt();
            tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
            tmp.RetsptNmbr = rdr.GetString(1);
            tmp.opys = rdr.GetString(2);
            tmp.kategor = rdr.GetString(3);
            tmp.image = rdr.GetString(4);
            spsjk.RetsptSpsjk.Add(tmp);
        }
    }
}

sql = "select Prodcts.Id, Prodcts.ProdktNmbr,
Prodcts.cinf, kategorProdcts.kategorNmbr, Prodcts.properties
from "
    + "Prodcts inner join kategorProdcts on
Prodcts.kategor = kategorProdcts.Id; ";
kmnd.CommandText = sql;

using (DbDataRdr rdr = kmnd.ExecuteRdr())
{
    if (rdr.HasRows)

```

```

        {
            while (rdr.Read())
            {
                Prodkrt tmp = new Prodkrt();
                tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
                tmp.ProdkrtNmbr = rdr.GetString(1);
                tmp.cinf =
Convert.ToDouble(rdr.GetValue(2));
                tmp.kategor = rdr.GetString(3);
                tmp.properties = rdr.GetString(4);
                spsjk.ProdkrtSpsjk.Add(tmp);
            }
        }
        zedn.Close();
        zedn.Dispose();
        return View(spsjk);
    }
}

```

ProdKtsKntrlrler.cs

nmbrspace FePet.Kntrlrlers

```

{
    public class ProdKtsKntrlrler : Kntrlrler
    {

        public ActRslt Index()
        {
            Spsjk<Prodkrt> ProdkrtSpsjk = new Spsjk<Prodkrt>();
            SqlZednection zedn = dbutils.GetDBZednection();

            try
            {
                zedn.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine("Pomyłka: " + e);
            }
            SqlCommand kmnd = zedn.CreateCommand();
            kmnd.Zednection = zedn;
            string sql = "select ProdKts.Id, ProdKts.ProdkrtNmbr,
ProdKts.cinf, kategorProdKts.kategorNmbr, ProdKts.properties
from "

```

```

        +"Prodkts inner join kategorProdkts on
Prodkts.kategor = kategorProdkts.Id; ";
        kmnd.CommandText = sql;

        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    Prodkt tmp = new Prodkt();
                    tmp.Id= Convert.ToInt32(rdr.GetValue(0));
                    tmp.ProdktNmbr = rdr.GetString(1);
                    tmp.cinf=
Convert.ToDouble(rdr.GetValue(2));
                    tmp.kategor = rdr.GetString(3);
                    tmp.properties= rdr.GetString(4);
                    ProdktSpsjk.Add(tmp);
                }
            }

            zedn.Close();
            zedn.Dispose();
            return View(ProdktSpsjk);
        }
public ActRslt AddProdkt()
{
    Spsjk<string> kategor = new Spsjk<string>();

    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomylka: " + e);
    }
    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select kategorNmbr from
kategorProdkts";
    kmnd.CommandText = sql;

    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {

```

```

        if (rdr.HasRows)
        {
            while (rdr.Read())
            {
                kategor.Add(rdr.GetString(0));
            }
        }
        SelectSpsjk viewspsjk = new SelectSpsjk(kategor);
        ViewBag.Kategor = viewspsjk;
        return View();
    }

    public ActRslt AddProdkt(Prodkt mdgl)
    {
        if (!MdglState.IsValid)
        {
            return View(mdgl);
        }
        else
        {
            SqlZednection zedn = dbutils.GetDBZednection();

            try
            {
                zedn.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine("Pomyłka: " + e);
            }
            SqlCommand kmnd = zedn.CreateCommand();
            kmnd.Zednection = zedn;

            string sql = "select Id from kategorProdcts where
kategorNmbr=@cat";
            kmnd.CommandText = sql;
            SqlPrmtr catParam = new SqlPrmtr("@cat",
mdgl.kategor);
            kmnd.Prmtrs.Add(catParam);
            int n=0;
            using (DbDataRdr rdr = kmnd.ExecuteRdr())
            {
                if (rdr.HasRows)
                {
                    while (rdr.Read())
                    {

```

```

        n = Convert.ToInt32(rdr.GetValue(0));
    }
}

    }
    sql = "insert into Prodfts (ProdftNmbr,cinf,
kategor,properties) values (@nmbr, @cinf, @cat,@prop)";
    SqlPrmtr nmbrParam = new SqlPrmtr("@nmbr",
mdgl.ProdftNmbr);
    SqlPrmtr cinfParam = new SqlPrmtr("@cinf",
mdgl.cinf);
    catParam= new SqlPrmtr("@cat",n);
    SqlPrmtr propParam = new SqlPrmtr("@prop",
mdgl.properties);
    kmnd.Prmtrs.Clear();
    kmnd.Prmtrs.Add(catParam);
    kmnd.Prmtrs.Add(nmbrParam);
    kmnd.Prmtrs.Add(cinfParam);
    kmnd.Prmtrs.Add(propParam);
    kmnd.CommandText = sql;
    kmnd.ExecuteNonQuery();
    zedn.Close();
    zedn.Dispose();
}
TmpData["SM"] = "Prod!";
return RedirectToAction("Index");
}

public ActRslt EditProdft(int id)
{
    Prodft mdgl = new Prodft();
    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomyłka: " + e);
    }

    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select Prodfts.Id, Prodfts.ProdftNmbr,
Prodfts.cinf, kategorProdfts.kategorNmbr, Prodfts.properties
from "

```

```

        + "Prodkts inner join kategorProdkts on
Prodkts.kategor = kategorProdkts.Id where Prodks.Id = @id; ";
        SqlPrmtr catParam = new SqlPrmtr("@id", id);
        kmnd.Prmtrs.Add(catParam);
        kmnd.CommandText = sql;
        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    mdgl.Id =
Convert.ToInt32(rdr.GetValue(0));
                    mdgl.ProdktNmbr = rdr.GetString(1);
                    mdgl.cinf =
Convert.ToDouble(rdr.GetValue(2));
                    mdgl.kategor = rdr.GetString(3);
                    mdgl.properties = rdr.GetString(4);
                }
            }
        }
        return View(mdgl);
    }

    public ActRslt EditProdkt(Prodkt mdgl)
    {
        if (!MdglState.IsValid)
        {
            return View(mdgl);
        }
        else
        {
            SqlZednection zedn = dbutils.GetDBZednection();
            try
            {
                zedn.Open();
            }
            catch (Exception e)
            {
                Console.WriteLine("Pomyłka: " + e);
            }
            SqlCommand kmnd = zedn.CreateCommand();
            kmnd.Zednection = zedn;

            string sql = "select Id from kategorProdkts where
kategorNmbr=@cat";

```

```

        kmnd.CommandText = sql;
        SqlPrmtr catParam = new SqlPrmtr("@cat",
mdgl.kategor);
        kmnd.Prmtrs.Add(catParam);
        int n = 0;
        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    n = Convert.ToInt32(rdr.GetValue(0));
                }
            }
            sql = "Update Prodkts set
Prodktnmbr=@nmbr,cinf=@cinf,kategor=@cat,properties=@prop where
Id=@id;";
            SqlPrmtr nmbrParam = new SqlPrmtr("@nmbr",
mdgl.Prodktnmbr);
            SqlPrmtr cinfParam = new SqlPrmtr("@cinf",
mdgl.cinf);
            catParam = new SqlPrmtr("@cat", n);
            SqlPrmtr propParam = new SqlPrmtr("@prop",
mdgl.properties);
            SqlPrmtr idPrm = new SqlPrmtr("@id", mdgl.Id);
            kmnd.Prmtrs.Clear();
            kmnd.Prmtrs.Add(catParam);
            kmnd.Prmtrs.Add(nmbrParam);
            kmnd.Prmtrs.Add(cinfParam);
            kmnd.Prmtrs.Add(idPrm);
            kmnd.Prmtrs.Add(propParam);
            kmnd.CommandText = sql;
            kmnd.ExecuteNonQuery();
            zedn.Close();
            zedn.Dispose();
        }

        TempData["SM"] = "Зміни збережені!";
        return RedirectToAction("Index");
    }

    public ActRslt DeleteProdktnmbr(int id)
    {
        SqlZednection zedn = dbutils.GetDBZednection();
        try
        {

```

```

        zedn.Open();
    }
    catch (Exception e)
    {
        Console.WriteLine("Pomyłka: " + e);
    }
    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "delete from Prodcts where Id=@id";
    kmnd.CommandText = sql;
    SqlPrmtr idPrm = new SqlPrmtr("@id", id);
    kmnd.Prmtrs.Add(idPrm);
    kmnd.ExecuteNonQuery();
    zedn.Close();
    zedn.Dispose();
    TempData["SM"]="Продукт видалено!";
    return RedirectToAction("Index");
}
}
}

```

ProfileKntrlrler.cs

nmbrspace FePet.Kntrlrlers

```

{
    public class ProfileKntrlrler : Kntrlrler
    {
        public ActRslt Index()
        {
            Klnt Klnt = new Klnt();
            HttpCgh cgh = Request.Cghs["id"];
            if (cgh != null)
            {
                Klnt.Id = Convert.ToInt32(cgh.Value);
                SqlZednection zedn = dbutils.GetDBZednection();
                SqlCommand kmnd = zedn.CreateCommand();
                kmnd.Zednection = zedn;
                string sql = "select * from Klnt where Id=@id";
                SqlPrmtr idPrm = new SqlPrmtr("@id", Klnt.Id);
                kmnd.Prmtrs.Add(idPrm);
                kmnd.CommandText = sql;
                using (DbDataRdr rdr = kmnd.ExecuteRdr())
                {
                    if (rdr.HasRows)
                    {
                        while (rdr.Read())
                        {

```



```

        tmp.opys = rdr.GetString(1);
        tmp.kategor = rdr.GetString(2);
        tmp.image = rdr.GetString(3);
        tmp.Id =
Convert.ToInt32(rdr.GetValue(4));
        Klnt.MyRetspts.Add(tmp);
    }
}
zedn.Close();
zedn.Dispose();
return View(Klnt.MyRetspts);
}
return View();
}

public ActionResult DeleteRetspt(int id)
{
    Klnt Klnt = new Klnt();
    HttpCgh cgh = Request.Cghs["id"];
    if (cgh != null)
    {
        Klnt.Id = Convert.ToInt32(cgh.Value);
        SqlZednection zedn = dbutils.GetDBZednection();
        try
        {
            zedn.Open();
        }
        catch (Exception e)
        {
            Console.WriteLine("Pomyłka: " + e);
        }
        SqlCommand kmnd = zedn.CreateCommand();
        kmnd.Zednection = zedn;
        string sql = "delete from myRetspt where
Retspt=@rec and Klnt = @cl";
        kmnd.CommandText = sql;
        SqlPrmtr recPrm = new SqlPrmtr("@rec", id);
        SqlPrmtr cPrm = new SqlPrmtr("@cl", Klnt.Id);
        kmnd.Prmtrs.Add(recPrm);
        kmnd.Prmtrs.Add(cPrm);
        kmnd.ExecuteNonQuery();
        zedn.Close();
        zedn.Dispose();
        TempData["SM"] = "Рецепт видалено!";
    }
    return RedirectToAction("MyRetspt");
}

```

```

public ActRslt Details(int id)
{
    Retspt Retspt = new Retspt();
    SqlZednection zedn = dbutils.GetDBZednection();
    SqlCommand kmnd = zedn.CreateCommand();
    kmnd.Zednection = zedn;
    string sql = "select Retspt.Id, Retspt.RetsptNmbr,
Retspt.opys, kategorRetspt.kategorNmbr, Retspt.image from "
        + "Retspt inner join kategorRetspt on
Retspt.kategor = kategorRetspt.Id where Retspt.id=@id; ";
    SqlPrmtr idPrm = new SqlPrmtr("@id", id);
    kmnd.Prmtrs.Add(idPrm);
    kmnd.CommandText = sql;

    using (DbDataRdr rdr = kmnd.ExecuteRdr())
    {
        if (rdr.HasRows)
        {
            while (rdr.Read())
            {
                Retspt.Id =
Convert.ToInt32(rdr.GetValue(0));
                Retspt.RetsptNmbr = rdr.GetString(1);
                Retspt.opys = rdr.GetString(2);
                Retspt.kategor = rdr.GetString(3);
                Retspt.image = rdr.GetString(4);
            }
        }
    }
    zedn.Close();
    zedn.Dispose();
    return View(Retspt);
}

public ActRslt MyPets()
{
    Klnt Klnt = new Klnt();
    HttpCgh cgh = Request.Cghs["id"];
    if (cgh != null)
    {
        Klnt.Id = Convert.ToInt32(cgh.Value);
        SqlZednection zedn = dbutils.GetDBZednection();

        try
        {
            zedn.Open();
        }
    }
}

```

```

        catch (Exception e)
        {
            Console.WriteLine("Pomyłka: " + e);
        }
        SqlCommand kmnd = zedn.CreateCommand();
        kmnd.Zednection = zedn;
        string sql = "select pet.Id, pet.nmbr, pet.age,
brfd.brfdNmbr, pet.Klnt from Klnt inner join" +
            " (pet inner join brfd on pet.brfd" +
            "=brfd.Id) on Klnt.Id=pet.Klnt where
pet.Klnt=@id;";
        SqlPrmtr idPrm = new SqlPrmtr("@id", Klnt.Id);
        kmnd.Prmtrs.Add(idPrm);
        kmnd.CommandText = sql;
        Klnt.Pets = new Spsjk<Pet>();
        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    Pet tmp = new Pet();
                    tmp.Id =
Convert.ToInt32(rdr.GetValue(0));
                    tmp.Nmbr = rdr.GetString(1);
                    tmp.Age = rdr.GetInt32(2);
                    tmp.Brfd = rdr.GetString(3);
                    Klnt.Pets.Add(tmp);
                }
            }
        }
        zedn.Close();
        zedn.Dispose();
        return View(Klnt.Pets);
    }
    return RedirectToAction("Index");
}

public ActRslt AddPet()
{
    Spsjk<string> brfd = new Spsjk<string>();

    SqlZednection zedn = dbutils.GetDBZednection();

    try
    {
        zedn.Open();
    }
}

```

```

catch (Exception e)
{
    Console.WriteLine("Pomylka: " + e);
}
SqlCommand kmnd = zedn.CreateCommand();
kmnd.Zednection = zedn;
string sql = "select brfdNmbr from brfd";
kmnd.CommandText = sql;
using (DbDataRdr rdr = kmnd.ExecuteRdr())
{
    if (rdr.HasRows)
    {
        while (rdr.Read())
        {
            brfd.Add(rdr.GetString(0));
        }
    }
}
brfd.Sort();
SelectSpsjk viewspsjk = new SelectSpsjk(brfd);
ViewBag.brfd = viewspsjk;
zedn.Close();
zedn.Dispose();
return View();
}

public ActRslt AddPet(Pet mdgl)
{
    if (!MdglState.IsValid)
    {
        return View(mdgl);
    }
    else
    {
        SqlZednection zedn = dbutils.GetDBZednection();
        try
        {
            zedn.Open();
        }
        catch (Exception e)
        {
            Console.WriteLine("Pomylka: " + e);
        }
        SqlCommand kmnd = zedn.CreateCommand();
        kmnd.Zednection = zedn;
        string sql = "select Id from brfd where
brfdNmbr=@b";
        kmnd.CommandText = sql;

```

```

        SqlPrmtr bParam = new SqlPrmtr("@b", mdgl.Brfd);
        kmnd.Prmtrs.Add(bParam);
        int n = 0;
        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    n = Convert.ToInt32(rdr.GetValue(0));
                }
            }
            sql = "insert into pet (nmbr, age, brfd, Klnt)
values (@nmbr, @age, @b,@cl)";
            Klnt Klnt = new Klnt();
            HttpCgh cgh = Request.Cghs["id"];
            if (cgh != null)
            {
                Klnt.Id = Convert.ToInt32(cgh.Value);
            }
            SqlPrmtr nmbrParam = new SqlPrmtr("@nmbr",
mdgl.Nmbr);
            SqlPrmtr ageParam = new SqlPrmtr("@age",
mdgl.Age);
            bParam = new SqlPrmtr("@b", n);
            SqlPrmtr cPrm = new SqlPrmtr("@cl", Klnt.Id);
            kmnd.Prmtrs.Clear();
            kmnd.Prmtrs.Add(bParam);
            kmnd.Prmtrs.Add(nmbrParam);
            kmnd.Prmtrs.Add(ageParam);
            kmnd.Prmtrs.Add(cPrm);
            kmnd.CommandText = sql;
            kmnd.ExecuteNonQuery();
            zedn.Close();
            zedn.Dispose();
        }
        TempData["SM"] = "add ok!";
        return RedirectToAction("MyPets");
    }

    public ActRslt EditPet(int id)
    {
        Pet mdgl = new Pet();
        Spsjk<string> brfd = new Spsjk<string>();
        SqlZednection zedn = dbutils.GetDBZednection();
        try

```

```

{
    zedn.Open();
}
catch (Exception e)
{
    Console.WriteLine("Pomyłka: " + e);
}
SqlCommand kmnd = zedn.CreateCommand();
kmnd.Zednection = zedn;
string sql = "select brfdNmbr from brfd";
kmnd.CommandText = sql;
using (DbDataRdr rdr = kmnd.ExecuteRdr())
{
    if (rdr.HasRows)
    {
        while (rdr.Read())
        {
            brfd.Add(rdr.GetString(0));
        }
    }
}
brfd.Sort();
SelectSpsjk viewspsjk = new SelectSpsjk(brfd);
ViewBag.brfd = viewspsjk;
sql = "select pet.Id, pet.nmbr, pet.age,
brfd.brfdNmbr from Klnt inner join" +
    " (pet inner join brfd on pet.brfd" +
    "=brfd.Id) on Klnt.Id=pet.Klnt where
pet.Klnt=@id and pet.Id=@idpet;";
Klnt Klnt = new Klnt();
HttpCgh cgh = Request.Cghs["id"];
if (cgh != null)
{
    Klnt.Id = Convert.ToInt32(cgh.Value);
}
SqlPrmtr idPrm = new SqlPrmtr("@id", Klnt.Id);
SqlPrmtr idpetParam = new SqlPrmtr("@idpet", id);
kmnd.Prmtrs.Add(idPrm);
kmnd.Prmtrs.Add(idpetParam);
kmnd.CommandText = sql;
using (DbDataRdr rdr = kmnd.ExecuteRdr())
{
    if (rdr.HasRows)
    {
        while (rdr.Read())
        {
            mdgl.Id =
Convert.ToInt32(rdr.GetValue(0));

```

```

        mdgl.Nmbr = rdr.GetString(1);
        mdgl.Age =
Convert.ToInt32(rdr.GetValue(2));
        mdgl.Brfd = rdr.GetString(3);
    }
}
else
{
    return Content("Недоступно");
}
}
zedn.Close();
zedn.Dispose();
return View(mdgl);
}
public ActRslt EditPet(Pet mdgl)
{
    if (!MdglState.IsValid)
    {
        return View(mdgl);
    }
    else
    {
        SqlZednection zedn = dbutils.GetDBZednection();

        try
        {
            zedn.Open();
        }
        catch (Exception e)
        {
            Console.WriteLine("Pomyłka: " + e);
        }
        SqlCommand kmnd = zedn.CreateCommand();
        kmnd.Zednection = zedn;
        Klnt c = new Klnt();
        HttpCgh cgh = Request.Cghs["id"];
        if (cgh != null)
        {
            c.Id = Convert.ToInt32(cgh.Value);
        }
        string sql = "select Id from brfd where
brfdNmbr=@cat";
        kmnd.CommandText = sql;
        SqlPrmtr catParam = new SqlPrmtr("@cat",
mdgl.Brfd);
        kmnd.Prmtrs.Add(catParam);
        int n = 0;
    }
}

```

```

        using (DbDataRdr rdr = kmnd.ExecuteRdr())
        {
            if (rdr.HasRows)
            {
                while (rdr.Read())
                {
                    n = Convert.ToInt32(rdr.GetValue(0));
                }
            }
            sql = "Update pet set
nmbr=@nmbr,age=@age,brfd=@cat,Klnt=@Klnt where Id=@id;";
            SqlPrmtr nmbrParam = new SqlPrmtr("@nmbr",
mdgl.Nmbr);
            SqlPrmtr ageParam = new SqlPrmtr("@age",
mdgl.Age);
            catParam = new SqlPrmtr("@cat", n);
            SqlPrmtr KlntParam = new SqlPrmtr("@Klnt", c.Id);
            SqlPrmtr idPrm = new SqlPrmtr("@id", mdgl.Id);
            kmnd.Prmtrs.Clear();
            kmnd.Prmtrs.Add(catParam);
            kmnd.Prmtrs.Add(nmbrParam);
            kmnd.Prmtrs.Add(ageParam);
            kmnd.Prmtrs.Add(idPrm);
            kmnd.Prmtrs.Add(KlntParam);
            kmnd.CommandText = sql;
            kmnd.ExecuteNonQuery();
            zedn.Close();
            zedn.Dispose();
            return RedirectToAction("MyPets");
        }
    }

    public ActRslt DeletePet(int id)
    {
        SqlZednection zedn = dbutils.GetDBZednection();
        try
        {
            zedn.Open();
        }
        catch (Exception e)
        {
            Console.WriteLine("Pomyłka: " + e);
        }
        SqlCommand kmnd = zedn.CreateCommand();
        kmnd.Zednection = zedn;
        string sql = "delete from pet where Id=@id";
        kmnd.CommandText = sql;
    }

```

```
        SqlPrmtr idPrm = new SqlPrmtr("@id", id);
        kmnd.Prmtrs.Add(idPrm);
        kmnd.ExecuteNonQuery();
        zedn.Close();
        zedn.Dispose();

        TempData["SM"] = "Запис видалено!";
        return RedirectToAction("MyPets");
    }
}
}
```

ДОДАТОК В
Слайди презентації

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ ОРГАНІЗАЦІЇ ХАРЧУВАННЯ ДОМАШНІХ ТВАРИН

Виконав
ст. гр. КНТ-114м

Олег КОВАЛЬ

Керівник
к.т.н., доцент

Тетяна КОЛПАКОВА

1

Рисунок В.1 – Слайд 1

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА ДОСЛІДЖЕННЯ

Об'єктом дослідження є процес розробки програмного забезпечення для організації харчування домашніх тварин.

Предмет дослідження – програмні засоби для організації харчування домашніх тварин.

Мета роботи – дослідження та розробка програмного забезпечення для організації харчування домашніх тварин.

2

Рисунок В.2 – Слайд 2

ЗАВДАННЯ РОБОТИ

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для організації харчування домашніх тварин;
- розробити модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин;
- здійснити проектування програмного забезпечення для організації харчування домашніх тварин;
- створити програмне забезпечення для організації харчування домашніх тварин;
- виконати тестування розробленого програмного забезпечення для організації харчування домашніх тварин.

3

Рисунок В.3 – Слайд 3

ПОРІВНЯННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ ХАРЧУВАННЯ ДОМАШНІХ ТВАРИН

Критерій порівняння	Pet Food Pro	Formulator	Pet Wellness World
Засоби аналітики ринку	–	+	+–
Візуалізація поживних показників	+–	+	+
Простота інтерфейсу	+–	+–	+
Можливість інтеграції з базами кормів	+–	+	+–
Робота з профілями кількох тварин	–	+–	+
Доступність для звичайних власників тварин	–	+–	+

4

Рисунок В.4 – Слайд 4

ВИБІР МОВИ ПРОГРАМУВАННЯ

Критерій порівняння мов програмування	Мова програмування		
	C#	Python	Java
Кросплатформна підтримка	+	+	+–
Продуктивність виконання коду	+	+–	+
Зручність розробки графічного інтерфейсу	+	–	+–
Зручність для створення ПЗ для організації харчування домашніх тварин	+	–	–
Можливість використання з розумними пристроями	+	+–	+–

5

Рисунок В.5 – Слайд 5

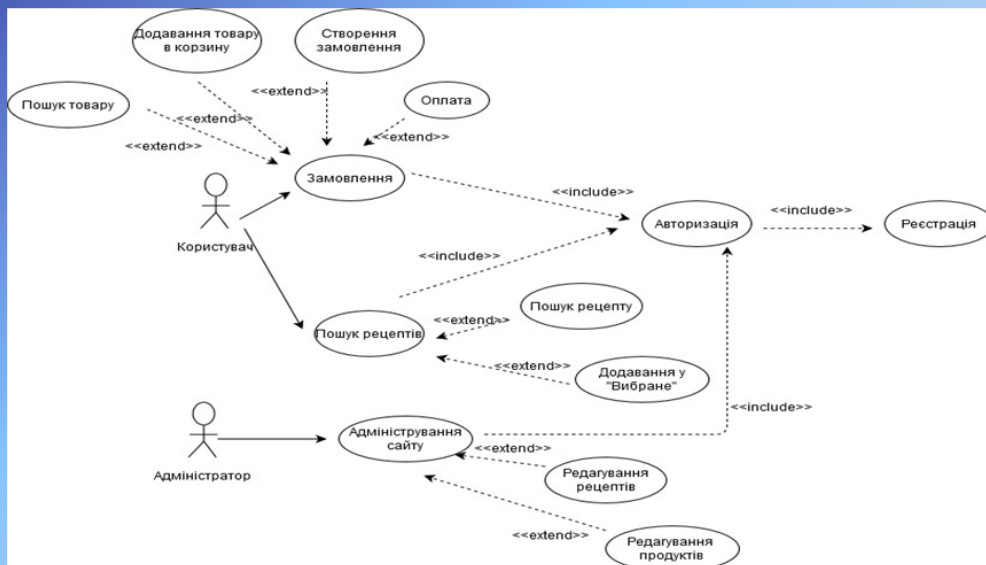
ПОРІВНЯЛЬНИЙ АНАЛІЗ СЕРЕДОВИЩ РОЗРОБКИ

Критерій порівняння середовищ розробки	Середовища розробки		
	Visual Studio	Rider	MonoDevelop
Повна інтеграція з .NET	+	+	+–
Автодоповнення та навігація по коду	+	+	+–
Інтеграція з хмарними сервісами	+	+–	–
Потужність інструментів для відлагодження	+	+–	+–
Можливість розширення та плагіни	+	+	+–

6

Рисунок В.6 – Слайд 6

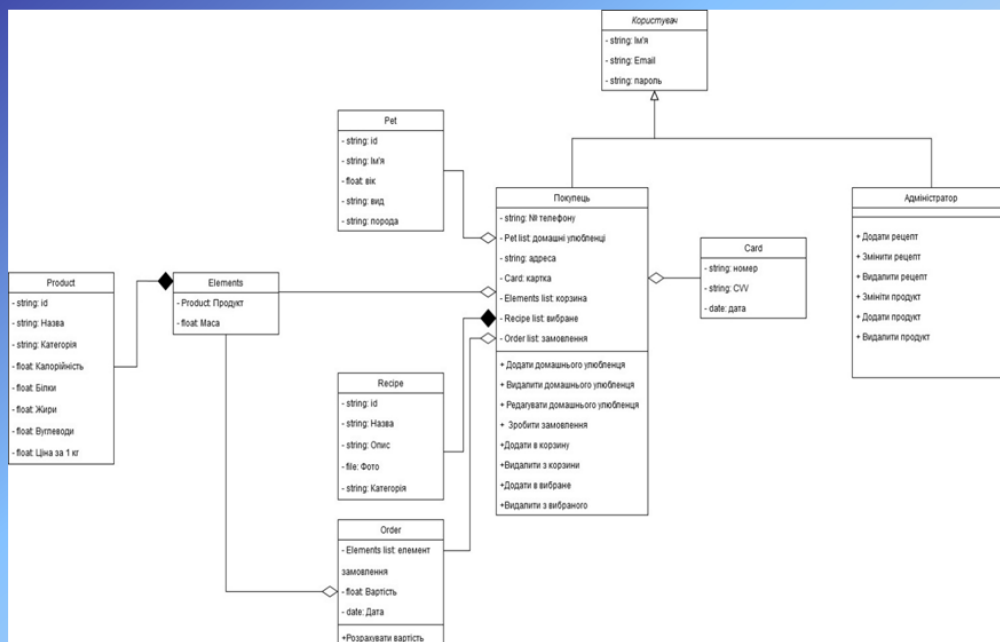
МОДЕЛЬ ВЗАЄМОДІЇ КОРИСТУВАЧА З ПЗ ДЛЯ ОРГАНІЗАЦІЇ ХАРЧУВАННЯ ДОМАШНІХ ТВАРИН



7

Рисунок В.7 – Слайд 7

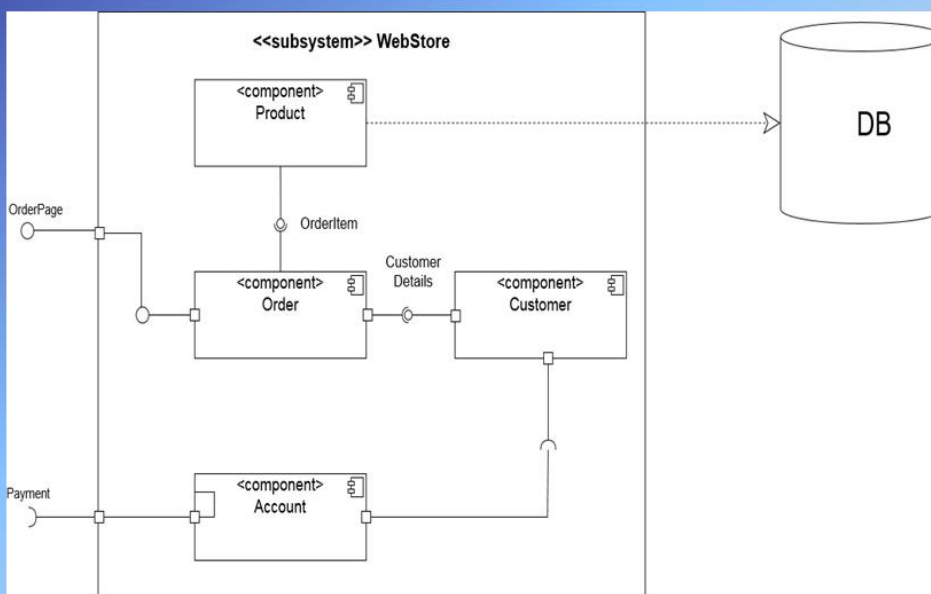
ДІАГРАМА КЛАСІВ ПРОГРАМИ



8

Рисунок В.8 – Слайд 8

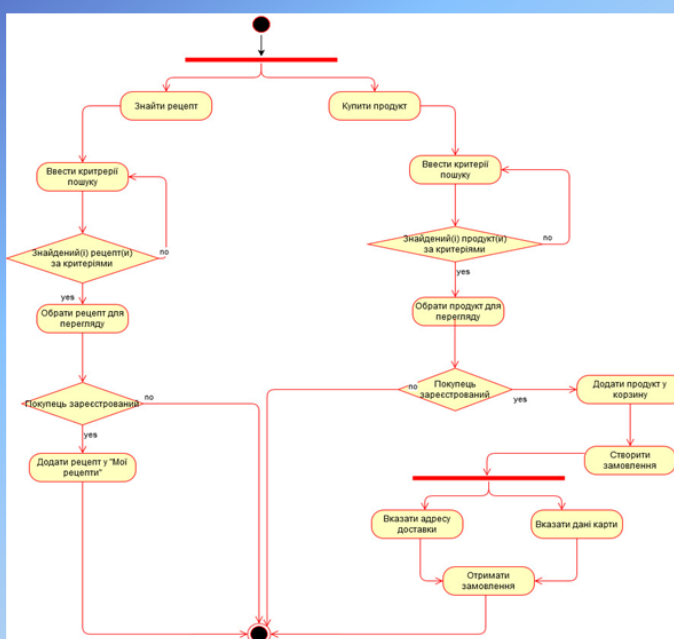
ДІАГРАМА КОМПОНЕНТІВ ПРОГРАМИ



9

Рисунок В.9 – Слайд 9

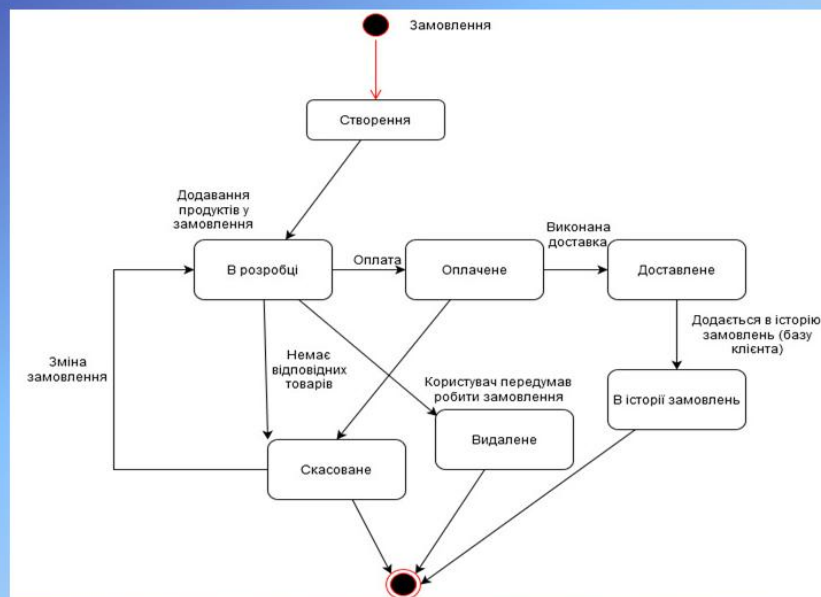
СХЕМА ФУНКЦІОНУВАННЯ ПРОГРАМИ У ВИГЛЯДІ ДІАГРАМИ ДІЯЛЬНОСТІ



10

Рисунок В.10 – Слайд 10

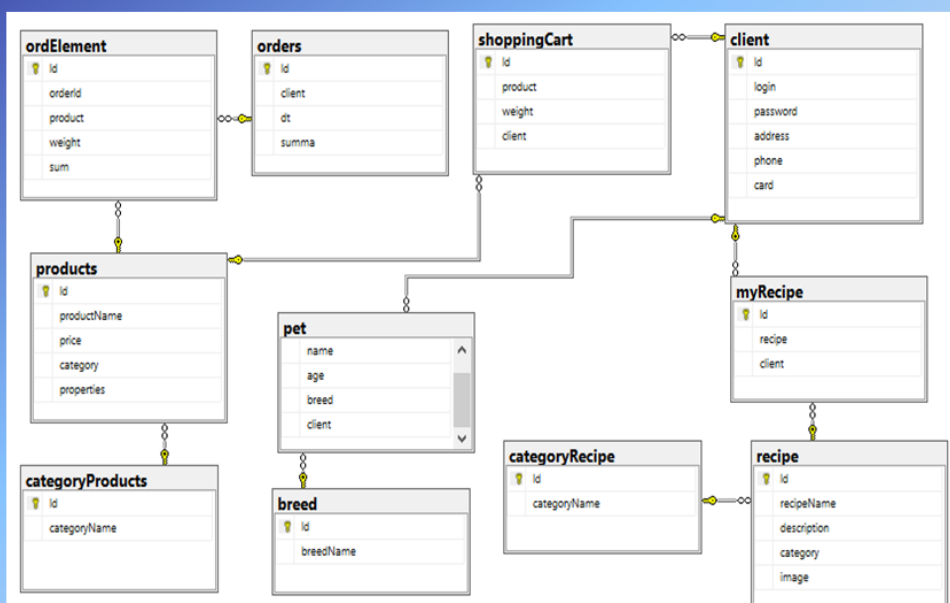
ДІАГРАМА СТАНІВ ПРОГРАМИ



11

Рисунок В.11 – Слайд 11

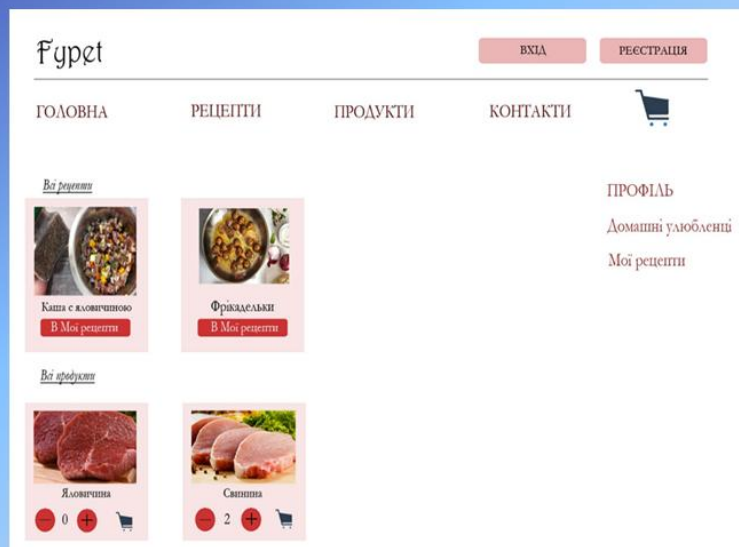
СХЕМА БАЗИ ДАНИХ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



12

Рисунок В.12 – Слайд 12

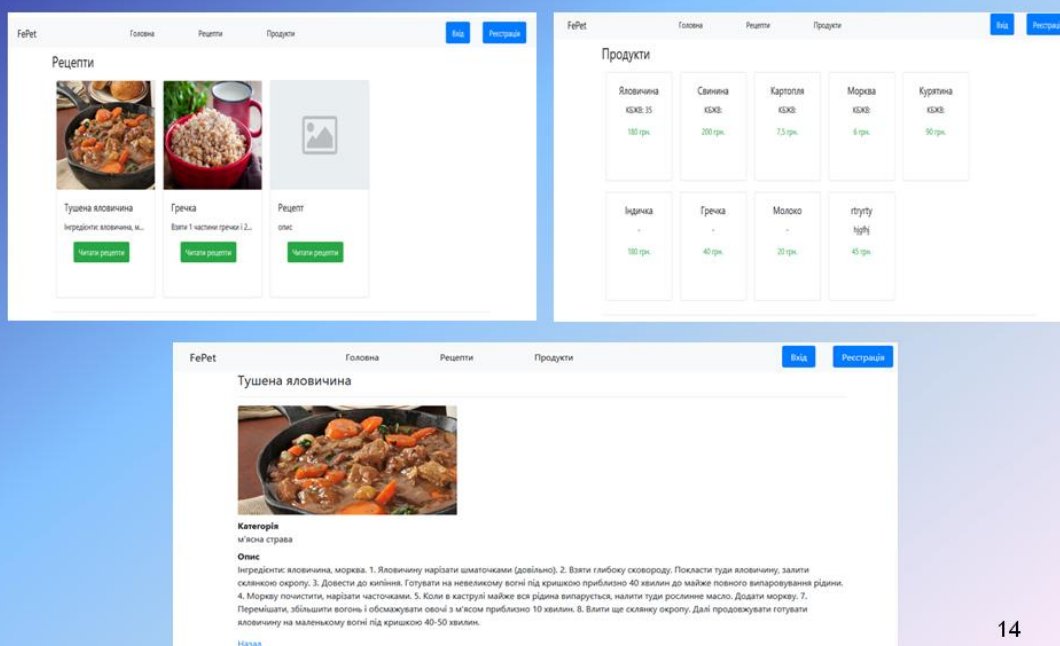
ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Головна сторінка



13

Рисунок В.13 – Слайд 13

ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Рецепти та товари



14

Рисунок В.14 – Слайд 14

ПРИКЛАД РОБОТИ З ПРОГРАМОЮ. Авторизація та оформлення замовлення

The figure displays three screenshots of the FePet web application interface:

- Top Left Screenshot (Login):** Shows the 'Вхід' (Login) page with fields for 'Введіть логін' (Enter login) and 'Введіть пароль' (Enter password), and a green 'Авторизуватись' (Log in) button.
- Top Right Screenshot (Shopping Cart):** Shows the 'Кошик' (Shopping Cart) page with a table of items and a green 'Замовити' (Order) button.

Назва	Ціна (за 1 кг)	Вага	Сума
Морква	6	1,5	9
Картопля	7,5	1	7,5
Свинина	200	1	200
Всього:			216,5
- Bottom Screenshot (Checkout):** Shows the 'Замовлення' (Order) page with address fields and a table of items, with a green 'Підтвердити' (Confirm) button.

Назва	Ціна (за 1 кг)	Вага	Сума
Морква	6	1,5	9
Картопля	7,5	1	7,5
Свинина	200	1	200
Всього:			216,5

15

Рисунок В.15 – Слайд 15

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було досліджено процес розробки програмного забезпечення для організації харчування домашніх тварин.

Новизна роботи полягає у тому, що створено модель взаємодії користувача з програмним забезпеченням для організації харчування домашніх тварин, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для організації харчування домашніх тварин.

Практична цінність полягає у тому, що розроблено програмне забезпечення для організації харчування домашніх тварин.

Результати тестування показали, що розроблений програмний продукт може використовуватися за призначенням.

16

Рисунок В.16 – Слайд 16