

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ РОЗШУКУ ЗАГУБЛЕНИХ ТВАРИН
LOST ANIMAL SEARCH SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-219сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

Чумак М.В.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Казурова А.Є.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Чумака Микити Вікторовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення розшуку загублених тварин
Lost Animal Search Software

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 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. Розробка архітектури програми.
3. Розробка програми. 4. Експлуатація та тестування програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

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

Студент(ка)

(підпис)

Чумак М.В.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 82 с., 26 рис., 2 табл., 3 дод., 13 джерел.

ЗАГУБЛЕНА ТВАРИНА, ДОМАШНЯ ТВАРИНА, РОЗШУК, ВЕБЗАСТОСУНОК, PYTHON, DJANGO.

Об'єкт роботи – процес розшуку загублених тварин. Предмет роботи – програмне забезпечення розшуку загублених тварин. Мета роботи – розробити програмне забезпечення розшуку загублених тварин для сприяння возз'єднанню загублених тварин зі своїми власниками.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мови Python, JavaScript, HTML, CSS, вебфреймворк Django, система керування базами даних MySQL.

Результати. Основним результатом роботи є розроблена програма, у процесі чого було виконано аналіз проблеми з визначенням вимог до програми, вибором засобів розробки, проектуванням бази даних, принципів структури програми і логіки роботи програми, а також описано основні прийняті рішення щодо реалізації програми і принципів роботи з її інтерфейсом користувачів.

Висновки. Розроблене програмне забезпечення дозволяє виконувати пошук загублених тварин власниками і пошук власників знайдених тварин, контактувати цим сторонам між собою, третім сторонам повідомляти про відповідність таких оголошень між собою, а також повідомляти власникам тварин про випадки, коли учасник програми бачив тварину після її загублення.

Галузь використання. Програмне забезпечення може використовуватися для взаємодії людей, які є власниками тварин і загубили їх, і з людьми, які їх знайшли, а також з притулками тварин.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 82 pages, 26 figures, 2 tables, 3 appendixes, 13 sources.

LOST ANIMAL, PET, SEARCH, WEBSITE, PYTHON, DJANGO.

The object of the work is the process of searching for lost animals. The subject of work is a software for searching for lost animals. The purpose of the work is to develop software for the search for lost animals to facilitate the reunification of lost animals with their owners.

Materials, methods and technical means: object-oriented programming, Python, JavaScript, HTML languages, CSS, Django web framework, MySQL database management system.

Results. The main result of the work is the developed program. During the development process the analysis of the problem with the definition of program requirements, the choice of development tools, the database design, the design of program structure and logic of the program were performed together with the main decisions made on program implementation and principles of interaction with users.

Conclusions. The developed software allows owners to search for lost animals and other participants to find owners of found animals, to contact these parties, inform third parties about the compliance of such ads with each other, as well as inform owners about cases when a user saw an animal after its loss.

Field of use. The software should be used to interact with people who have owned and lost animals and with people who have found them, as well as with animal shelters.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Огляд програмного забезпечення розшуку загублених тварин	10
1.2 Порівняння аналогів	14
1.3 Висновки за розділом 1	16
2 Розробка архітектури програми.....	17
2.1 Вибір засобів проєктування і розробки	17
2.2 Визначення функціональних вимог до програмного забезпечення	18
2.3 Визначення структури бази даних	19
2.4 Визначення архітектури програми	21
2.5 Висновки за розділом 2	23
3 Розробка програми	24
3.1 Логіка функціонування програми	24
3.2 Опис реалізованих одиниць вебзастосунку.....	26
3.3 Висновки за розділом 3	40
4 Експлуатація та тестування програми	41
4.1 Опис програми.....	41
4.2 Експлуатація програми.....	41
4.3 Тестування програми.....	47
4.4 Висновки за розділом 4	47
Висновки	48
Перелік джерел посилання	49
Додаток А Технічне завдання	51
Додаток Б Текст програми	55
Додаток В Слайди презентації.....	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Проблема зі зниклими домашніми тваринами в Сполучених Штатах Америки загальноновизнана, але фактичну статистику важко отримати, оскільки не існує національної структури звітності, адже агенції з догляду та контролю за тваринами не зобов'язані вести статистику кількості тварин, які були взяті, вилучені тощо, при цьому можна загалом виділити наступне:

- за даними Американської асоціації гуманності, 1 з 3 домашніх тварин губляться в певний момент свого життя, і близько 10 мільйонів собак і кішок губляться або викрадаються в Сполучених Штатах Америки щороку;

- за даними Коаліції за возз'єднання домашніх тварин і сімей менше 23 % втрачених домашніх тварин у Сполучених Штатах возз'єднуються зі своїми власниками;

- з 1000 притулків, які відповіли на опитування Національної ради з вивчення популяції домашніх тварин і політики – найточнішого вимірювача фактів і цифр щодо евтаназії домашніх тварин у Сполучених Штатах Америки, створеного Американською асоціацією гуманності, було встановлено, що ними було оброблено 4,3 мільйони тварин і приблизно 64 % від загальної кількості тварин, які потрапили в ці притулки, були еванізовані, тобто приблизно 2,7 мільйони тварин лише в цих 1000 притулках, при цьому в Сполучених Штатах Америки приблизно 3500 притулків для тварин, тобто з цих цифр можна визначити приблизний рівень евтаназії;

- одна третина всіх собак і котів у Сполучених Штатах вважаються зниклими безвісти за життя, більше 80 % ніколи не знаходять, а від 9 450 000 до 9 632 000 домашніх тварин, які потрапляють у притулки в Сполучених Штатах Америки, гинуть [1].

Усе це говорить про існування великої проблеми у всьому світі, що проілюстровано на прикладі однієї з найбільших і найрозвинутіших країни у світі. Ця ж проблема існує і в Україні, а через війну на території країни ця

проблема додатково посилилась. Звичайно, що в сучасному світі будь-яка велика проблема потребує певної підтримки за допомогою програмного забезпечення (ПЗ). У процесі розшуку загублених тварин важливість ПЗ є ще більшою, оскільки саме від якості програми хоч частково, але залежить успішність пошуку загублених тварин.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд програмного забезпечення розшуку загублених тварин

Першим етапом огляду ПЗ розшуку загублених тварин був аналіз таких можливостей в Україні. Цей огляд встановив, що основним інструментом на даний момент в Україні є соціальні мережі, через які публікуються відповідні оголошення (рис. 1.1) [2]. При цьому користувачі намагаються скористатися розширеною базою користувачів, доступною на платформі, але при цьому немає ніяких зручних засобів пошуку серед таких оголошень. Окрім того оголошення фрагментовані між великою кількістю спільнот у таких соціальних мережах.

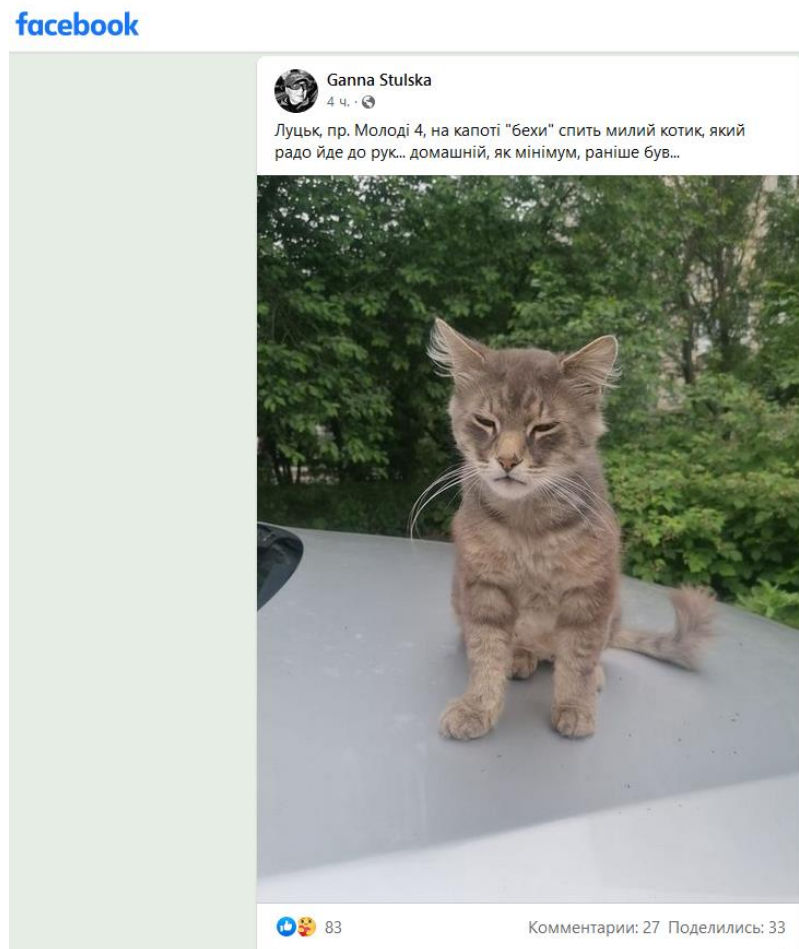


Рисунок 1.1 – Розшук тварин через соціальну мережу Facebook

Однак у світі є і спеціалізовані інструменти, направлені саме на розшук загублених тварин. Найбільш відповідною програмою для цього випадку є Petco LoveLost (рис. 1.2) [3].

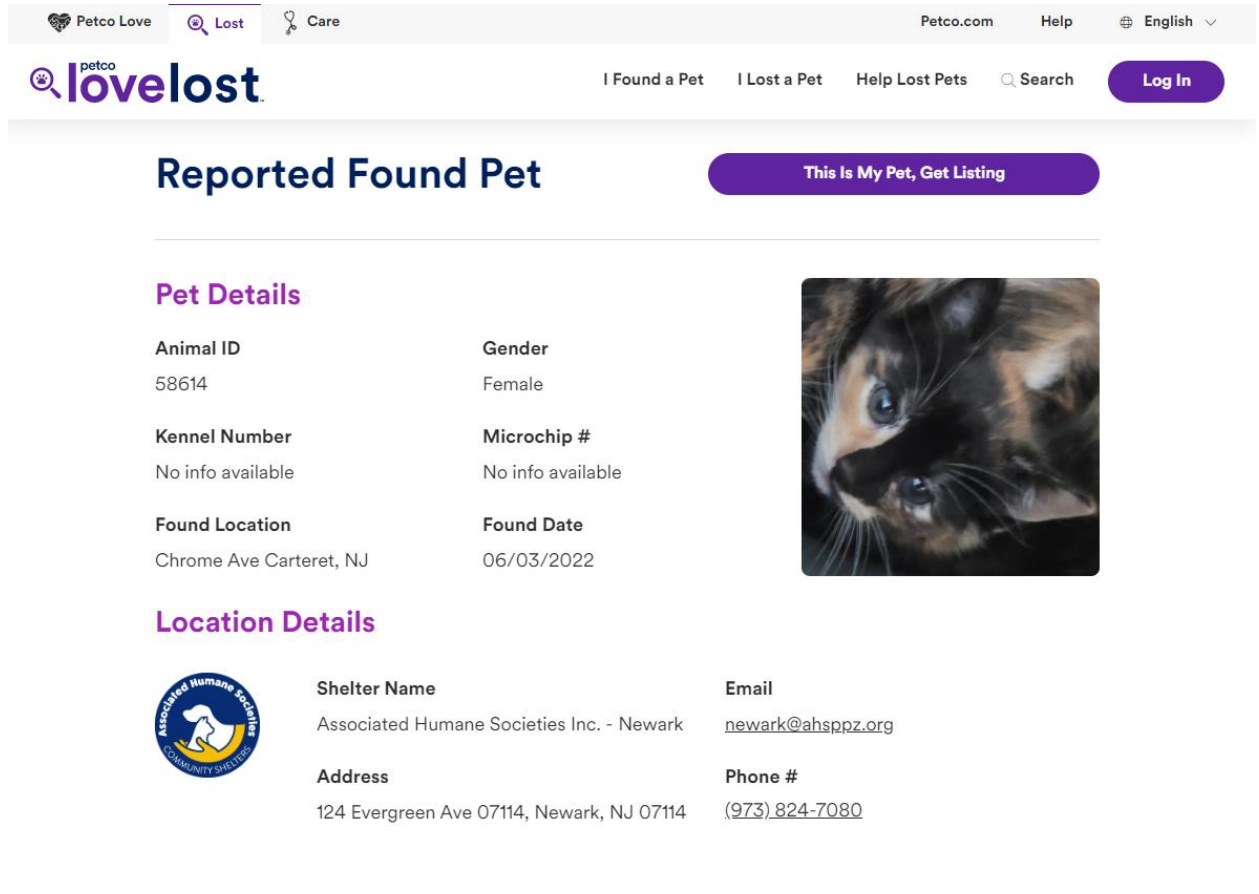


Рисунок 1.2 – Petco LoveLost

Програма надає можливість опублікувати власне оголошення про загублену тварину, про знайдену тварину. Також програма надає можливість виконати пошук серед загублених і знайдених тварин. Ці два типи оголошень розділені окремо. Таким чином користувач виходить з того, що є його проблемою: загублена власна тварина чи знайдена тварина. Надається також можливість повідомити про те, що знайдена тварина належить користувачу, який знайшов це оголошення. Цю програму можна вважати платформою, яка в основному відповідає ідеї, яка переслідується в дипломній роботі.

Однак ця програма не є єдиною, яку так чи інакше можна застосовувати для розшуку тварин.

Існують приклади окремих вебсайтів притулків або міських сайтів, які дозволяють виконувати пошук серед знайдених тварин. Наприклад, вебсайт міста Мілдьюра в Австралії має подібну функціональність (рис. 1.3) [4].

Lost animals

This register contains details about animals currently at the Mildura Animal Pound. For more information about collecting your lost pet or rehoming an animal from the pound contact us on the details below.

Conditions of Entry

Must be wearing enclosed footwear, hearing protection, supervise children at all times and adhere to current COVID-19 restrictions including wearing a face mask and social distancing.

Contact Information & Opening Hours



Animals currently in the pound

12 Result(s) Found

CRM 10899/2022

Found on 02 June 2022

Council Reference Number : CRM 10899/2022

Location Found : etiwanda avenue

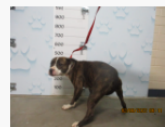
Breed : Staff x

Colour : Brindle and white

Sex : Male

Desexed : No

Tagged as : Dog



> [Click to enlarge](#)

Search Lost Animals

By Phrase or Keyword

By Animal Type

Search

CRM 10771/2022

Found on 01 June 2022

Council Reference Number : CRM 10771/2022

Location Found : Benetook ave

Breed : Jack Russell x

Colour : WHITE

Sex : Female



> [Click to enlarge](#)



Рисунок 1.3 – Інформація про знайдених тварин на міському сайті міста Мілдьюра

На вебсайті розташовано набір оголошень, які інформують про наявних знайдених тварин. За кожною твариною доступна її фотографія, дата, коли її було знайдено, номер, місце, де тварину було знайдено, порода, колір та стать. Пошук є достатньо обмеженим, адже він відбувається з визначенням тільки самої тварини та з зазначенням ключових слів (тобто

фактично стандартний пошук за текстом без наявності окремих інструментів для пошуку, що відповідають розшуку саме тварин). Таким чином, можна говорити про обмеженість функціональних можливостей таких варіантів роботи над розшуком тварин.

Ще одним прикладом такого інструменту є вебсайт одного з благодійних фондів – RSPCA Victoria (рис. 1.4) [5].

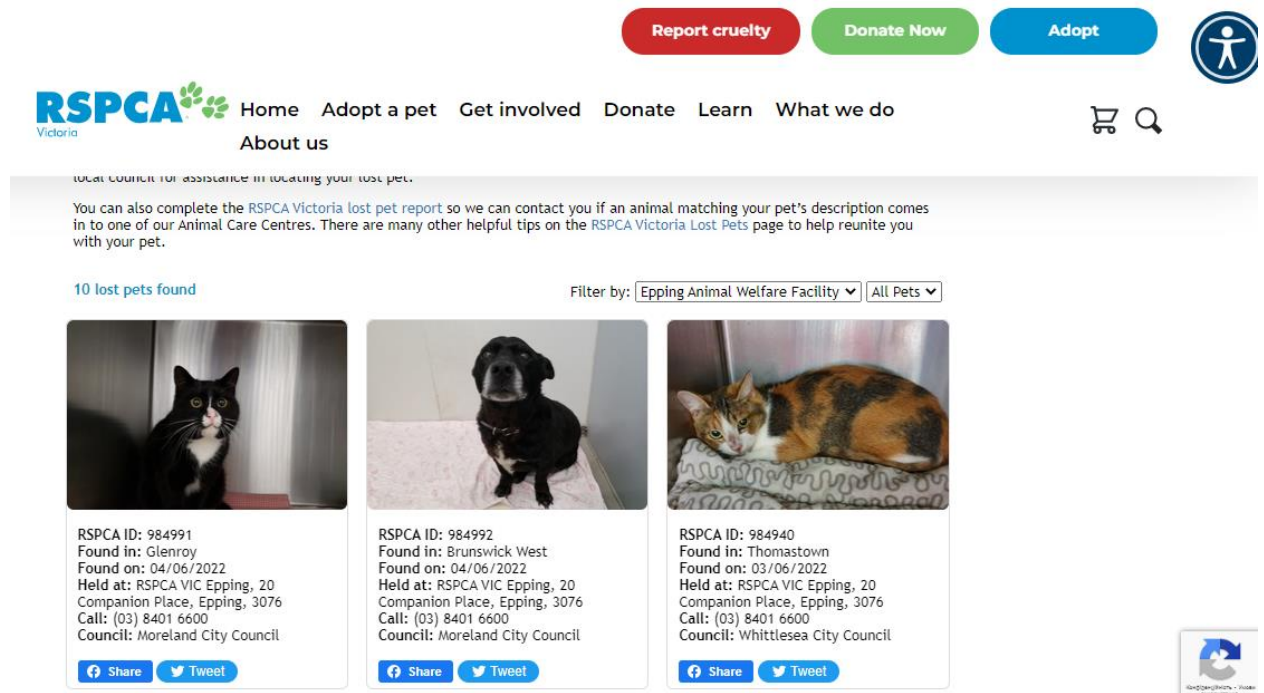


Рисунок 1.4 – RSPCA Victoria

У даному випадку інформація про тварин подається за декількома містами. Але набір інструментів є достатньо подібним на попередній. Тут також надається список всіх знайдених тварин, що знаходяться в притулку. За кожною твариною надається її номер, місце, де вона була знайдена, притулок, де вона знаходиться, номер телефону, фотографія. Засоби пошуку включають тільки засоби фільтрування оголошень і певного їх впорядкування.

1.2 Порівняння аналогів

Для порівняння аналогів було виділено спочатку набір програм, які були проаналізовані в підрозділі 1.1, не включаючи соціальні мережі через їх неспеціалізованість у проблемі. Це вебсайт благодійного фонду, Petco LoveLost і програма, що має бути створена в дипломній роботі. Результати порівняння представлені в табл. 1.1.

Таблиця 1.1 – Порівняння аналогів

Функція	RSPCA Victoria	Petco LoveLost	LostAnimal SearchSoftware
Пошук загублених тварин	Відсутній	Присутній	Присутній
Пошук власників знайдених тварин	Присутній	Присутній	Присутній
Внесення даних загубленої тварини	Відсутній	Присутній	Присутній
Внесення даних знайденої тварини	Відсутній	Присутній	Присутній
Інформування про власництво тварини	Відсутній	Присутній	Присутній
Повідомлення про відповідність між оголошеннями	Відсутній	Відсутній	Присутній
Повідомлення про тимчасову зустріч із твариною	Відсутній	Відсутній	Присутній
Знаходження автоматичної відповідності між оголошеннями	Відсутній	Присутній	Присутній

Під час порівняння ПЗ було використано наступні критерії:

- пошук загублених тварин – це функція, яка є важливою для власників, адже вона дозволяє не тільки переглядати виставлені оголошення про знайдені тварини, але і потенційно створити майданчик для комунікації людей, які загубили тварин, з людьми, які цих тварин знайшли;

- пошук власників знайдених тварин – це функція присутня фактично у всіх подібних програмах і навіть дошках оголошень у соціальних мережах, адже зазвичай людина знаходить тварину і намагається встановити її власника, щоб владнати її долю;

- внесення даних загубленої тварини: важливо, щоб користувач між не тільки переглядати дані, але і бути активним учасником процесу;

- внесення даних знайденої тварини: даний критерій важливий за тією ж логікою, що і попередній;

- інформування про власництво тварини: це важливий інструмент для того, щоб встановити контакт між сторонами;

- повідомлення про відповідність між оголошеннями: пошук є важливим, але і повідомлення від користувачів, які вручну знайшли відповідність між оголошеннями, є також важливими, адже це фактично створює спільноту ентузіастів;

- повідомлення про тимчасову зустріч із твариною: це важливо для того, щоб провести розслідування, якщо власник готовий на більш тривалий процес пошуку, тоді інформація від тих, хто бачив тварину випадково, може бути корисною;

- знаходження автоматичної відповідності між оголошеннями.

Програма, яка має бути розроблена, має ряд переваг над програмами, що існують на даний момент. Це дає їй переваги щодо самого розшуку, дозволяючи потенційно збільшити ефективність розшуку.

1.3 Висновки за розділом 1

Виконано огляд основних аналогів ПЗ розшуку тварин, включаючи спеціалізовані інструменти, вебсайти муніципалітетів, благодійних фондів. Виконано порівняння цих програм з програмою, яку заплановано розробити в дипломній роботі.

Завданнями роботи для сукупного досягнення мети є:

- аналіз програмних аналогів;
- визначення засобів проєктування і розробки програми;
- проєктування програми;
- реалізація і налагодження програми.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Мова Python [6]-[8] є широко використовуваною мовою програмування з призначенням, яке є загальним, оскільки вона дозволяє виконувати і веброзробку, що є критичним у даному випадку, дозволяє застосовувати штучний інтелект, що важливо у випадку подальшого розвитку вебзастосунку з наданням можливостей пошуку відповідностей між оголошеннями за фотографіями, дозволяє створювати звичайні застосунки для Windows. Усе це вказує на більш широкі перспективи подальшого розвитку створеного ПЗ. Тому вона стала засобом розробки ПЗ дипломної роботи.

У табл. 2.1 зведено принципи порівняння мов програмування Python і PHP.

Таблиця 2.1 – Вибір мови програмування

Показник	Python	PHP
Тип мови	Мова загального призначення	Мова для веброзробки
Використання	Веброзробка, штучний інтелект, застосунки для стаціонарних систем	Веброзробка
Попередньо збудовані бібліотеки широкого спрямування	Велика кількість	Набагато менше
Підтримка основних баз даних	Так	Так
Читабельність	Краща	Гірша

2.2 Визначення функціональних вимог до програмного забезпечення

ПЗ повинно надавати засоби роботи з наступними функціями:

- пошук загубленої тварини;
- пошук за знайденою твариною;
- внесення даних про загублену тварину;
- внесення даних про знайдену тварину;
- визначення знайденої тварини власником;
- повідомлення про тварину;
- відправлення повідомлення учаснику;
- перегляд повідомлень між учасниками;
- повідомлення про подібність між загубленою та знайденою тваринами;
- зв'язування учасників з притулком;
- перегляд розшукуваних тварин у притулку, заявок власників;
- перегляд останніх знайдених тварин;
- перегляд останніх загублених тварин;
- перегляд власних знайдених або загублених тварин або за заданим учасником;
- перегляд власних внесених заявок на власність, повідомлень про подібність або відслідковувань;
- перегляд адресованих користувачу заявок на власність, повідомлень про подібність або відслідковувань;
- авторизація і реєстрація в програмі.

Користуючись даним списком, було виконано моделювання сценаріїв роботи учасника програми розшуку з ПЗ (рис. 2.1). Учасником програми розшуку є будь-який користувач, який звернувся до програми.

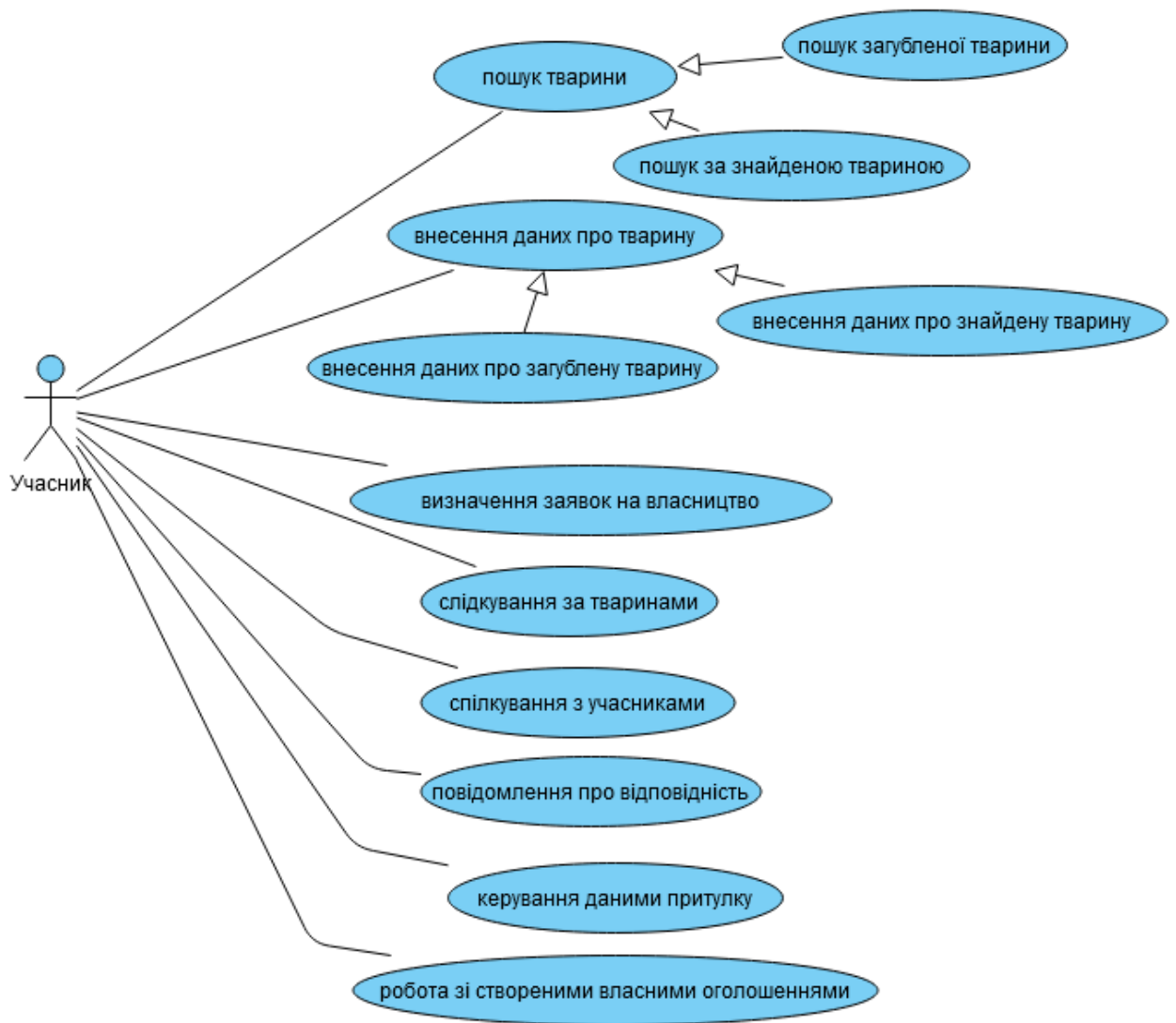


Рисунок 2.1 – Діаграма прецедентів

2.3 Визначення структури бази даних

Під час визначення структури бази даних (БД) використано систему керування MySQL. З застосуванням особливостей програми і сценаріїв роботи учасників програми розшуку виконано виділення таблиця БД, що включають наступні:

- таблиця Animal представляє тварину (наприклад, собака, кіт, хом'як тощо):
- таблиця Breed представляє конкретну породу тварини (перський кіт, наприклад);

- таблиця Lost представляє інформацію про загублену або знайдену тварину, це фактично оголошення про розшук власника тварини або про розшук своєї тварини власником;
- таблиця LostCharvalues представляє значення характеристик загубленої або знайденої тварини (наприклад, конкретний колір шерсті, колір очей, довжина хвоста тощо);
- таблиця Participant представляє учасника програми розшуку загублених тварин, тобто це фактично користувач програми;
- таблиця Characteristic представляє характеристики, якими можуть описуватися знайдені або загублені тварини (наприклад, характеристика кольору шерсті, кольору очей);
- таблиця Charvalue представляє можливі значення характеристики, якою описується тварина (наприклад, рудий, чорний, білий, карі);
- таблиця Photo представляє фотографію загубленої або знайденої тварини, тобто це фотографія, яка використовується в оголошенні;
- таблиця Application представляє заявку на те, що загублена або знайдена тварина була розпізнана;
- таблиця Meeting представляє інформацію про зустріч з загубленою твариною, це називається в програмі також слідкуванням за твариною, коли хтось випадково побачив тварину, яку загубив її власник;
- таблиця Message представляє повідомлення від одного учасника до іншого;
- таблиця Kennel представляє притулок для тварин, в якому може знаходитися тварина після її знаходження;
- таблиця City представляє місто;
- таблиця Knock представляє інформацію від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину.

Схема створеної БД представлена на рис. 2.2.

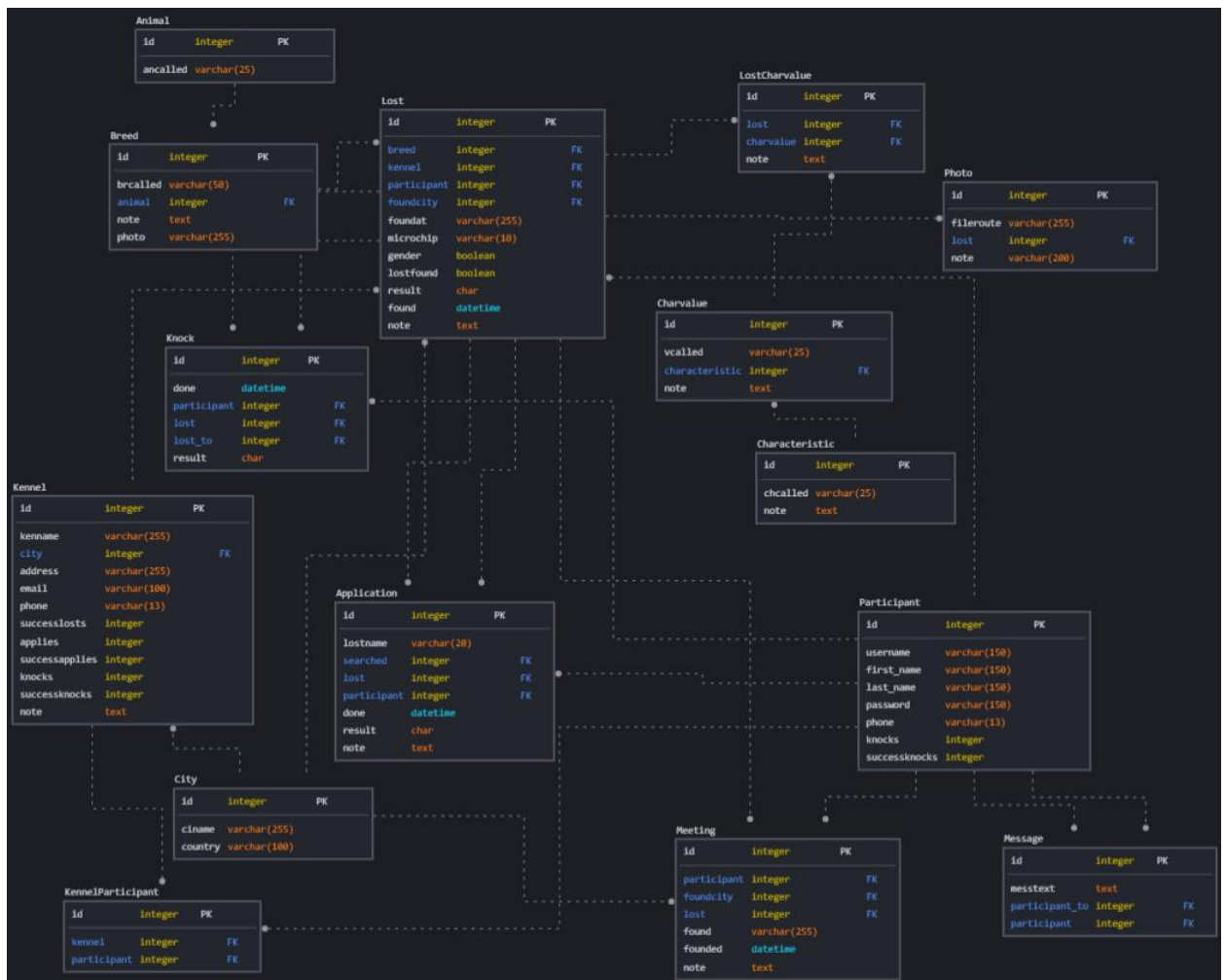


Рисунок 2.2 – Схема бази даних

2.4 Визначення архітектури програми

Архітектуру програми було вирішено побудувати на основі багатоварової моделі. Таке рішення сприяє повторному використанню створеного коду, розширенню системи, що дозволяє розглядати розробку, що виконується, як центральну складову більш широкої системи.

Для побудови архітектури ПЗ було вирішено використовувати шаблон проєктування Model-View-Template (рис. 2.3), який лежить в основі вебфреймворку Django, що було вирішено використовувати для побудови вебзастосунку.

Model-View-Template – це шаблон проєктування, який передбачає три основні складові:

- модель, яка допомагає працювати з БД: це рівень доступу до даних, який обробляє дані;
- шаблон – це рівень презентації, який повністю обробляє частину інтерфейсу користувача;
- представлення використовується для виконання бізнес-логіки та взаємодії з моделлю для перенесення даних і візуалізації шаблону: користувач запитує ресурс до Django, Django працює як контролер і перевіряє доступний ресурс в URL, а далі URL-мапи визивають представлення, яке взаємодіє з моделлю та шаблоном, воно відтворює шаблон [9]-[12].

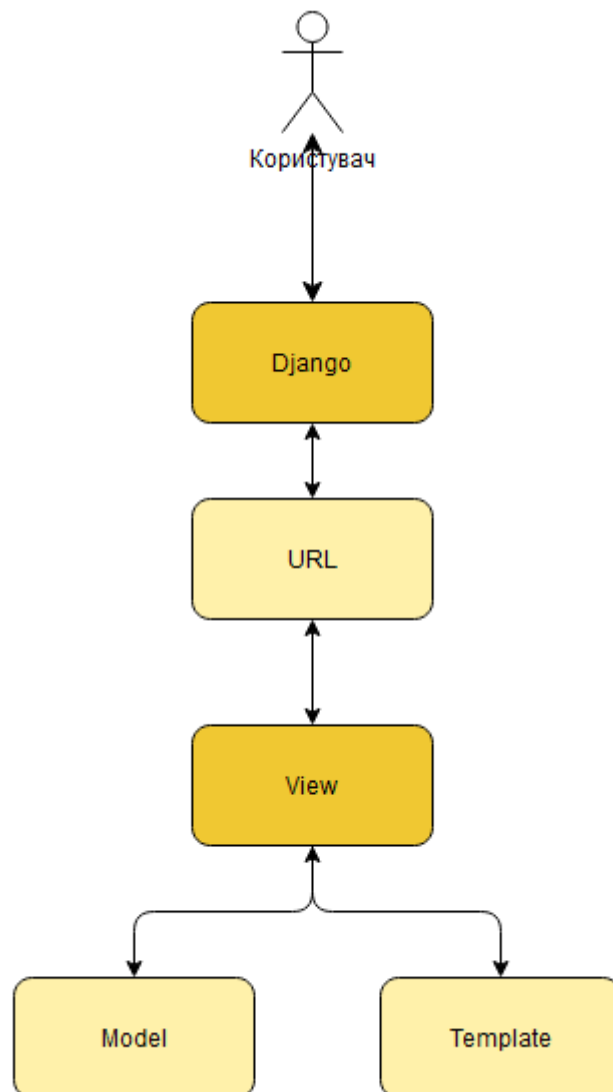


Рисунок 2.3 – Шаблон проєктування Model-View-Template

2.5 Висновки за розділом 2

У другому розділі виконано вибір засобів, які визначали проєктування і розробку програми, включаючи мову Python та вебфреймворк Django зокрема, промодельовано сценарії роботи учасника програми розшуку тварин з програмою, виконано проєктування БД, обрано шаблон проєктування ПЗ.

3 РОЗРОБКА ПРОГРАМИ

3.1 Логіка функціонування програми

Центральним елементом, з яким взаємодіють учасники програми, звичайно є оголошення про загублену або знайдену тварину. При чому в ПЗ може знаходитися одночасно два оголошення про одну й ту саму тварину. Задача програми та її учасників у такому випадку – встановити між ними відповідність і возз'єднати власників з твариною. Для вираження цієї відповідності використовується сутність заявки, яка визначає, що власник вважає певну тварину за оголошенням своєю. Окрім того ця відповідність також може виражатися через ще одну сутність, яка відповідає знахідці третьої сторони, сторонньої особи, учасника програми. Ця сутність – повідомлення про відповідність.

Саме оголошення може перебувати в трьох станах загалом (рис. 3.1). Першим станом є стан створений. Він виникає тоді, коли учасник створює оголошення.

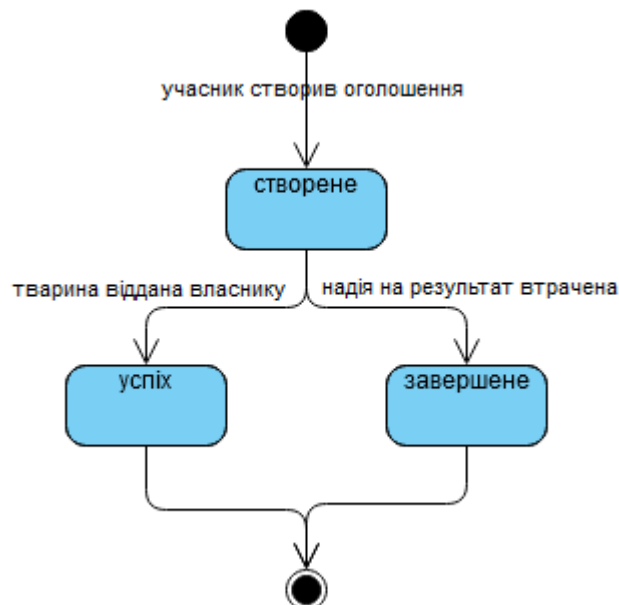


Рисунок 3.1 – Діаграма станів оголошення

У стані створеного оголошення може перебувати як завгодно довго, допоки в крайньому випадку автор оголошення не втратить надію на результат. У такому випадку оголошення буде перебувати у стані завершеному. Перервати цей перехід може ситуація, коли тварина була віддана власнику. Цьому випадку може відповідати зміна станів повідомлення або заявки. Тоді за результатами даних процесів оголошення переходить до стану успіху.

Відповідно для зміни стану оголошення заявка або повідомлення мають пройти свій розвиток. Для цього спочатку учасник програми знаходить власну тварину як загублену і повідомляє про цю знахідку автора оголошення. Тоді заявка або повідомлення переходять до стану створених (рис. 3.2).

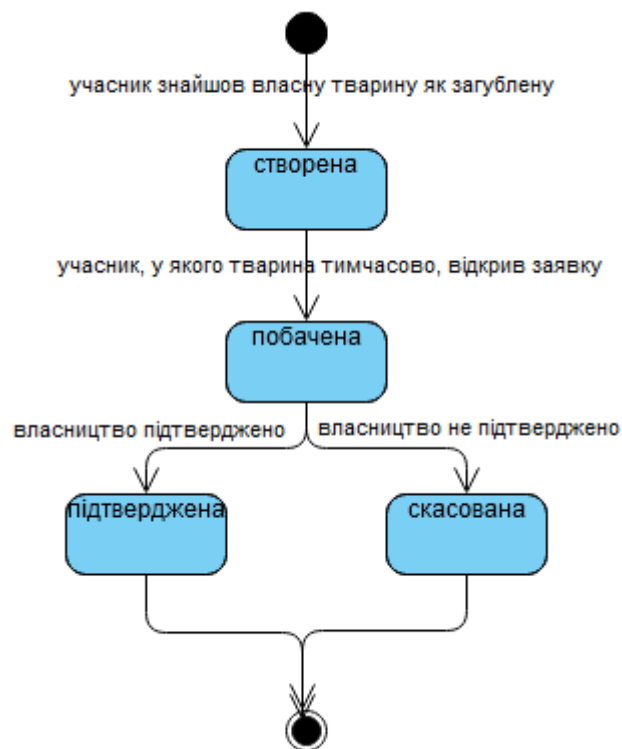


Рисунок 3.2 – Діаграма станів заявки

Після того учасник, у якого тварина знаходиться тимчасово, має відкрити заявку або одна зі сторін має відкрити повідомлення. Тоді заявка переходить до стану побаченої. Далі учасник розглядає відповідність

власництва, зустрічається з заявником або з іншою стороною у випадку повідомлення, і якщо це виявилось правдою, то заявка або повідомлення перейдуть до стану підтверджених. Якщо ж ні, то заявка або повідомлення будуть скасованими.

3.2 Опис реалізованих одиниць вебзастосунку

Модель `Animal` представляє тварину, використовуючи поле `ancalled` – назву тварини, що є символьним полем з максимальною кількістю символів – 25: `models.CharField(max_length=25)`.

Модель `Breed` представляє конкретну породу тварини, використовуючи наступний набір полів:

- `brcalled` – назва породи, що є символьним полем з максимальною кількістю символів – 50: `models.CharField(max_length=50)`;

- `note` – опис породи тварини, що є текстовим полем: `models.TextField()`;

- `animal` – тварина, порода якої подається, є зовнішнім ключем з моделі `Animal`: `models.ForeignKey(Animal, related_name='animalbreeds', on_delete=models.CASCADE)`;

- `photo` – фотографія даної породи, що є файловим полем: `models.FileField(upload_to='img/breeds', max_length=255)`.

Модель `Lost` представляє інформацію про загублену або знайдену тварину, використовуючи наступний набір полів:

- `pote` – опис загубленої або знайденої тварини, що включає додаткову інформацію окрім іншої, що її описує, є текстовим полем: `models.TextField()`;

- `found` – час, коли було знайдено загублену тварину або коли приблизно було її загублено, є полем дати і часу: `models.DateTimeField()`;

- `breed` – порода тварини, яку було знайдено або загублено, є зовнішнім ключем з моделі `Breed`: `models.ForeignKey(Breed, related_name='breedlosts', on_delete=models.CASCADE)`;

– `foundcity` – місто, де було знайдено або загублено тварину, є зовнішнім ключем з моделі `City`: `models.ForeignKey(City, related_name='citylosts', on_delete=models.CASCADE);`

– `foundat` – адреса місця, де було знайдено або де було приблизно загублено тварину, є символьним полем з максимальною довжиною в 255 символів: `models.CharField(max_length=255);`

– `microchip` – номер мікрочіпу знайденої тварини, є символьним полем з максимальною довжиною 10: `models.CharField(max_length=10);`

– `kennel` – притулок, в якому знаходиться знайдена тварина (для випадків, коли вона знайдена), є зовнішнім ключем моделі `Kennel`: `models.ForeignKey(Kennel, related_name='kennellosts', on_delete=models.CASCADE, blank = True)`

– `gender` – стать тварини, є булевим полем (два можливі значення) `models.BooleanField();`

– `participant` – учасник програми, який або загубив цю тварину, або створив увідомлення про знайдену тварину (він може або тривати її у себе, або передати в притулок, представник якого і вносить ці дані), є зовнішнім ключем моделі `Kennel`: `models.ForeignKey(Kennel, related_name='participantlosts', on_delete=models.CASCADE);`

– `lostfound` – визначає, про що було опубліковано дану інформацію, про знайденні тварини чи про загублення, є булевим полем: `models.BooleanField();`

– `result` – результат розшуку за цим увідомленням, є символьним полем з довжиною в 1 символ і обмеженим набором значень `LOSTRESULTS`: `models.CharField(max_length=1, choices= LOSTRESULTS).`

Даний клас має метод `save`, у якому при збереженні змін інформації про загублену або знайдену тварину, виконується наступний порядок дій. Перевіряється результат розшуку і якщо він успішний, то виконується переобчислення показника кількості успішного знаходження тварини, яка

перебувала в тому ж притулку, що і та, відносно якої зберігається інформація.

Модель `Lostcharvalues` представляє значення характеристик загубленої або знайденої тварини, використовуючи наступний набір полів:

- `note` – опис особливостей даного значення, якщо ці особливості є у конкретної тварини, є текстовим полем: `models.TextField()`;

- `lost` – визначає, інформації про яку загублену або знайдену тварину стосується це значення, є зовнішнім ключем моделі `Lost`: `models.ForeignKey(Lost, related_name='charvalues', on_delete=models.CASCADE)`;

- `charvalue` – конкретне значення характеристики, що асоціюється з цією твариною, є зовнішнім ключем моделі `Charvalue`: `models.ForeignKey(Charvalue, related_name='charvalueinlosts', on_delete=models.CASCADE)`.

Модель `Participant` представляє учасника програми пошуку загублених тварин, використовуючи наступний набір полів:

- `phone` – номер телефону учасника, є символьним полем з максимальною довжиною у 13 символів: `models.CharField(max_length=13)`;

- `knocks` – кількість випадків, коли цей учасник знайшов подібність між оголошеннями про загублену і про знайдену тварин, є цілочисельним полем: `models.IntegerField()`;

- `successknocks` – кількість підтверджених випадків, коли цей учасник знайшов подібність між оголошеннями про загублену і про знайдену тварин, що виявилось правдою, є цілочисельним полем: `models.IntegerField()`.

Модель `Characteristic` представляє характеристики, якими можуть описуватися знайдені або загублені тварини, використовуючи наступний набір полів:

- `chcalled` – назва характеристики, є символьним полем з максимальною довжиною 25 символів: `models.CharField(max_length=25)`;

- `note` – опис характеристики, є текстовим полем `models.TextField()`.

Модель Charvalue представляє можливі значення характеристики, якою описується тварина, використовуючи наступний набір полів:

- note – опис значення характеристики, є текстовим полем: `models.TextField();`
- vcalled – назва значення характеристики (фактично саме значення), є символьним полем з максимальною довжиною 25 символів: `models.CharField(max_length=25);`
- characteristic – характеристика, якій відповідає це значення, є зовнішнім ключем моделі Characteristic: `models.ForeignKey(Characteristic, related_name='charactvalues', on_delete=models.CASCADE).`

Модель Photo представляє фотографію загубленої або знайденої тварини, використовуючи наступний набір полів:

- note – опис фотографії, є символьним полем з максимальною довжиною 200 символів: `models.CharField(max_length=200);`
- fileroute – шлях до файлу з фотографією, є файловим полем з максимальною довжиною шляху в 255 символів: `models.FileField(upload_to='img/breeds', max_length=255);`
- lost – загублена або знайдена тварина, до якої стосується ця фотографія, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, on_delete=models.CASCADE).`

Модель Application представляє заявку на те, що загублена або знайдена тварина була розпізнана, використовуючи наступний набір полів:

- note – опис деталей, якщо це потрібно, є текстовим полем: `models.TextField();`
- lostname – ім'я тварини (використовується, коли господар тварини знаходить її за оголошенням про знайдену тварину), є символьним полем з максимальною довжиною в 20 символів: `models.CharField(max_length=20);`
- participant – учасник, який вважає дану тварину своєю, є зовнішнім ключем моделі Participant: `models.ForeignKey(Participant, related_name='participantapplications', on_delete=models.CASCADE);`

– lost – загублена або знайдена тварина, якої стосується ця знахідка, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, related_name='lostapplications', on_delete=models.CASCADE);`

– searched – зворотнє оголошення про відповідно знайдення або загублення тварини, за яким учасник надавав інформацію в програмі, перед тим як створив цю заявку, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, related_name='seachapplications', on_delete=models.CASCADE);`

– done – час, коли була створена заявка, є полем дати і часу: `models.DateTimeField();`

– result – результат роботи над заявкою, який є символьним полем з довжиною в 1 символ, значення якого належить обмеженому набору значень RESULTS: `models.CharField(max_length=1, choices= RESULTS).`

Даний клас має метод save, у якому при збереженні змін заявки виконується наступний порядок дій. Перевіряється результат заявки і якщо він успішний, то виконується переобчислення показника кількості успішного знаходження тварини, яка перебувала в тому ж притулку, що і та, відносно якої зберігається інформація, на основі поданих заявок та кількості загалом заявок за тваринами з притулку. Якщо результат неуспішний, то оновлюється загальна кількість заявок щодо тварин з цього притулку.

Модель Meeting представляє інформацію про зустріч з загубленою твариною, використовуючи наступний набір полів:

– note – опис деталей зустрічі, є текстовим полем: `models.TextField();`

– participant – учасник, який побачив тварину, є зовнішнім ключем моделі Participant: `models.ForeignKey(Participant, related_name='participantmeetings', on_delete=models.CASCADE);`

– lost – тварина, яку зустрів учасник, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, related_name='lostmeetings', on_delete=models.CASCADE);`

- `foundcity` – місто, в якому учасник побачив тварину, є зовнішнім ключем моделі `City`: `models.ForeignKey(City, related_name='citymeetings', on_delete=models.CASCADE);`

- `found` – адреса, за якою учасник бачив тварину: є символьним полем з максимальною кількістю символів 255: `models.CharField(max_length=255);`

- `founded` – час, коли це відбулось? є полем дати і часу: `models.DateTimeField();`

Модель `Message` представляє повідомлення від одного учасника до іншого, використовуючи наступний набір полів:

- `participant` – учасник, який є автором повідомлення, є зовнішнім ключем моделі `Participant`: `models.ForeignKey(Participant, related_name='participantmessages', on_delete=models.CASCADE);`

- `participant_to` – учасник, якому адресовано повідомлення, є зовнішнім ключем моделі `Participant`: `models.ForeignKey(Participant, related_name='participantmessagesto', on_delete=models.CASCADE);`

- `messtext` – текст повідомлення, є текстовим полем: `models.TextField();`

Модель `Kennel` представляє притулок для тварин, в якому може знаходитися тварина після її знаходження, використовуючи наступний набір полів:

- `note` – опис притулку, є текстовим полем: `models.TextField();`

- `kenname` – назва притулку, є символьним полем з максимальною довжиною 255 символів: `models.CharField(max_length=255);`

- `address` – адреса, за якою знаходиться притулок, є символьним полем з максимальною довжиною 255 символів: `models.CharField(max_length=255);`

- `email` – електронна поштова адреса притулку, є символьним полем з максимальною довжиною 100 символів: `models.CharField(max_length=100);`

- `phone` – номер телефону притулку, є символьним полем з максимальною довжиною 13 символів: `models.CharField(max_length=13);`

– city – місто, в якому знаходиться притулок, є зовнішнім ключем моделі City: `models.ForeignKey(City, related_name='citykennels', on_delete=models.CASCADE);`

– successlost – кількість знайдених тварин у притулку, для яких було знайдено їх господарів, є цілочисельним полем: `models.IntegerField();`

– applies – кількість оброблених заявок від учасників, які вважали себе господарями тварин, які знаходились у притулку, є цілочисельним полем: `models.IntegerField();`

– successapplies – кількість оброблених заявок від учасників, які вважали себе господарями тварин, які знаходились у притулку, і які виявилися фактично їх господарями, є цілочисельним полем: `= models.IntegerField();`

– knocks – кількість оброблених випадків, коли сторонні учасники знаходили відповідність між різними оголошеннями стосовно тварин, які знаходились у притулку, є цілочисельним полем: `models.IntegerField();`

– successknocks – кількість оброблених випадків, коли сторонні учасники знаходили відповідність між різними оголошеннями стосовно тварин, які знаходились у притулку, і які виявляли фактичних господарів цих тварин, є цілочисельним полем: `models.IntegerField();`

Модель City представляє місто, використовуючи наступний набір полів:

– ciname – назва міста, є символьним полем з максимальною довжиною 255 символів: `models.CharField(max_length=255);`

– country – назва країни, в якій розташовано місто, є символьним полем з максимальною довжиною 100 символів: `models.CharField(max_length=100);`

Модель Knock представляє інформацію від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину, використовуючи наступний набір полів:

– participant – сторонній учасник, який створив увідомлення, є зовнішнім ключем моделі Participant: `models.ForeignKey(Participant, related_name='participantknocks', on_delete=models.CASCADE);`

– lost – загублена тварина, відносно якої створено увідомлення, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, related_name='lostknocks', on_delete=models.CASCADE);`

– lost_to – знайдена тварина, відносно якої створено увідомлення, є зовнішнім ключем моделі Lost: `models.ForeignKey(Lost, related_name='lostknocksto', on_delete=models.CASCADE);`

– done – час, коли було створено увідомлення, є полем дати і часу: `models.DateTimeField();`

– result – результат оброблення увідомлення, який представляє собою 1 символ, що відповідає значенню з обмеженого набору RESULTS: `models.CharField(max_length=1, choices= RESULTS).`

Даний клас має метод `save`, у якому при збереженні змін увідомлення виконується наступний порядок дій. Перевіряється результат увідомлення і якщо він успішний, то виконується переобчислення показника кількості успішного знаходження тварини, яка перебувала в тому ж притулку, що і та, відносно якої зберігається інформація, на основі поданих увідомлень та кількості загалом увідомлень за тваринами з притулку, а також такої кількості увідомлень і успішних увідомлень у даного учасника, який створив це увідомлення. Якщо результат неуспішний, то оновлюється загальна кількість увідомлень щодо тварин з цього притулку та загальна кількість увідомлень у учасника, який створив це увідомлення.

Множина можливих результатів увідомлення RESULTS включає наступні значення:

- створене: перевірка не виконана, але увідомлення створене;
- побачене: перевірка не виконана, але увідомлення побачене однією зі сторін учасників;
- підтверджене: це дійсно одна й та сама тварина;

– скасоване: виявилось, що увідомлення некоректне.

Множина можливих результатів оголошення про загублену або знайдену тварину LOSTRESULTS включає наступні значення:

- створене;
- успіх: господаря або тварину успішно знайдено;
- завершене: пошук завершено.

Підготовка даних для сторінок вебзастосунку реалізовувалась через представлення. Шаблони створені на основі вільного шаблону [13].

Клас `LostView` є класом представлення детального перегляду інформації про загублену або знайдену тварину.

Клас `SearchLostsView` є класом представлення формування і перегляду результатів пошуку інформації серед загублених тварин. Пошук реалізується на основі параметрів, що включають:

- назву тварини (параметр запиту `animal`);
- назву породи тварини (параметр запиту `breed`);
- місто, де загублена тварина (параметр запиту `city`);
- початкову дату, з якої могла бути загублена тварина (параметр запиту `date1`);
- кінцеву дату, до якої могла бути загублена тварина (параметр запиту `date2`);
- номер мікрочипу тварини (параметр запиту `microchip`);
- стать тварини (параметр `gender`);
- значення характеристик тварин (параметр `charvalues`).

Будь-яке значення з цих параметрів може бути пропущено, а серед перших двох параметрів передбачається, що буде вказано або тварину, або її породу.

Окремо аналізуються значення параметра `charvalues`. Вони відділяються комами одне від одного, а між комами розташовані пари, що включають характеристику і значення, яке відділене від характеристики пробілом. Після виділення цих пар, далі виконується аналіз наявності

кожного заданого значення характеристики в кожному оголошенні. Якщо в якомусь оголошенні така характеристика не вказана, то воно вилучається зі списку.

Клас `SearchAdLossesView` є класом представлення формування і перегляду результатів пошуку інформації серед загублених тварин на основі власного оголошення про знайдену тварину, тобто фактично надаються користувачу оголошення, які відповідають потенційно його оголошенню і можуть бути результатом його пошуку. Параметри пошуку, аналогічні попередньому представленню, беруться з оголошення, яке використовується як приклад, і включають породу тварини, місто, де загублена тварина, дату, коли була загублена тварина, номер мікрочипу тварини, стать тварини, значення характеристик тварини.

Клас представлення `LostCreate` є класом внесення інформації про загублену або знайдену тварину з програмним внесенням учасника програми, який авторизувався, в якості автора оголошення.

Клас представлення `LostUpdate` є класом оновлення інформації про загублену або знайдену тварину.

Клас представлення `KennelLossesView` є класом представлення перегляду списку знайдених тварин за притулком з заданим номером. Параметри формування списку передаються через параметри запиту до вебсервера і включають:

- номер притулку (параметр запиту `kennel`);
- початкову дату, з якої могла бути знайдена тварина (параметр запиту `date1`);
- кінцеву дату, до якої могла бути знайдена тварина (параметр запиту `date2`).

Клас представлення `LastLossesView` є класом представлення перегляду списку 10 останніх оголошень про загублених тварин. Параметром формування списку, який передається через параметри запиту до вебсервера, є параметр `city`, що визначає місто, де була загублена тварина.

Клас представлення `YourLostsView` є класом представлення перегляду списку власних оголошень авторизованого учасника про загублених тварин, сортуючи їх за датою.

Клас представлення `YourFindsView` є класом представлення перегляду списку власних оголошень авторизованого учасника про знайдених тварин, сортуючи їх за датою.

Клас представлення `ParticipantLostsView` є класом представлення перегляду списку оголошень учасника, номер якого передано через параметр запиту `participant`, про загублених тварин, сортуючи їх за датою.

Клас представлення `ParticipantLostsView` є класом представлення перегляду списку власних оголошень учасника, номер якого передано через параметр запиту `participant`, про знайдених тварин, сортуючи їх за датою.

Клас `SearchFindsView` є класом представлення формування і перегляду результатів пошуку інформації серед знайдених тварин. Пошук реалізується на основі параметрів, що включають:

- назву тварини (параметр запиту `animal`);
- назву породи тварини (параметр запиту `breed`);
- місто, де загублена тварина (параметр запиту `city`);
- початкову дату, з якої могла бути загублена тварина (параметр запиту `date1`);
- кінцеву дату, до якої могла бути загублена тварина (параметр запиту `date2`);
- номер мікрочипу тварини (параметр запиту `microchip`);
- стать тварини (параметр `gender`);
- значення характеристик тварин (параметр `charvalues`).

Будь-яке значення з цих параметрів може бути пропущено, а серед перших двох параметрів передбачається, що буде вказано або тварину, або її породу.

Окремо аналізуються значення параметра `charvalues`. Вони відділяються комами одне від одного, а між комами розташовані пари, що

включають характеристику і значення, яке відділене від характеристики пробілом. Після виділення цих пар, далі виконується аналіз наявності кожного заданого значення характеристики в кожному оголошенні. Якщо в якомусь оголошенні така характеристика не вказана, то воно вилючається зі списку.

Клас `SearchAdFindsView` є класом представлення формування і перегляду результатів пошуку інформації серед знайдених тварин на основі власного оголошення про загублену тварину, тобто фактично надаються користувачу оголошення, які відповідають потенційно його оголошенню і можуть бути результатом його пошуку. Параметри пошуку, аналогічні попередньому представленню, беруться з оголошення, яке використовується як приклад, і включають породу тварини, місто, де знайдена тварина, дату, коли була знайдена тварина, номер мікрочипу тварини, стать тварини, значення характеристик тварини.

Клас `ApplicationView` є класом представлення детального перегляду заявки на те, що загублена або знайдена тварина була розпізнана. Цей клас є особливим, оскільки окрім детального перегляду він ще суміщає форму оброблення перегляду, за якою при відкритті сторінки учасником, якому заявка адресована, стан заявки змінюється на переглянутий.

Клас представлення `ApplicationCreate` є класом внесення заявки на те, що загублена або знайдена тварина була розпізнана власником, з програмним внесенням учасника програми, який авторизувався, в якості автора заявки.

Клас представлення `ApplicationUpdate` є класом оновлення заявки на те, що загублена або знайдена тварина була розпізнана власником.

Клас представлення `KennelApplicationsView` є класом представлення перегляду списку заявок на те, що тварина з притулку з заданим номером була розпізнана власником. Параметри формування списку передаються через параметри запиту до вебсервера і включають:

- номер притулку (параметр запиту `kennel`);

- початкову дату, з якої могла бути знайдена тварина (параметр запиту date1);
- кінцеву дату, до якої могла бути знайдена тварина (параметр запиту date2);
- стан заявки (параметр запиту result).

Клас представлення `YourApplicationsView` є класом представлення перегляду списку власних заявок авторизованого учасника на те, що загублена або знайдена тварина була розпізнана власником, сортуючи їх за номером за зменшенням.

Клас представлення `ToYouApplicationsView` є класом представлення перегляду списку заявок, адресованих авторизованому учаснику про те, що загублена або знайдена тварина була розпізнана власником, сортуючи їх за номером за зменшенням.

Клас `KnockView` є класом представлення детального перегляду інформації від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину. Цей клас є особливим, оскільки окрім детального перегляду він ще суміщає форму оброблення перегляду, за якою при відкритті сторінки учасником, якому інформація адресована, стан заявки змінюється на переглянутий.

Клас представлення `KnockCreate` є класом внесення інформації від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину, з програмним внесенням учасника програми, який авторизувався, в якості автора інформації.

Клас представлення `KnockUpdate` є класом оновлення інформації від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину.

Клас представлення `KennelKnocksView` є класом представлення перегляду списку повідомлень від стороннього учасника програми, який

вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину, яка знаходиться в притулку з заданим номером. Параметри формування списку передаються через параметри запиту до вебсервера і включають:

- номер притулку (параметр запиту `kennel`);
- початкову дату, з якої могла бути знайдена тварина (параметр запиту `date1`);
- кінцеву дату, до якої могла бути знайдена тварина (параметр запиту `date2`);
- стан заявки (параметр запиту `result`).

Клас представлення `YourKnocksView` є класом представлення перегляду списку власних повідомлень про те, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину.

Клас представлення `YourKnockedView` є класом представлення перегляду списку повідомлень від стороннього учасника програми, який вважає, що одне оголошення про загублену тварину відповідає іншому оголошенню про знайдену тварину, авторизованому в даний момент учаснику.

Клас `MeetingView` є класом представлення детального перегляду інформації про зустріч з загубленою твариною.

Клас представлення `MeetingCreate` є класом внесення інформації про зустріч з загубленою твариною з програмним внесенням учасника програми, який авторизувався, в якості автора інформації.

Клас представлення `MeetingUpdate` є класом оновлення інформації про зустріч з загубленою твариною.

Клас представлення `ToYouMeetingsView` є класом представлення перегляду списку повідомлень про зустріч тварини, яку загубив авторизований учасник.

Клас представлення YourMeetingsView є класом представлення перегляду списку повідомлень авторизованого учасника про зустріч тварини, які були загублені.

3.3 Висновки за розділом 3

У третьому розділі представлена логіка функціонування програми, що базується на зміні станів основних сутностей, які включають оголошення, заявку та повідомлення. Для цих сутностей побудовано діаграми станів. Описано реалізовані одиниці вебзастосунків на основі логіки першочергового створення коду моделей для роботи з даними та в подальшому – представлень.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис програми

Домашні улюбленці часом губляться в місті і тоді для господаря починається ціле випробування для того, щоб знайти загублену тварину. На даний момент в Україні найчастіше це вирішується через соціальні мережі. До того ж зараз, у зв'язку з війною, проблема загублених тварин значно зросла в розмірі.

Розроблене ПЗ призначене для розшуку загублених тварин, коли власник загубив домашнього улюбленця і намагається його знайти або коли людина знайшла чужого улюбленця і намагається з'ясувати його власника. З метою забезпечення гнучкості процесу розшуку надаються інструменти пошуку за параметрами, створення заявок про власність, інформування третіх осіб про подібність між розшукуваними і знайденими тваринами тощо.

Для підтримки роботи програми необхідно, щоб апаратна частина включала процесор (4 ядра, не менше 2,1 ГГц), оперативну пам'ять (від 4 Гб), жорсткий диск з наявним вільним простором (від 500 Мб).

4.2 Експлуатація програми

Експлуатація програми всіма користувачами, як то представниками притулків, так і звичайними учасниками програми, так і неавторизованими користувачами відбувається за допомогою одного єдиного інтерфейсу, який з'являється перед користувачем одразу після першого звернення до вебсайту, за яким розташовано вебзастосунок.

Роботу усіх описаних користувачів реалізовано на основі наступного загального меню з наступних пунктів.

Пункт меню «Шукаю» дозволяє працювати з оголошеннями, в яких власники тварин повідомляють про загублення власних улюбленців (рис. 4.1).

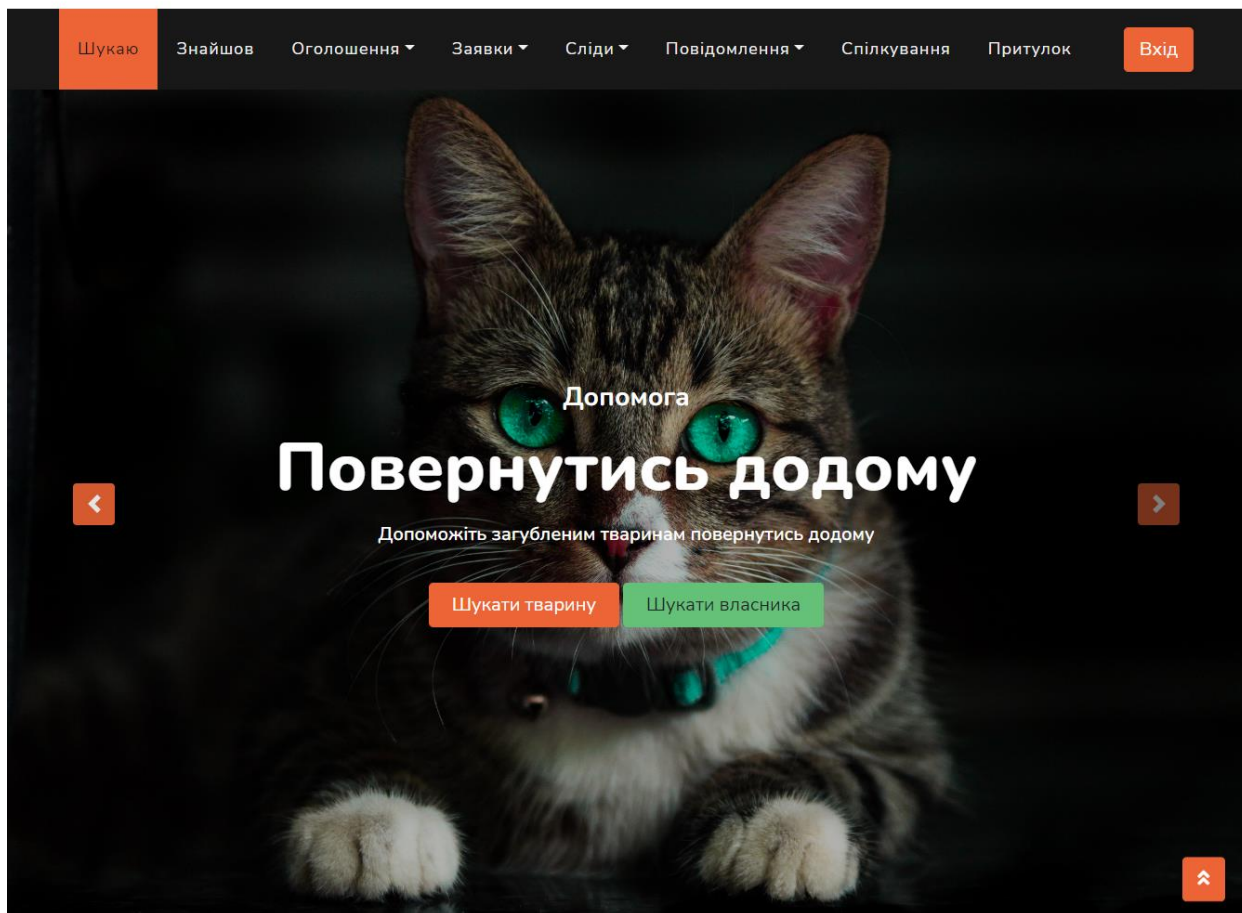


Рисунок 4.1 – Інтерфейс програми

Сторінка оголошень про втрату тварин окрім загальної інформації про програму містить також кнопки «Шукати тварину» та «Шукати власника» для виконання пошуку відповідних оголошень.

Окрім того нижче на сторінці знаходиться список останніх 10 оголошень про загублених тварин та форма пошуку серед цих оголошень.

Список останніх загублених тварин (рис. 4.2) має випадаючий список для вибору міста, за яким ці 10 останніх оголошень потрібно продемонструвати. Значення «всі міста» дозволяє відображати оголошення без визначення міста. За кожним оголошенням відображається фотографія тварини, натискаючи на яку можна перейти до повного оголошення. Нижче фотографії розташовано інформацію про місто, де було загублено тварину, та час, коли це відбулось.

Рисунок 4.2 – Перегляд останніх загублених тварин

Форма пошуку включає наступні поля:

- назву тварини, що вибирається з випадаючого списку з можливістю введення;
- порodu тварини, що вибирається з випадаючого списку з можливістю введення;
- місто, де було загублено тварину, що вибирається з випадаючого списку з можливістю введення;
- стать тварини;
- початкова дата, з якої могла бути загублена тварина;
- кінцева дата, коли могла бути загублена тварина;
- номер мікрочіпу.

Будь-яке поле пошуку може бути пропущене.

За допомогою кнопки «Шукати» відбувається перехід до результатів пошуку (рис. 4.3).

Кнопка «Розширений пошук» дозволяє відобразити додаткові поля, які дозволяють задати додаткові характеристики тварини і вказати їх значення.

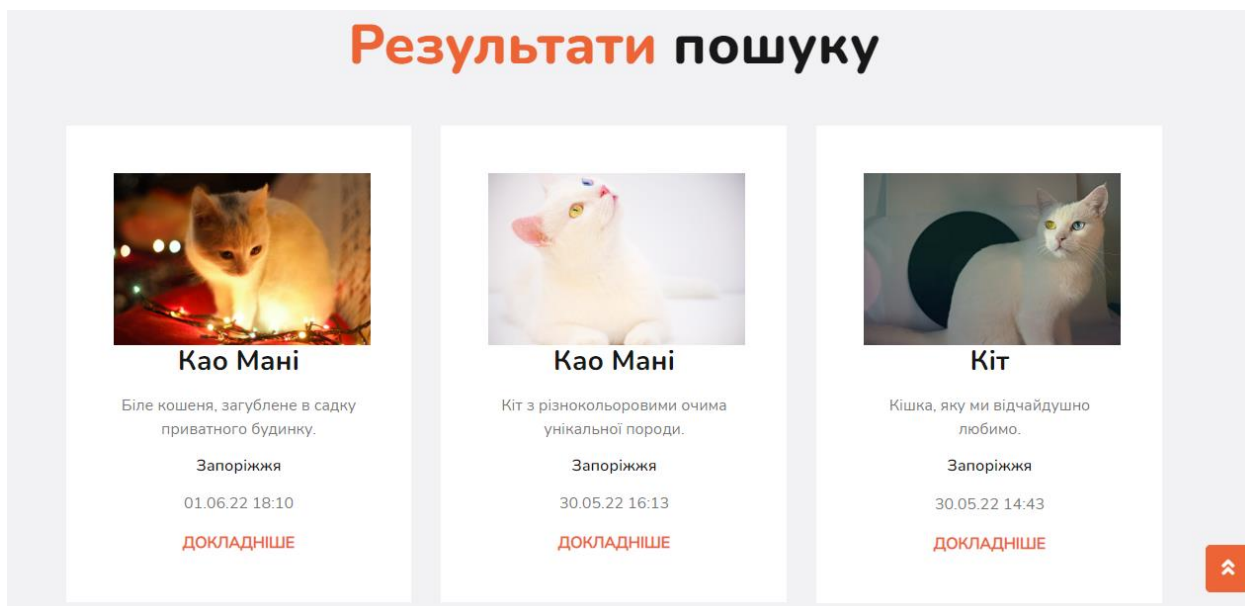


Рисунок 4.3 – Результати пошуку серед загублених тварин

Результати пошуку включають за кожним оголошенням фотографію тварини, коротку частину опису тварини, місто, де було загублено тварину, час цієї події. Посилання «Докладніше» дозволяє перейти до сторінки самого оголошення з повним відображенням цієї інформації.

Пункт меню «Знайшов» дозволяє виконувати всі ті самі дії, але над оголошеннями щодо знаходження тварин, а не загублення.

Сторінка оголошення (рис. 4.4) дозволяє, по-перше, переглянути основну інформацію про тварину, що включає фотографію тварини, породу тварини, опис тварини, місто і адресі, де було загублено (або відповідно знайдено) тварину, час загублення або знаходження, стать тварини, номер мікрочіпу (якщо він є), притулок, де містить тварина (якщо вона була знайдена), користувача, який створив оголошення, а також список окремих характеристик та значення цих характеристик. Нижче на сторінці розташовано наступні можливості:

- кнопка «Моя тварина» дозволяє створити заявку власника тварини;
- кнопка «Повідомити про відповідність» дозволяє вказати на відповідність даного оголошення оголошенню, де навпаки було загублено (або знайдено) тварину;

– список відповідних оголошень, якщо оголошення переглядає власник (для всіх інших недоступно).

Знайдена тварина

Європейський короткошерстний кіт

Рудий кіт з білими смужками. Тепер, після лікування, дуже грайливий, активний. Знайдений на поверсі розбитої багатоповерхівки.

- ✓ Харків
- ✓ Салтівка
- ✓ 22.05.22 11:07
- ✓ чоловіча стать
- ✓ мікрочіпу немає
- ✓ Притулок: Черкаський будинок допомоги
- ✓ Користувач: ivan
- ✓ Колір: рудий
- ✓ Очі: карі
- ✓ Шерсть: коротка
- ✓ Хвіст: середній

Моя тварина

Повідомити про подібність



Рисунок 4.4 – Перегляд оголошення

Пункт меню «Оголошення» включає підпункти «Знайдені» та «Загублені», що дозволяє працювати з власними оголошеннями, в тому числі редагувати їх, переглядати, створювати нові, стосовно знайдених тварин або загублених.

Пункт меню «Заявки» включає підпункти «Власні», що дозволяє переглядати власні заявки про власність тварин, та «Адресовані», що дозволяє переглянути заявки, які були адресовані авторизованому учаснику.

Пункт меню «Сліди» включає підпункти «Власні», що дозволяє переглядати власні відслідковування інших тварин, та «Адресовані», що дозволяє переглянути випадки, коли інші учасники бачили тварин, яких розшукує учасник.

Пункт меню «Повідомлення» включає підпункти «Власні», що дозволяє переглядати власні повідомлення про відповідність між оголошеннями, та «Адресовані», що дозволяє переглянути випадки, коли інші учасники повідомили про відповідність власних оголошень авторизованого учасника іншим оголошенням (рис. 4.5).

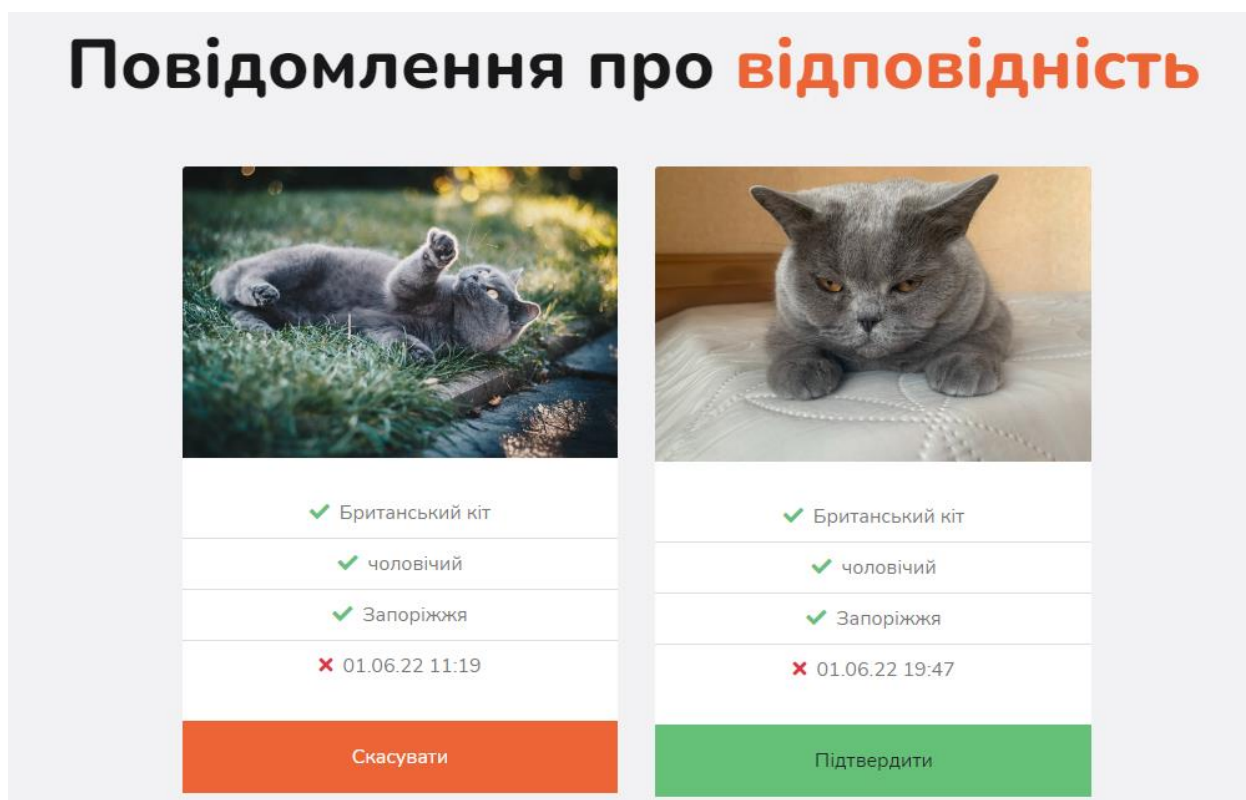


Рисунок 4.5 – Повідомлення про відповідність

Усі такі повідомлення розташовані як пара оголошень. Одне – оголошення про загублення тварини. Інше – оголошення про знаходження тварини. З цим повідомленням можна погодитись, натиснувши на кнопку «Підтвердити», або не погодитись, натиснувши на кнопку «Скасувати». Для детального перегляду оголошення можна натиснути на фотографію тварини. Якщо зазначено якийсь опис, то він виводиться нижче окремим полем під таблицею.

Пункт меню «Спілкування» дозволяє переглянути спілкування учасника з іншими учасниками та створити нові повідомлення один одному.

Пункт меню «Притулок» дозволяє керувати оголошеннями, що стосуються тварин у притулку. Скористатися цим пунктом фактично може учасник, асоційований з притулком. Окрім того звідти учасник може асоціювати інших учасників з притулком. Усі оголошення, заявки доступні за допомогою відповідних кнопок.

4.3 Тестування програми

Реалізацію сценаріїв роботи учасників програми розшуку тварин через ПЗ було перевірено послідовно. Під час тестування використовувався браузер Chrome останньої актуальної версії на момент тестування з персонального ноутбуку під керуванням Windows 10 та мобільного телефону під керуванням Android 11. Тестування програми підтвердило успішність реалізації.

4.4 Висновки за розділом 4

Розглянуто конструкцію інтерфейсу користувача у вебзастосунку. Описано можливості виконання основних функцій програми учасником програми розшуку тварин. Виконано тестування програми для перевірки виконання запланованих вимог, що підтвердило успішну реалізацію.

ВИСНОВКИ

У дипломній роботі розроблено ПЗ розшуку загублених тварин, що було направлено на сприяння возз'єднанню загублених тварин зі своїми власниками. З цією метою з одного боку було вирішено створити програму, яка б була єдиним майданчиком для взаємодії тих, хто загубив тварину, з тими, хто її знайшов, або з притулком, який її утримує після знаходження в себе, а з іншого – надати додаткові інструменти, які полегшували б процес встановлення відповідності між оголошеннями цих двох сторін.

Основним результатом роботи є розроблена програма, а для цього у процесі розробки було виконано аналіз проблеми з оглядом існуючих програмних аналогів, визначенням вимог до програми, вибором засобів розробки, включаючи мови Python, JavaScript, HTML, CSS, вебфреймворк Django, систему керування БД MySQL, виконано проєктування БД, логіки роботи програми. Логіка функціонування програми, що базується на зміні станів основних сутностей, які включають оголошення, заявку та повідомлення, була представлена відповідними діаграмами станів. Окрім того описано реалізовані одиниці вебзастосунків на основі логіки першочергового створення коду моделей для роботи з даними та в подальшому – представлень. Описано основні принципи роботи з інтерфейсами програми користувачів.

Розроблене ПЗ дозволяє виконувати пошук загублених тварин власниками і пошук власників знайдених тварин, контактувати цим сторонам між собою, третім сторонам повідомляти про відповідність таких оголошень між собою, а також повідомляти власникам тварин про випадки, коли учасник програми бачив тварину після її загублення.

ПЗ фактично в результаті на практиці може використовуватися для взаємодії людей, які є власниками тварин і загубили їх, і з людьми, які їх знайшли, а також з притулками тварин.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hamilton, M. US Missing Pet Epidemic and Euthanasia Statistics [Electronic resource] : Facts/Figures / M. Hamilton. – Access mode : <https://peeva.co/blog/missing-pet-epidemic-facts-and-figures>.
2. Загублені та знайдені тварини. Луцьк та Волинська обл. Facebook [Електрон. ресурс]. – Режим доступу : <https://uk-ua.facebook.com/groups/1126808634017634/>.
3. Petco Love Lost [Electronic resource]. – Access mode : <https://lost.petcolove.org/>.
4. Lost animals [Electronic resource]. – Access mode : <https://www.mildura.vic.gov.au/Services/Animals-Pets/Lost-animals>.
5. Lost Pets. RSPCA [Electronic resource]. – Access mode : <https://rspcavic.org/lostpets/>.
6. Ramalho, L. Fluent Python [Text] : Clear, Concise, and Effective Programming / L. Ramalho. – Massachusetts : O'Reilly Media, 2022. – 1012 p.
7. Campbell, S. Python Vs PHP [Electronic resource] : What's the Difference Between Python and PHP? / S. Campbell. – Access mode : <https://www.guru99.com/python-vs-php.html>.
8. What is Python used for? 10 practical Python uses [Electronic resource]. – Access mode : <https://www.futurelearn.com/info/blog/what-is-python-used-for>.
9. Django MVT [Electronic resource]. – Access mode : <https://www.javatpoint.com/django-mvt>.
10. Mele, A. Django 3 By Example [Text] : Build powerful and reliable Python web applications from scratch, 3rd Edition / A. Mele. – Birmingham : Packt Publishing, 2020. – 568 p.
11. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.
12. Django Tutorials – Real Python [Electronic resource]. – Access

mode : <https://realpython.com/tutorials/django/>.

13. PetLover Free Website Template. Free CSS Tepmlates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page276/petlover>.

ДОДАТОК А
Технічне завдання

Вступ

Домашні улюбленці часом губляться в місті і тоді для господаря починається ціле випробування для того, щоб знайти загублену тварину. На даний момент в Україні найчастіше це вирішується через соціальні мережі. До того ж зараз, у зв'язку з війною, проблема загублених тварин значно зросла в розмірі. Тому розробка програми для підтримки розшуку тварин є важливим завданням.

A.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується на основі завдання, виданого за темою «Програмне забезпечення розшуку загублених тварин» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для розшуку загублених тварин, коли власник загубив домашнього улюбленця і намагається його знайти або коли людина знайшла чужого улюбленця і намагається з'ясувати його власника. З метою забезпечення гнучкості процесу розшуку надаються інструменти пошуку за параметрами, створення заявок про власність, інформування третіх осіб про подібність між розшукуваними і знайденими тваринами тощо.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Набір функціональних вимог, які має забезпечувати програма, включає:

- пошук загубленої тварини;
- пошук за знайденою твариною;
- внесення даних про загублену тварину;
- внесення даних про знайдену тварину;
- визначення знайденої тварини власником;
- повідомлення про тварину;
- відправлення повідомлення учаснику;
- перегляд повідомлень між учасниками;
- повідомлення про подібність між загубленою та знайденою тваринами;
- зв'язування учасників з притулком;
- перегляд розшукуваних тварин у притулку, заявок власників;
- перегляд останніх знайдених тварин;
- перегляд останніх загублених тварин;
- перегляд власних знайдених або загублених тварин або за заданим учасником;
- перегляд власних внесених заявок на власність, повідомлень про подібність або відслідковувань;
- перегляд адресованих користувачу заявок на власність, повідомлень про подібність або відслідковувань;
- авторизація і реєстрація в програмі.

A.3.2 Умови експлуатації

Умовами експлуатації є виділення менеджера або менеджерів для реалізації внесення даних в систему. Менеджер може також виконувати роль адміністратора, але краще виділити для цього окремо системного адміністратора, який також буде керувати базою даних.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для підтримки роботи програми необхідно, щоб апаратна частина включала процесор (4 ядра, не менше 2,1 ГГц), оперативну пам'ять (від 4 Гб), жорсткий диск з наявним вільним простором (від 500 Мб).

А.4 Порядок контролю та приймання

Дипломна кваліфікаційна робота має виконуватися у відповідності з визначеним календарним планом згідно встановлених завдань, їх строків і результатів виконання.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```

from django.db import models
from django.contrib.auth.models import AbstractUser
from django.urls import reverse
from django.utils import timezone
from django.conf import settings
import datetime

RESULTS = (
    ('A', 'створене'),
    ('B', 'побачене'),
    ('C', 'підтверджене'),
    ('D', 'скасоване')
)

LOSTRESULTS = (
    ('A', 'створене'),
    ('B', 'успіх'),
    ('C', 'завершене')
)

class Animal (models.Model):
    ancalled = models.CharField(max_length=25)

    def __str__(self):
        return self.ancalled

class Breed (models.Model):
    brcalled = models.CharField(max_length=50)
    note = models.TextField()
    animal = models.ForeignKey(Animal,
related_name='animalbreeds', on_delete=models.CASCADE)
    photo = models.FileField(upload_to='img/breeds',
max_length=255)

    def __str__(self):
        return self.brcalled

    def get_absolute_url(self):
        return reverse('breeddetail', kwargs={'pk':
self.pk})

```

```

class Lost (models.Model):
    note = models.TextField()
    found = models.DateTimeField()
    breed = models.ForeignKey(Breed,
related_name='breedlosts', on_delete=models.CASCADE)
    foundcity = models.ForeignKey(City,
related_name='citylosts', on_delete=models.CASCADE)
    foundat = models.CharField(max_length=255)
    microchip = models.CharField(max_length=10)
    kennel = models.ForeignKey(Kennel,
related_name='kennellosts', on_delete=models.CASCADE, blank =
True)

    gender = models.BooleanField()
    participant = models.ForeignKey(Kennel,
related_name='participantlosts', on_delete=models.CASCADE)
    lostfound = models.BooleanField()
    result = models.CharField(max_length=1, choices=
LOSTRESULTS, default='A')

    def __str__(self):
        return self.breed.brcalled

    def get_absolute_url(self):
        return reverse('lostdetail', kwargs={'pk':
self.pk})

    def save(self, *args, **kwargs):
        if self.result == 'C':
            ken = self.kennel
            losts = Lost.objects.filter(kennel = ken)
            succlostscounts = losts.filter(result =
'B').count()

            ken.successlosts = succlostscounts
            ken.save()
            super(Lost, self).save(*args, **kwargs)

class LostCharvalue (models.Model):
    note = models.TextField()
    lost = models.ForeignKey(Lost,
related_name='charvalues', on_delete=models.CASCADE)
    charvalue = models.ForeignKey(Charvalue,
related_name='charvalueinlosts', on_delete=models.CASCADE)

class Participant (AbstractUser):
    phone = models.CharField(max_length=13)
    knocks = models.IntegerField()

```

```

        successknocks = models.IntegerField()

    def __str__(self):
        return self.first_name

    def get_absolute_url(self):
        return reverse('participantdetail', kwargs={'pk':
self.pk}))

class Characteristic (models.Model):
    chcalled = models.CharField(max_length=25)
    note = models.TextField()

    def __str__(self):
        return self.chcalled

class Charvalue (models.Model):
    note = models.TextField()
    vcalled = models.CharField(max_length=25)
    characteristic = models.ForeignKey(Characteristic,
related_name='charactvalues', on_delete=models.CASCADE)

    def __str__(self):
        return self.vcalled

class Photo (models.Model):
    note = models.CharField(max_length=200)
    fileroute = models.FileField(upload_to='img/breeds',
max_length=255)
    lost = models.ForeignKey(Lost,
on_delete=models.CASCADE)

    def __str__(self):
        return self.note

class Application (models.Model):
    note = models.TextField()
    lostname = models.CharField(max_length=20)
    participant = models.ForeignKey(Participant,
related_name='participantapplications',
on_delete=models.CASCADE)
    lost = models.ForeignKey(Lost,
related_name='lostapplications', on_delete=models.CASCADE)
    searched = models.ForeignKey(Lost,
related_name='seachapplications', on_delete=models.CASCADE)

```

```

        done =
models.DateTimeField(default=datetime.datetime.now)
        result = models.CharField(max_length=1, choices=
RESULTS, default='A')

    def get_absolute_url(self):
        return reverse('applicationdetail', kwargs={'pk':
self.pk})

    def save(self, *args, **kwargs):
        if self.result == 'C':
            ken = self.lost.kennel
            applies =
Application.objects.filter(lost__kennel = ken)
            appliescounts = applies.filter(result =
'C').count()

            ken.successapplies = appliescounts
            appliescounts = applies.count()
            ken.applies = appliescounts
            ken.save()
        else:
            ken = self.lost.kennel
            applies =
Application.objects.filter(lost__kennel = ken)
            ken.applies = applies.counts()
            ken.save()
        super(Application, self).save(*args, **kwargs)

class Meeting (models.Model):
    note = models.TextField()
    participant = models.ForeignKey(Participant,
related_name='participantmeetings', on_delete=models.CASCADE)
    lost = models.ForeignKey(Lost,
related_name='lostmeetings', on_delete=models.CASCADE)
    foundcity = models.ForeignKey(City,
related_name='citymeetings', on_delete=models.CASCADE)
    found = models.CharField(max_length=255)
    founded =
models.DateTimeField(default=datetime.datetime.now)

    def get_absolute_url(self):
        return reverse('meetingdetail', kwargs={'pk':
self.pk})

```

```

class Message (models.Model):
    participant = models.ForeignKey(Participant,
related_name='participantmessages', on_delete=models.CASCADE)
    participant_to = models.ForeignKey(Participant,
related_name='participantmessagesto', on_delete=models.CASCADE)
    messtext = models.TextField()

    def __str__(self):
        return self.messtext

class Kennel (models.Model):
    note = models.TextField()
    kenname = models.CharField(max_length=255)
    address = models.CharField(max_length=255)
    email = models.CharField(max_length=100)
    phone = models.CharField(max_length=13)
    city = models.ForeignKey(City,
related_name='citykennels', on_delete=models.CASCADE)
    successlosts = models.IntegerField()
    applies = models.IntegerField()
    successapplies = models.IntegerField()
    knocks = models.IntegerField()
    successknocks = models.IntegerField()
    participants = models.ManyToManyField(Participant,
related_name = 'participantkennels', db_table =
'KennelParticipants')

    def get_absolute_url(self):
        return reverse('kenneldetail', kwargs={'pk':
self.pk})

    def __str__(self):
        return self.kenname

class City (models.Model):
    ciname = models.CharField(max_length=255)
    country = models.CharField(max_length=100)

    def get_absolute_url(self):
        return reverse('citydetail', kwargs={'pk':
self.pk})

    def __str__(self):
        return self.ciname

```

```

class Knock (models.Model):
    participant = models.ForeignKey(Participant,
related_name='participantknocks', on_delete=models.CASCADE)
    lost = models.ForeignKey(Lost,
related_name='lostknocks', on_delete=models.CASCADE)
    lost_to = models.ForeignKey(Lost,
related_name='lostknocksto', on_delete=models.CASCADE)
    done =
models.DateTimeField(default=datetime.datetime.now)
    result = models.CharField(max_length=1, choices=
RESULTS)

    def get_absolute_url(self):
        return reverse('knockdetail', kwargs={'pk':
self.pk})

    def save(self, *args, **kwargs):
        if self.result == 'C':
            ken = self.lost.kennel
            knocks = Knock.objects.filter(lost__kennel =
ken)

            knockcounts = knocks.filter(result =
'C').count()

            ken.successknocks = knockcounts
            knockcounts = knocks.count()
            ken.knocks = knockcounts
            ken.save()
            me = self.participant
            knocks = Knock.objects.filter(participant =
me)

            knockcounts = knocks.filter(result =
'C').count()

            me.successknocks = knockcounts
            knockcounts = knocks.count()
            me.knocks = knockcounts
            me.save()
        else:
            ken = self.lost.kennel
            knocks = Knock.objects.filter(lost__kennel =
ken)

            ken.knocks = knocks.counts()
            ken.save()
            me = self.participant
            knocks = Knock.objects.filter(participant =
me)

            me.knocks = knocks.counts()

```

```

        me.save()
    super(Knock, self).save(*args, **kwargs)

```

Б.2 Текст файла lostviews.py

```

from django.views.generic.edit import CreateView
from django.views.generic import ListView, UpdateView
from django.views.generic.detail import DetailView
from .models import Lost, Breed, City, LostCharvalue,
Kennel, Animal, Charvalue, Participant
import datetime
from django.urls import reverse_lazy

class LostView (FormMixin, DetailView):
    template_name = 'lost.html'
    model = Lost

class SearchLostsView(ListView):
    template_name = 'searchlosts.html'
    model = Lost

    def get_queryset(self):
        losts = Lost.objects.filter(found = True)
        losts = losts.order_by('-found')
        animal = self.request.GET.get('animal')
        if animal:
            animal = Animal.objects.get(id = animal)
            losts = losts.filter(breed__animal__id =
animal)

        breed = self.request.GET.get('breed')
        if breed:
            breed = Breed.objects.get(id = breed)
            losts = losts.filter(breed__id = breed)
        city = self.request.GET.get('city')
        if city:
            city = City.objects.get(id = city)
            losts = losts.filter(foundcity__id = city)
        datel = self.request.GET.get('date1')
        if datel:
            datel = datetime.datetime.strptime(datel,
'%d.%m.%y')

            losts = losts.filter(found__gte = datel)
        date2 = self.request.GET.get('date2')
        if date2:

```

```

        date2 = datetime.datetime.strptime(date2,
'%d.%m.%y')

        losts = losts.filter(found__lte = date2)
        microchip = self.request.GET.get('microchip')
        if microchip:
            losts = losts.filter(microchip = microchip)
        gender = self.request.GET.get('gender')
        if gender:
            losts = losts.filter(gender = gender)
        charvalues = self.request.GET.get('charvalues')
        vals = charvalues.split(',')
        vs = []
        for val in vals:
            v = val.split(' ')
            char = v[0]
            chval = v[1]
            vs.append(Charvalue.objects.get(vcalled =
chval, characteristic__chcalled = v))
        rlosts = losts
        for lost in losts:
            for v in vs:
                r = LostCharvalue.objects.filter(lost =
lost, charvalue = v)

                if not r:
                    rlosts.delete(lost)
        rlosts = rlosts.filter(lostfound = True)
        return losts

def get_context_data(self, **kwargs):
    dcontext = super(ParticipantFindsView,
self).get_context_data(**kwargs)
    animal = self.request.GET.get('animal')
    dcontext['animal'] = animal
    breed = self.request.GET.get('breed')
    dcontext['breed'] = breed
    city = self.request.GET.get('city')
    dcontext['city'] = city
    date1 = self.request.GET.get('date1')
    dcontext['date1'] = date1
    date2 = self.request.GET.get('date2')
    dcontext['date2'] = date2
    microchip = self.request.GET.get('microchip')
    dcontext['microchip'] = microchip
    gender = self.request.GET.get('gender')
    dcontext['gender'] = gender
    charvalues = self.request.GET.get('charvalues')

```

```

        dcontext['charvalues'] = charvalues
        return dcontext

class SearchAdLostsView(ListView):
    template_name = 'searchadlosts.html'
    model = Lost

    def get_queryset(self):
        losts = Lost.objects.filter(lostfound = True)
        losts = losts.order_by('-found')
        find = self.request.GET.get('find')
        find = Lost.objects.get(id = find)
        losts = losts.filter(breed__id = find.breed)
        losts = losts.filter(foundcity__id =
find.foundcity)
        losts = losts.filter(found__gte = find.found)
        losts = losts.filter(microchip = find.microchip)
        losts = losts.filter(gender = find.gender)
        charvalues = find.charvalues
        losts = losts.filter(lostfound = True)
        rlosts = losts
        for lost in losts:
            for v in charvalues:
                r = LostCharvalue.objects.filter(lost =
lost, charvalue = v)
                if not r:
                    rlosts.delete(lost)
        return rlosts

    def get_context_data(self, **kwargs):
        dcontext = super(SearchAdLostsView,
self).get_context_data(**kwargs)
        find = self.request.GET.get('find')
        dcontext['find'] = find
        return dcontext

class LostCreate(CreateView):
    model = Lost
    fields = ['found', 'breed', 'foundcity', 'foundat',
'microchip', 'kennel', 'gender', 'participant', 'lostfound',
'note']
    template_name = 'createlost.html'

    def get_success_url(self):
        return reverse_lazy('lost-new', kwargs={'pk':
self.object.pk})

```

```

        def form_valid(self, form):
            part = get_object_or_404(Participant,
id=self.request.user)
            self.participant = part
            return super().form_valid(form)

class LostUpdate(UpdateView):
    model = Lost
    fields = ['found', 'breed', 'foundcity', 'foundat',
'microchip', 'kennel', 'gender', 'participant', 'lostfound',
'note', 'result']
    template_name = 'updatelost.html'

    def get_success_url(self):
        return reverse_lazy('lost-update', kwargs={'pk':
self.object.pk})

class KennelLossesView(ListView):
    template_name = 'kennellosts.html'
    model = Lost
    paginate_by = 20

    def get_queryset(self):
        ken = self.request.GET.get('kennel')
        ken = Kennel.objects.get(id = ken)
        losts = ken.kennellosts
        datel = self.request.GET.get('date1')
        if datel:
            datel = datetime.datetime.strptime(datel,
'%d.%m.%y')

            losts = losts.filter(found__gte = datel)
            date2 = self.request.GET.get('date2')
            if date2:
                date2 = datetime.datetime.strptime(date2,
'%d.%m.%y')

                losts = losts.filter(found__lte = date2)
        return losts

    def get_context_data(self, **kwargs):
        dcontext = super(KennelLossesView,
self).get_context_data(**kwargs)
        datel = self.request.GET.get('date1')
        dcontext['start'] = datel
        date2 = self.request.GET.get('date1')
        dcontext['end'] = date2
        ken = self.request.GET.get('kennel')

```

```

        dcontext['kennel'] = ken
    return dcontext

class LastLossesView(ListView):
    template_name = 'lastlosses.html'
    model = Lost

    def get_queryset(self):
        city = self.request.GET.get('city')
        city = City.objects.get(id = city)
        if city:
            losses = Lost.objects.filter(foundcity = city)
            losses = losses.order_by('-found')[:10]
            return losses

    def get_context_data(self, **kwargs):
        dcontext = super(LastLossesView,
self).get_context_data(**kwargs)
        city = self.request.GET.get('city')
        dcontext['city'] = city
        return dcontext

class YourLossesView(LoginRequiredMixin, ListView):
    template_name = 'yourlosses.html'
    model = Lost

    def get_queryset(self):
        you = self.request.user
        losses = Lost.objects.filter(participant = you)
        losses = losses.filter(found = True)
        losses = losses.order_by('-found')
        return losses

class YourFindsView(LoginRequiredMixin, ListView):
    template_name = 'yourfinds.html'
    model = Lost

    def get_queryset(self):
        you = self.request.user
        losses = Lost.objects.filter(participant = you)
        losses = losses.filter(found = False)
        losses = losses.order_by('-found')
        return losses

class ParticipantLossesView(LoginRequiredMixin, ListView):
    template_name = 'partlosses.html'

```

```

model = Lost
def get_queryset(self):
    part = self.request.GET.get('participant')
    losts = Lost.objects.filter(participant = part)
    losts = losts.filter(lostfound = True)
    losts = losts.order_by('-found')
    return losts

def get_context_data(self, **kwargs):
    dcontext = super(ParticipantLostsView,
self).get_context_data(**kwargs)
    participant = self.request.GET.get('participant')
    dcontext['participant'] = participant
    return dcontext

class ParticipantFindsView(LoginRequiredMixin, ListView):
    template_name = 'partfinds.html'
    model = Lost

def get_queryset(self):
    part = self.request.GET.get('participant')
    losts = Lost.objects.filter(participant = part)
    losts = losts.filter(lostfound = False)
    losts = losts.order_by('-found')
    return losts

def get_context_data(self, **kwargs):
    dcontext = super(ParticipantFindsView,
self).get_context_data(**kwargs)
    participant = self.request.GET.get('participant')
    dcontext['participant'] = participant
    return dcontext

class SearchFindsView(ListView):
    template_name = 'searchfinds.html'
    model = Lost

def get_queryset(self):
    losts = Lost.objects.filter(lostfound = True)
    losts = losts.order_by('-found')
    animal = self.request.GET.get('animal')
    if animal:
        animal = Animal.objects.get(id = animal)
        losts = losts.filter(breed__animal__id =
animal)

    breed = self.request.GET.get('breed')

```

```

    if breed:
        breed = Breed.objects.get(id = breed)
        losts = losts.filter(breed__id = breed)
    city = self.request.GET.get('city')
    if city:
        city = City.objects.get(id = city)
        losts = losts.filter(foundcity__id = city)
    datel = self.request.GET.get('date1')
    if datel:
        datel = datetime.datetime.strptime(datel,
'%d.%m.%y')

        losts = losts.filter(found__gte = datel)
    date2 = self.request.GET.get('date2')
    if date2:
        date2 = datetime.datetime.strptime(date2,
'%d.%m.%y')

        losts = losts.filter(found__lte = date2)
    microchip = self.request.GET.get('microchip')
    if microchip:
        losts = losts.filter(microchip = microchip)
    gender = self.request.GET.get('gender')
    if gender:
        losts = losts.filter(gender = gender)
    charvalues = self.request.GET.get('charvalues')
    vals = charvalues.split(',')
    vs = []
    for val in vals:
        v = val.split(' ')
        char = v[0]
        chval = v[1]
        vs.append(Charvalue.objects.get(vcalled =
chval, characteristic__chcalled = v))
    rlosts = losts
    for lost in losts:
        for v in vs:
            r = LostCharvalue.objects.filter(lost =
lost, charvalue = v)

            if not r:
                rlosts.delete(lost)
    rlosts = rlosts.filter(lostfound = False)
    return rlosts

def get_context_data(self, **kwargs):
    dcontext = super(SearchFindView,
self).get_context_data(**kwargs)
    animal = self.request.GET.get('animal')

```

```

dcontext['animal'] = animal
breed = self.request.GET.get('breed')
dcontext['breed'] = breed
city = self.request.GET.get('city')
dcontext['city'] = city
date1 = self.request.GET.get('date1')
dcontext['date1'] = date1
date2 = self.request.GET.get('date2')
dcontext['date2'] = date2
microchip = self.request.GET.get('microchip')
dcontext['microchip'] = microchip
gender = self.request.GET.get('gender')
dcontext['gender'] = gender
charvalues = self.request.GET.get('charvalues')
dcontext['charvalues'] = charvalues
return dcontext

class SearchAdFindsView(ListView):
    template_name = 'searchadfinds.html'
    model = Lost

    def get_queryset(self):
        losts = Lost.objects.filter(lostfound = True)
        losts = losts.order_by('-found')
        find = self.request.GET.get('find')
        find = Lost.objects.get(id = find)
        losts = losts.filter(breed__id = find.breed)
        losts = losts.filter(foundcity__id =
find.foundcity)
        losts = losts.filter(found__gte = find.found)
        losts = losts.filter(microchip = find.microchip)
        losts = losts.filter(gender = find.gender)
        charvalues = find.charvalues
        losts = losts.filter(lostfound = False)
        rlosts = losts
        for lost in losts:
            for v in charvalues:
                r = LostCharvalue.objects.filter(lost =
lost, charvalue = v)
                if not r:
                    rlosts.delete(lost)
        return rlosts

    def get_context_data(self, **kwargs):
        dcontext = super(SearchAdFindsView,
self).get_context_data(**kwargs)

```

```

lost = self.request.GET.get('lost')
dcontext['lost'] = lost
return dcontext

```

Б.3 Текст файла applicationviews.py

```

from django.views.generic.edit import CreateView, FormMixin
from django.views.generic import ListView, UpdateView
from django.views.generic.detail import DetailView
from .models import Application, Kennel, Participant
import datetime
from django.urls import reverse_lazy

class ApplicationView (FormMixin, DetailView):
    template_name = 'application.html'
    model = Application

    def form_valid(self, form):
        if form.result == 'A':
            form.result = 'B'
        form.save()
        return super(ApplicationView,
self).form_valid(form)

class ApplicationCreate(CreateView):
    model = Application
    fields = ['lostname', 'lost', 'searched', 'note']
    template_name = 'createapplication.html'

    def get_success_url(self):
        return reverse_lazy('application-new',
kwargs={'pk': self.object.pk})

    def form_valid(self, form):
        part = get_object_or_404(Participant,
id=self.request.user)
        self.participant = part
        return super().form_valid(form)

class ApplicationUpdate(UpdateView):
    model = Application
    fields = ['lostname', 'lost', 'searched', 'note',
'result']
    template_name = 'updateapplication.html'

```

```

    def get_success_url(self):
        return reverse_lazy('application-update',
kwargs={'pk': self.object.pk})

class KennelApplicationsView(ListView):
    template_name = 'kennelapplications.html'
    model = Application
    paginate_by = 20
    def get_queryset(self):
        ken = self.request.GET.get('kennel')
        ken = Kennel.objects.get(id = ken)
        apps = Application.objects.filter(lost__kennel__id
= ken)

        res = self.request.GET.get('result')
        apps = apps.filter(result = res)
        datel = self.request.GET.get('date1')
        if datel:
            datel = datetime.datetime.strptime(datel,
'%d.%m.%y')

            apps = apps.filter(done__gte = datel)
        date2 = self.request.GET.get('date2')
        if date2:
            date2= datetime.datetime.strptime(date2,
'%d.%m.%y')

            apps = apps.filter(done__lte = date2)
        return apps

    def get_context_data(self, **kwargs):
        dcontext = super(KennelLostsView,
self).get_context_data(**kwargs)
        datel = self.request.GET.get('date1')
        dcontext['start'] = datel
        date2 = self.request.GET.get('date2')
        dcontext['end'] = date2
        ken = self.request.GET.get('kennel')
        dcontext['kennel'] = ken
        res = self.request.GET.get('result')
        dcontext['result'] = res
        return dcontext

class YourApplicationsView(LoginRequiredMixin, ListView):
    template_name = 'yourapps.html'
    model = Application

    def get_queryset(self):
        you = self.request.user

```

```

        apps = Application.objects.filter(participant =
you)
        apps = apps.order_by('-id')
        return apps

class ToYouApplicationsView(LoginRequiredMixin, ListView):
    template_name = 'toyourapps.html'
    model = Application
    def get_queryset(self):
        you = self.request.user
        apps =
Application.objects.filter(lost__participant = you)
        apps = apps.order_by('-id')
        return apps

```

Б.4 Текст файла knockviews.py

```

from django.views.generic.edit import CreateView, FormMixin
from django.views.generic import ListView, UpdateView
from django.views.generic.detail import DetailView
from .models import Knock, Participant
import datetime
from django.urls import reverse_lazy

class KnockView (FormMixin, DetailView):
    template_name = 'knock.html'
    model = Knock

    def form_valid(self, form):
        if form.result == 'A':
            form.result = 'B'
        form.save()
        return super(KnockView, self).form_valid(form)

class KnockCreate(CreateView):
    model = Knock
    fields = ['lost_to', 'lost']
    template_name = 'createknock.html'

    def get_success_url(self):
        return reverse_lazy('knock-new', kwargs={'pk':
self.object.pk})

```

```

        def form_valid(self, form):
            part = get_object_or_404(Participant,
id=self.request.user)
            self.participant = part
            return super().form_valid(form)

class KnockUpdate(UpdateView):
    model = Knock
    fields = ['lost_to', 'lost', 'result']
    template_name = 'updateknock.html'

    def get_success_url(self):
        return reverse_lazy('application-update',
kwargs={'pk': self.object.pk})

class KennelKnocksView(ListView):
    template_name = 'kennelknocks.html'
    model = Knock
    paginate_by = 20

    def get_queryset(self):
        ken = self.request.GET.get('kennel')
        ken = Kennel.objects.get(id = ken)
        knocks = Knock.objects.filter(Q(lost__kennel__id =
ken) | Q(lost_to__kennel__id = ken))
        res = self.request.GET.get('result')
        knocks = knocks.filter(result = res)
        datel = self.request.GET.get('datel')
        if datel:
            datel = datetime.datetime.strptime(datel,
'%d.%m.%y')
            knocks = knocks.filter(done__gte = datel)
        date2 = self.request.GET.get('date2')
        if date2:
            date2= datetime.datetime.strptime(date2,
'%d.%m.%y')
            knocks = knocks.filter(done__lte = date2)
        return knocks

    def get_context_data(self, **kwargs):
        dcontext = super(KennelKnocksView,
self).get_context_data(**kwargs)
        datel = self.request.GET.get('datel')
        dcontext['start'] = datel
        date2 = self.request.GET.get('date2')

```

```

        dcontext['end'] = date2
        ken = self.request.GET.get('kennel')
        dcontext['kennel'] = ken
        res = self.request.GET.get('result')
        dcontext['result'] = res
        return dcontext

class YourKnocksView(LoginRequiredMixin, ListView):
    template_name = 'yourknocks.html'
    model = Knock

    def get_queryset(self):
        you = self.request.user
        knocks = Knock.objects.filter(participant = you)
        knocks = knocks.order_by('-done')
        return knocks

class YourKnockedView(LoginRequiredMixin, ListView):
    template_name = 'yourknocked.html'
    model = Knock

    def get_queryset(self):
        you = self.request.user
        knocks = Knock.objects.filter(Q(lost__participant
= you) | Q(lost_to__participant = you))
        knocks = knocks.order_by('-done')
        return knocks

```

Б.5 Текст файла meetingviews.py

```

from django.views.generic.edit import CreateView
from django.views.generic import ListView, UpdateView
from django.views.generic.detail import DetailView
from .models import Knock, Participant
import datetime
from django.urls import reverse_lazy

class MeetingView (DetailView):
    template_name = 'meeting.html'
    model = Meeting

class MeetingCreate(CreateView):
    model = Meeting
    fields = ['note', 'lost', 'foundcity', 'found']
    template_name = 'createmeeting.html'

```

```

        def get_success_url(self):
            return reverse_lazy('meeting-new', kwargs={'pk':
self.object.pk})

        def form_valid(self, form):
            part = get_object_or_404(Participant,
id=self.request.user)
            self.participant = part
            return super().form_valid(form)

class MeetingUpdate(UpdateView):
    model = Meeting
    fields = ['note', 'lost', 'foundcity', 'found']
    template_name = 'updatemeeting.html'

    def get_success_url(self):
        return reverse_lazy('meeting-update',
kwargs={'pk': self.object.pk})

class YourMeetingsView(LoginRequiredMixin, ListView):
    template_name = 'yourmeetings.html'
    model = Meeting

    def get_queryset(self):
        you = self.request.user
        meetings = Meeting.objects.filter(participant =
you)

        meetings = meetings.order_by('-founded')
        return meetings

class ToYouMeetingsView(LoginRequiredMixin, ListView):
    template_name = 'toyoumeetings.html'
    model = Meeting

    def get_queryset(self):
        you = self.request.user
        meetings =
Meeting.objects.filter(lost__participant = you)
        meetings = meetings.order_by('-founded')
        return meetings

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Програмне забезпечення розшуку загублених тварин

Lost Animal Search Software

Виконав

студент групи КНТ-219сп

М. В. Чумак

Керівник

к.т.н., доц.

О. О. Степаненко

2022

Рисунок В.1 – Слайд № 1

Об'єкт роботи – процес розшуку загублених тварин.

Предмет роботи – програмне забезпечення розшуку загублених тварин.

Мета роботи – розробити програмне забезпечення розшуку загублених тварин для сприяння возз'єднанню загублених тварин зі своїми власниками.

Завдання роботи:

- аналіз програмних аналогів;
- визначення засобів проектування і розробки програми;
- проектування програми;
- реалізація і налагодження програми.

2

Рисунок В.2 – Слайд № 2

Порівняння аналогів			
Функція	RSPCA Victoria	Petco LoveLost	LostAnimal SearchSoftware
Пошук загублених тварин	Відсутній	Присутній	Присутній
Пошук власників знайдених тварин	Присутній	Присутній	Присутній
Внесення даних загубленої тварини	Відсутній	Присутній	Присутній
Внесення даних знайденої тварини	Відсутній	Присутній	Присутній
Інформування про власництво тварини	Відсутній	Присутній	Присутній
Повідомлення про відповідність між оголошеннями	Відсутній	Відсутній	Присутній
Повідомлення про тимчасову зустріч із твариною	Відсутній	Відсутній	Присутній
Знаходження автоматичної відповідності між оголошеннями	Відсутній	Присутній	Присутній

3

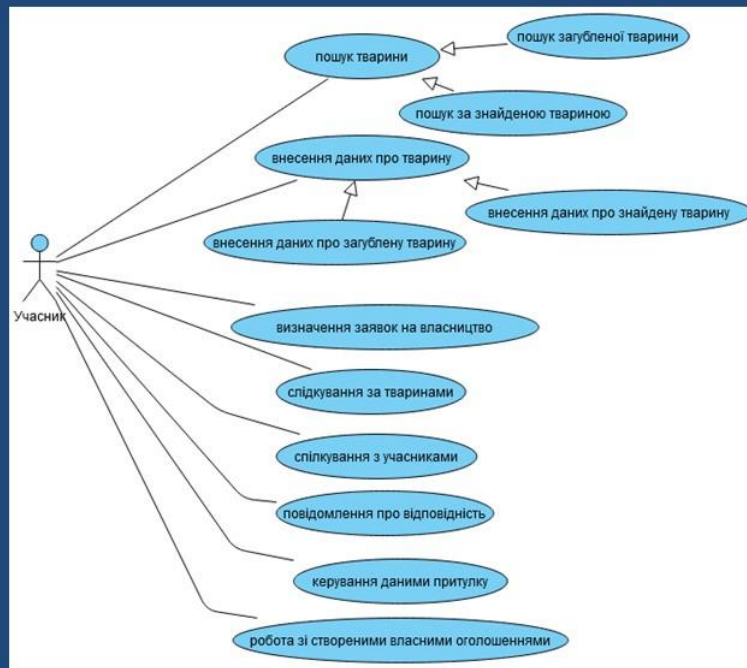
Рисунок В.3 – Слайд № 3

Вибір мови програмування		
Показник	Python	PHP
Тип мови	Мова загального призначення	Мова для веброзробки
Використання	Веброзробка, штучний інтелект, застосунки для стаціонарних систем	Веброзробка
Попередньо збудовані бібліотеки широкого спрямування	Велика кількість	Набагато менше
Підтримка основних баз даних	Так	Так
Читабельність	Краща	Гірша

4

Рисунок В.4 – Слайд № 4

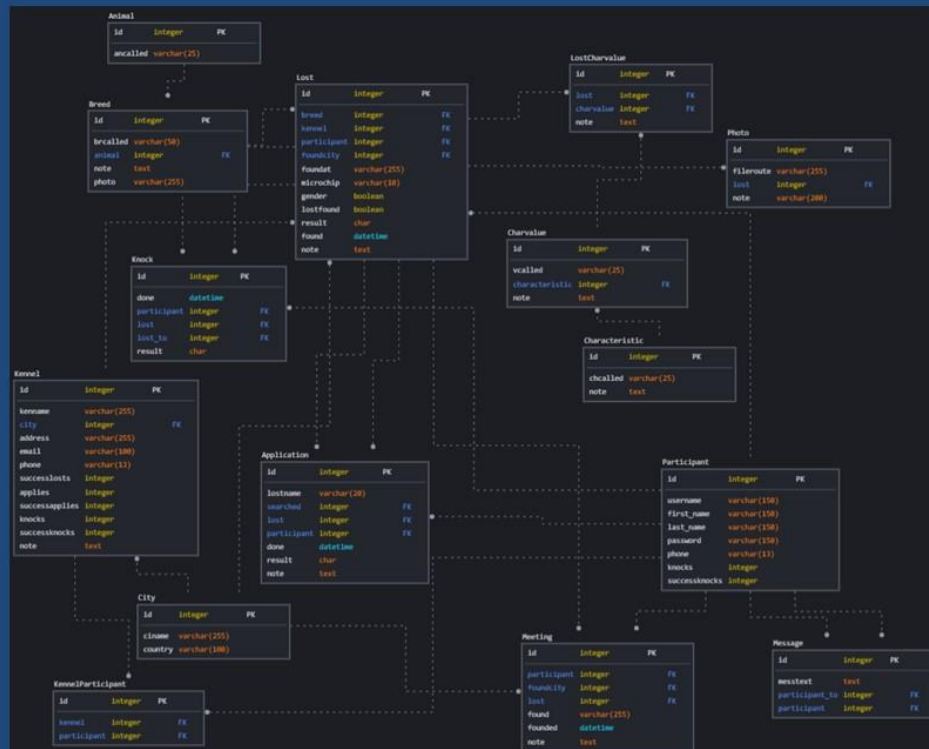
Діаграма прецедентів



5

Рисунок В.5 – Слайд № 5

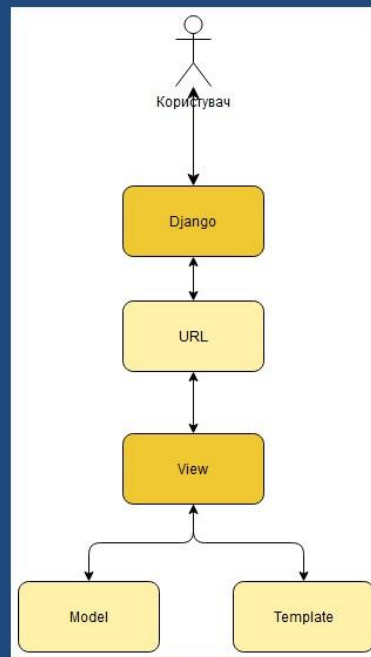
Схема бази даних



6

Рисунок В.6 – Слайд № 6

Шаблон проєктування Model-View-Template



7

Рисунок В.7 – Слайд № 7

Діаграми станів оголошення і заявки



8

Рисунок В.8 – Слайд № 8

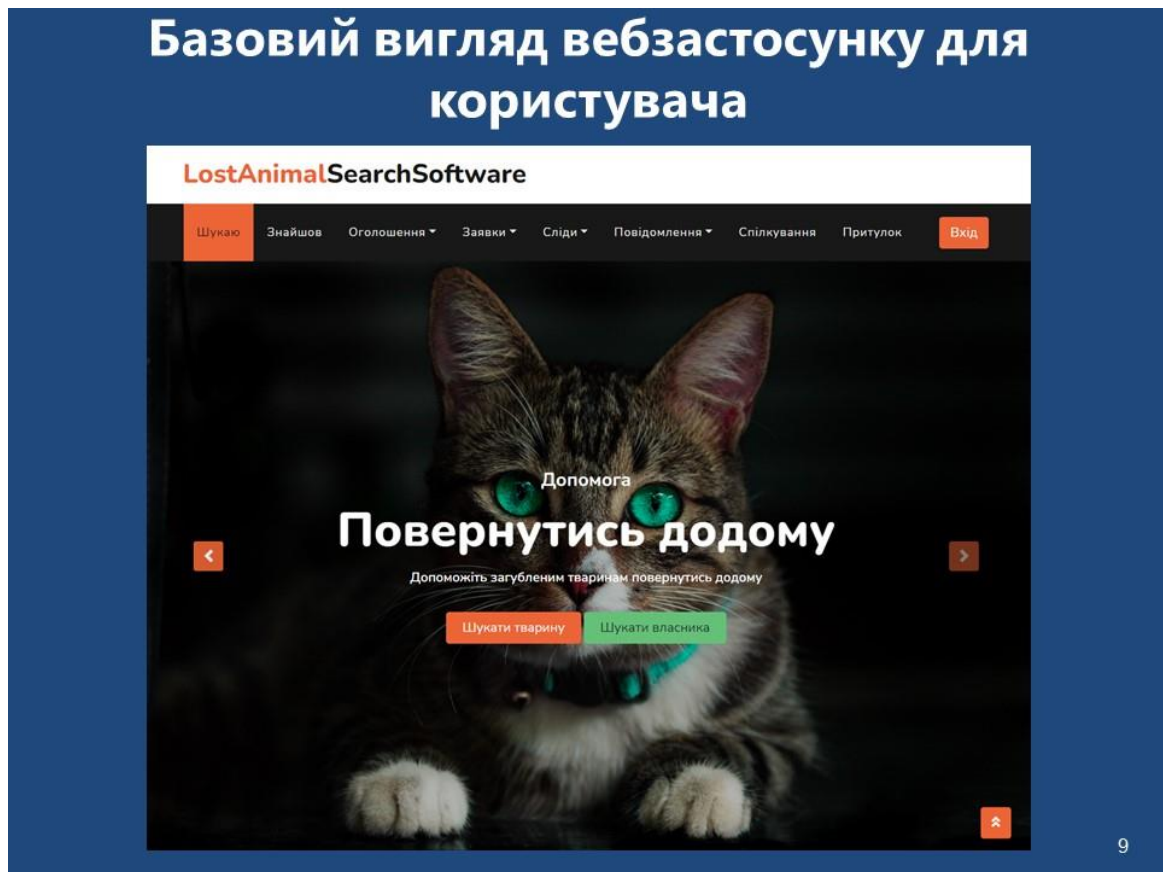


Рисунок В.9 – Слайд № 9

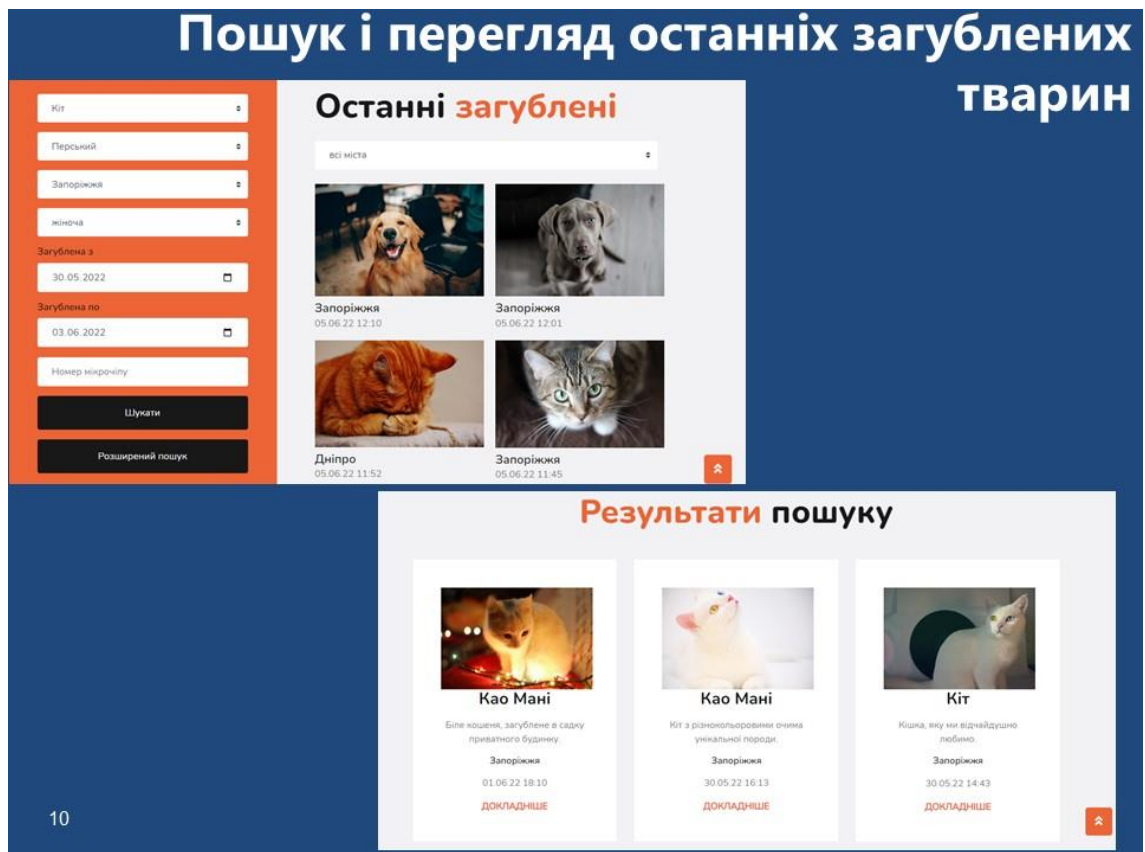


Рисунок В.10 – Слайд № 10

Оголошення і повідомлення про відповідність

Знайдена тварина

Європейський короткошерстний кіт

Рудий кіт з білими смужками. Тепер, після лікування, дуже грайливий, активний. Знайдений на поверсі розбитої багатоповерхівки.

- ✓ Харків
- ✓ Салтівка
- ✓ 22.05.22 11:07
- ✓ чоловіча стать
- ✓ мікрочіпу немає
- ✓ Притулок: Черкаський будинок допомоги
- ✓ Користувач: Ivan
- ✓ Колір: рудий
- ✓ Очі: карі
- ✓ Шерсть: коротка
- ✓ Хвіст: середній

Моя тварина
Повідомити про подібність



Повідомлення про відповідність



✓ Британський кіт
✓ чоловічий
✓ Запоріжжя
✗ 01.06.22 11:19
Скасувати



✓ Британський кіт
✓ чоловічий
✓ Запоріжжя
✗ 01.06.22 19:47
Підтвердити

Рисунок В.11 – Слайд № 11

Висновки

Основним результатом роботи є розроблена програма, а для цього у процесі розробки було виконано аналіз проблеми з оглядом існуючих програмних аналогів, визначенням вимог до програми, вибором засобів розробки, включаючи мови Python, JavaScript, HTML, CSS, вебфреймворк Django, систему керування базами даних MySQL, виконано проєктування бази даних, логіки роботи програми. Логіка функціонування програми, що базується на зміні станів основних сутностей, які включають оголошення, заявку та повідомлення, була представлена відповідними діаграмами станів. Окрім того описано реалізовані одиниці вебзастосунків на основі логіки першочергового створення коду моделей для роботи з даними та в подальшому – представлень. Описано основні принципи роботи з інтерфейсами програми користувачів.

Програмне забезпечення фактично в результаті на практиці може використовуватися для взаємодії людей, які є власниками тварин і загубили їх, і з людьми, які їх знайшли, а також з притулками тварин.

Рисунок В.12 – Слайд № 12