

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПОШУКУ

(назва теми)

ГОСПОДАРІВ ЗНАЙДЕНИХ РЕЧЕЙ

SOFTWARE TO SEARCH FOR OWNERS OF FOUND ITEMS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-220сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ПЛАХОТНИК А.Ю.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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.
Сергій СУББОТІН
« » 2023 року

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

ПЛАХОТНИК Анастасії Юхимівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення пошуку господарів знайдених речей. Software to Search for Owners of Found Items

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 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. Реалізація програми. 4. Експлуатація та тестування програми.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	БЕЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок 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)

_____ Анастасія ПЛАХОТНИК
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Степан СКРУПСЬКИЙ
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 94 с., 22 рис., 2 табл., 3 дод., 9 джерел.

ЗНАХІДКА, ВТРАТА, ОГОЛОШЕННЯ, РЕЧІ, ОПИТУВАННЯ, ПРОГРАМНА СИСТЕМА.

Об'єктом роботи є процес пошуку господарів знайдених речей і господарями втрачених речей. Предметом роботи є програмне забезпечення пошуку господарів знайдених речей. Мета роботи – розробити програмне забезпечення пошуку господарів знайдених речей для сприяння підвищенню результативності даного процесу.

Матеріали, методи та технічні засоби: мова Python, фреймворк для підтримки веброзробки Django, система керування базами даних MySQL.

Результати. Виявлені базові функції та необхідні функціональні особливості для програмного забезпечення пошуку господарів знайдених речей і одночасно господарями власних речей, забезпечено моделювання, структуризацію даних і програмної системи та її реалізацію.

Висновки. Існування простого і зручного у використанні інструмента для пошуку господарів знайдених речей на практиці може сприяти тому, щоб люди активніше намагалися допомогти тим, хто загубив власні речі, які можуть бути цінними взагалі тільки господарю. Тому наявність такої платформи пошуку сприяє позитивному соціальному ефекту, довірі в суспільстві в певній мірі. Внесення чіткої інформації про виявлену знахідку, підтримка авторів оголошень щодо виявлення шахраїв, визначення повного ланцюжка переміщень при виявленні знахідки в транспорті та врахування цього ланцюжка в пошуку потенційно мають важливий ефект для досягнення результативності всього пошуку.

Галузь використання – пошук втрачених речей.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 94 pages, 22 figures, 2 tables, 3 appendixes, 9 sources.

FOUND, LOST, ADVERTISEMENT, ITEMS, SURVEY, SOFTWARE.

The object of the work is the process of finding the owners of found items and the owners of lost items. The subject of the work is software for finding owners of found items. The purpose of the work is to develop software for finding owners of found items to help increase the effectiveness of this process.

Materials, methods and technical means: Python language, Django web development support framework, MySQL database management system.

The results. The basic functions and necessary functional features for the software to search for the owners of found items and at the same time for the lost items by its owners have been identified. Modeling, structuring of data and the implementation of the software system have been provided.

Conclusions. The existence of a simple and easy-to-use tool for finding the owners of found items in practice can contribute to people more actively trying to help those who have lost their own items, which may be valuable in general only to its owner. Therefore, the presence of such a search platform contributes to a positive social effect, trust in society to a certain extent. Providing clear information about the situation, supporting the authors of ads to identify fraudsters, defining the complete chain of movements when a find is made in transport and taking this chain into account in the search have the potential important effect on achieving the effectiveness of the entire search.

Field of use – searching for lost things.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Аналіз програмних засобів пошуку господарів знайдених речей	10
1.2 Визначення концепції розроблюваної програми	13
1.3 Висновки за розділом 1 та постановка завдань роботи	14
2 Розробка архітектури програми.....	16
2.1 Вибір технологій розробки	16
2.2 Моделювання вимог до програми	17
2.3 Проєктування бази даних	19
2.4 Висновки за розділом 2	33
3 Реалізація програми	34
3.1 Порядок виконання пошуку господарів знайдених речей.....	34
3.2 Програмна реалізація вебсистеми	36
3.3 Висновки за розділом 3	50
4 Експлуатація та тестування програми	51
4.1 Призначення програми	51
4.2 Умови виконання програми	51
4.3 Виконання і тестування програми.....	51
4.4 Висновки за розділом 4	56
Висновки	57
Перелік джерел посилання	59
Додаток А Технічне завдання	60
Додаток Б Текст програми	64
Додаток В Слайди презентації.....	88

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

URL – Uniform Resource Locator;

БД – база даних.

ВСТУП

Якщо людина загубила свої речі, це ще не означає, що вона втратила їх назавжди. Однак у випадках, коли речі не були вкрадені, а просто десь залишені та потім кимось підібрані, не так просто повернути їх власнику. Для цього потрібен програмний інструмент, який дозволить знайти людині, яка загубила речі, людину, яка їх знайшла. Саме від результативності цього процесу залежить успішність процесу повернення. Це передбачає потужні інструменти пошуку, а також перевірку власності. Саме для надання таких інструментів має бути створено розробку в роботі.

Розробка в роботі має зокрема враховувати специфічні ситуації, коли це може трапитися і за яких передумов цей пошук має виконуватися.

Однією з таких передумов є шахрайські задуми, коли людина, побачивши речі в оголошенні, вирішує неправомірно ними заволодіти. З розвитком як ботів, так і штучного інтелекту кількість таких запитів може збільшуватися, тому в певних ситуаціях може бути дуже корисною автоматична перевірка заявників. Це ж може бути корисним і у випадках невеликої кількості заявок, але для людей, які просто не мають достатньо часу, щоб виявляти шахраїв, але знайшовши речі, бажають повернути їх власнику.

Ще однією передумовою є ускладнення пошуку тоді, коли речі були загублені під час переїзду. Тобто якщо речі загублені в потязі, автобусі, то важко визначити місце пошуку. Власник речей міг виїхати в одному місці, вийти в іншому, залишивши речі в потязі або автобусі, а знайдені вони були в третьому, при тому що людина, яка знайшла речі, вийшла з транспорту взагалі в четвертому місці, подолавши ще декілька між ними. Тож де саме виконувати пошук, може бути невідомим. Для таких ситуацій варто відслідкувати весь маршрут транспорту. Це потрібно враховувати в параметрах оголошення і потім під час пошуку.

Усі ці передумови мають бути враховані і забезпечені під час досягнення мети роботи, що полягає в розробці програмного забезпечення пошуку господарів знайдених речей для сприяння підвищенню результативності даного процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз програмних засобів пошуку господарів знайдених речей

iLost [1] (рис. 1.1) є одним з найвідоміших вебдодатків, які використовуються для пошуку господарів знайдених речей. Посилання на нього зустрічаються зокрема на вебсайтах деяких центрів з пошуку втрат і знахідок транспортних вузлів.

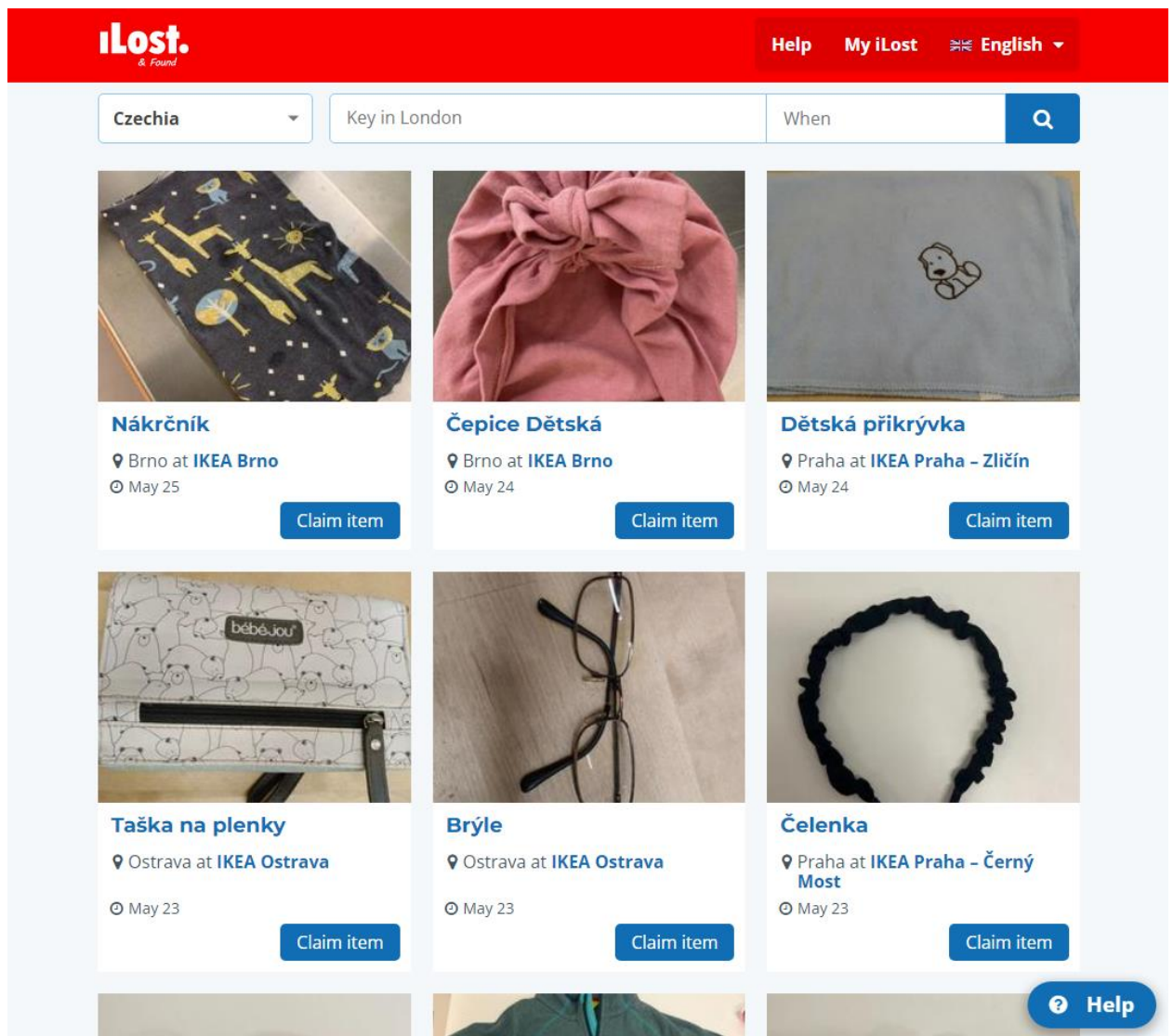


Рисунок 1.1 – Вебдодаток iLost

iLost має на основній сторінці просту форму пошуку, яка включає визначення втрати (немає окремих показників, тільки загальне текстове поле)

та країну пошуку. На сторінці результатів можна також задати дату, коли було загублено речі. За кожним оголошенням можна створити заявку на власництво речі. Також на початковій сторінці присутня кнопка, яка дозволяє визначити опис навпаки знайденої речі. Така форма включає текстовий опис речі, фотографію, адресу, місто, країну, де було знайдено річ, ім'я користувача, який знайшов річ, номер телефону та електронну адресу.

Даний вебсайт не дозволяє об'єднувати знайдені речі, потрібно створювати окремі оголошення про кожну річ.

Lost and found networks [2] (рис. 1.2) має більш широкую кількість засобів пошуку, об'єднуючи всі в єдиний пошук.

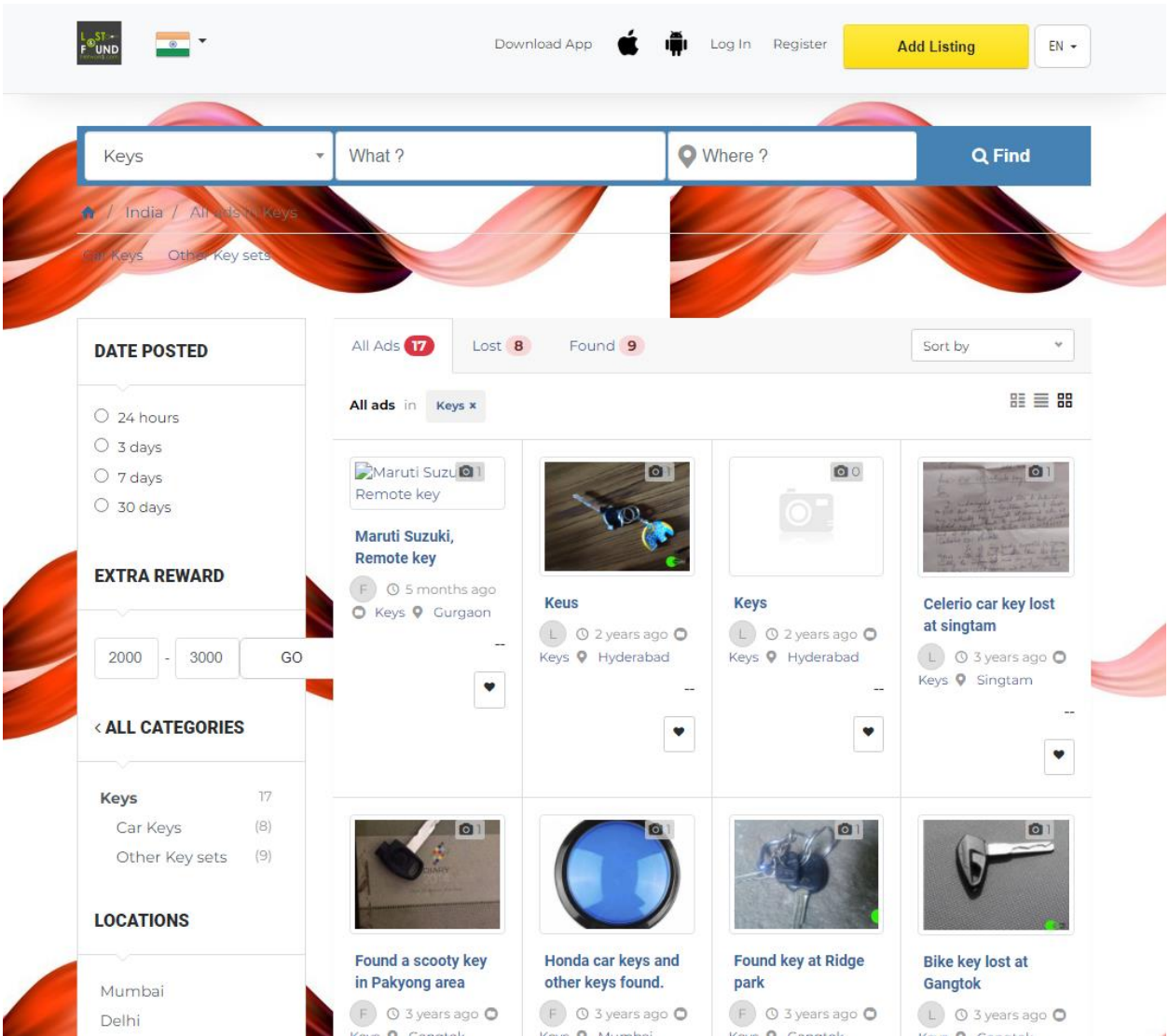
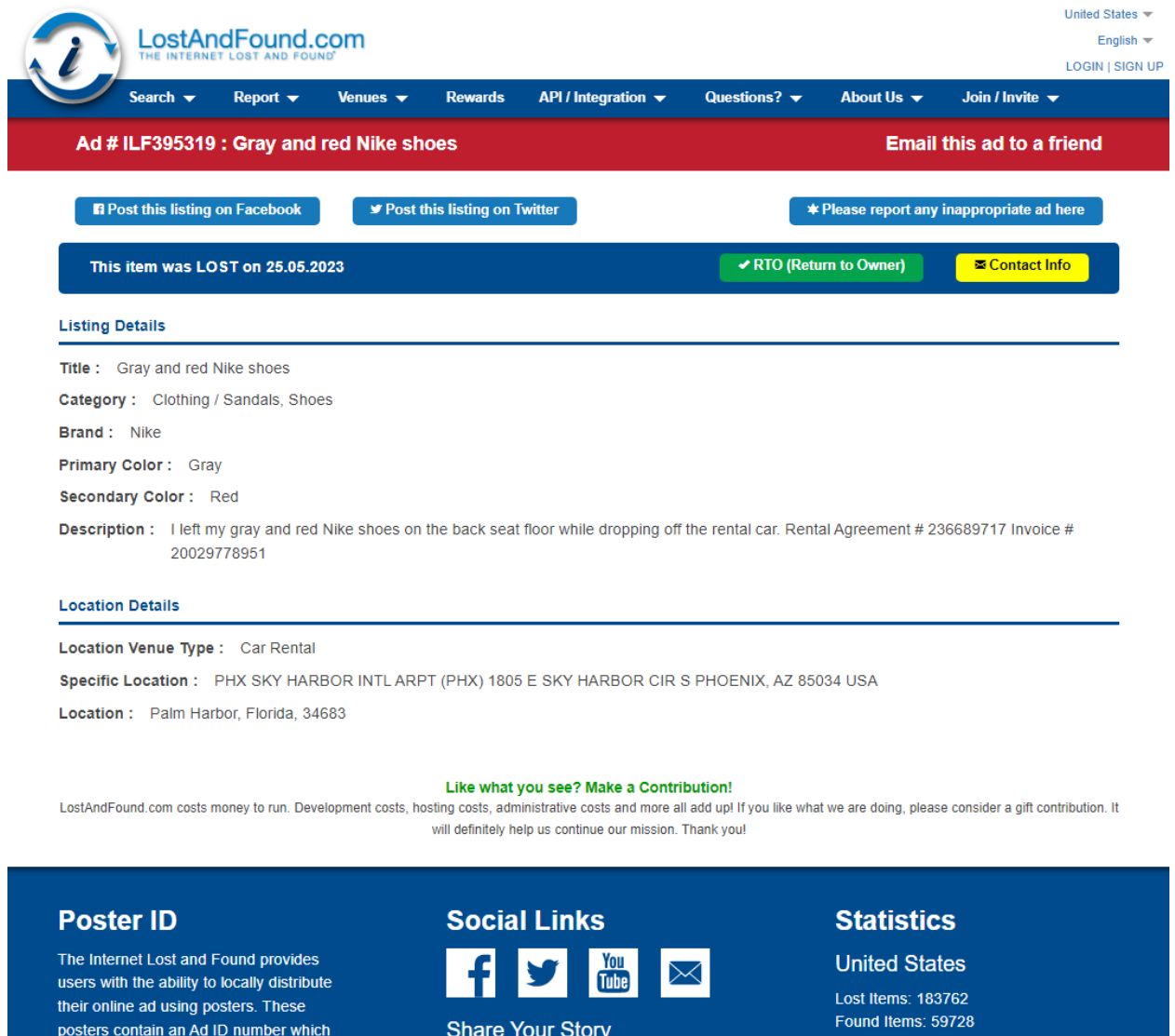


Рисунок 1.2 – Вебдодаток Lost and found networks

Але на відміну від iLost результати мають три вкладки: всі, знахідки та втрати. Пошук виконується за категоріями, за періодом втрати або знахідки, за місцем втрати або знахідки. За кожним оголошенням приводиться одна фотографія, опис, набір тегів.

Вебдодаток LostAndFound [3] (рис. 1.3) має окремий пошук для знахідок та окремий пошук для втрат.



The screenshot displays the LostAndFound.com website interface. At the top, there is a navigation bar with a logo and links for Search, Report, Venues, Rewards, API / Integration, Questions?, About Us, and Join / Invite. A red banner below the navigation bar shows the ad title 'Ad # ILF395319 : Gray and red Nike shoes' and a link to 'Email this ad to a friend'. Below the banner are buttons for 'Post this listing on Facebook', 'Post this listing on Twitter', and 'Please report any inappropriate ad here'. A blue bar indicates 'This item was LOST on 25.05.2023' with buttons for 'RTO (Return to Owner)' and 'Contact Info'. The main content area is divided into 'Listing Details' and 'Location Details'. The 'Listing Details' section includes fields for Title, Category, Brand, Primary Color, Secondary Color, and Description. The 'Location Details' section includes fields for Location Venue Type, Specific Location, and Location. At the bottom, there is a blue footer with three columns: 'Poster ID' (explaining the service), 'Social Links' (with icons for Facebook, Twitter, YouTube, and Email), and 'Statistics' (showing 'United States', 'Lost Items: 183762', and 'Found Items: 59728').

United States
 English
 LOGIN | SIGN UP

Search Report Venues Rewards API / Integration Questions? About Us Join / Invite

Ad # ILF395319 : Gray and red Nike shoes Email this ad to a friend

Post this listing on Facebook Post this listing on Twitter Please report any inappropriate ad here

This item was LOST on 25.05.2023 RTO (Return to Owner) Contact Info

Listing Details

Title : Gray and red Nike shoes
Category : Clothing / Sandals, Shoes
Brand : Nike
Primary Color : Gray
Secondary Color : Red
Description : I left my gray and red Nike shoes on the back seat floor while dropping off the rental car. Rental Agreement # 236689717 Invoice # 20029778951

Location Details

Location Venue Type : Car Rental
Specific Location : PHX SKY HARBOR INTL ARPT (PHX) 1805 E SKY HARBOR CIR S PHOENIX, AZ 85034 USA
Location : Palm Harbor, Florida, 34683

Like what you see? Make a Contribution!
 LostAndFound.com costs money to run. Development costs, hosting costs, administrative costs and more all add up! If you like what we are doing, please consider a gift contribution. It will definitely help us continue our mission. Thank you!

Poster ID
 The Internet Lost and Found provides users with the ability to locally distribute their online ad using posters. These posters contain an Ad ID number which

Social Links
 Share Your Story

Statistics
 United States
 Lost Items: 183762
 Found Items: 59728

Рисунок 1.3 – Вебдодаток LostAndFound

Кожне оголошення містить назву, перелік категорій, назву виробника, кольори, опис та деталі місця, де було втрачено або знайдено речі. Ці деталі включають тип місця, повну адресу.

Форма пошуку включає ключові слова, категорії, штат, місто, індекс.

1.2 Визначення концепції розроблюваної програми

Усі проаналізовані програмні засоби були порівняні та виділено їх основні можливості відносно особливостей, які винесені в розробку (табл. 1.1).

Таблиця 1.1 – Порівняння програмних засобів пошуку господарів знайдених речей

Характеристика	iLost	Lost and found networks	LostAndFound	FIAPP
Об'єднання декількох речей (оголошень) в одне оголошення	—	—	—	+
Пошук втрат і знахідок	—	+	+	+
Маршрути для знахідок у транспорті	—	—	—	+
Підтримка визначення доставки	+	—	—	+
Перевірка шахрайства	—	—	—	+

Ні iLost, ні Lost and found networks, ні LostAndFound не мають можливості об'єднувати знайдені речі в одне оголошення. Іноді буває так, що

речі знайдені разом, але для спрощення пошуку, а можливо і перевірки на шахрайство важливим є об'єднання цих речей разом. Окрім того при цьому не потрібно виконувати окремі описи з визначенням місця знахідки кожен раз заново. У розроблюваній програмі вирішено надати таку можливість, що в певних випадках може бути корисним.

Як і було описано раніше, важливими засобами є визначення маршрутів знахідок у транспорті та перевірка шахрайства. Аналіз дозволив встановити, що таких можливостей у аналогів немає.

Тільки iLost має можливість створення заявки на власництво речей і визначення параметрів доставки, що для підприємств (готелів, наприклад) може бути як раз важливим для спрощення подальших процедур і уникнення необхідності особистого контактування.

Пошук втрат і знахідок у Lost and found networks реалізується таким чином, що форма пошуку єдина, результати відображаються разом, але з них можна окремо відібрати знахідки, а окремо – втрати. LostAndFound має окремий пошук для знахідок і втрат. Саме другий підхід вирішено використовувати в розробці FIAPP.

1.3 Висновки за розділом 1 та постановка завдань роботи

У першому розділі виконано виділення програмних засобів пошуку господарів знайдених речей: iLost, Lost and found networks, LostAndFound. Проаналізовано наявні та відсутні у них можливості. Запропоновано в програмному додатку, що розробляється, представити засоби визначення маршрутів для знахідок у транспорті, перевірки шахрайства, що є важливим для людей, які знайшли речі, та готові повернути їх господарю, але не готові витратити час на хибні заявки, кількість яких може збільшитися значно, якщо речі є цінними.

Завдання, визначені на роботу:

- аналіз програмних засобів пошуку господарів знайдених речей і визначення концепції програми;
- проєктування програмного забезпечення пошуку господарів знайдених речей;
- реалізація вебдодатку пошуку господарів знайдених речей.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір технологій розробки

Python [4]-[5] і Ruby [6] розглянуті як можливість для розробки програмного забезпечення, враховуючи при цьому, що це має бути веброзробка. Загалом обидві мови мають достатньо високий рівень для розв'язання цієї проблеми (табл. 2.1), однак за сукупністю переваг і недоліків, враховуючи, що Python поступається за швидкістю, але переважає Ruby за можливостями наслідування, вимогами до пам'яті, простішим процесом відлагодження, саме Python використано в подальшому для розробки. Найпотужнішим фреймворком для спрощення веброзробки є Django [7], який використано в подальшому.

Таблиця 2.1 – Порівняння мов Ruby і Python

Характеристика	Ruby	Python
Основна парадигма програмування	Об'єктно-орієнтована	Об'єктно-орієнтована
Підтримка множинного наслідування	Ні	Так
Швидкість виконання коду	Вища	Менша
Вимоги до пам'яті	Більші	Менші
Знаходження помилок в коді	Складніше	Простіше
Фреймворк для спрощення веброзробки	Ruby on Rails	Django

2.2 Моделювання вимог до програми

Будь-який користувач програми має можливість скористатися наступними функціями:

- публікація оголошення про знаходження речей;
- модифікація оголошення про знаходження речей;
- деактивація оголошення про знаходження речей;
- публікація оголошення про втрату речей;
- модифікація оголошення про втрату речей;
- деактивація оголошення про втрату речей;
- пошук речей серед знайдених;
- пошук речей серед втрачених;
- заявлення про власництво речей;
- відхилення заявки про власництво речей;
- підтвердження власництва речей;
- відтворення переліку заявок про власництво за знахідкою;
- створення опитування для перевірки заявника за знахідкою;
- перевірка власництва речі шляхом опитування;
- відтворення переліку всіх знахідок користувача;
- відтворення переліку всіх втрат користувача;
- відправлення повідомлення користувачу;
- заявлення про знаходження речей;
- відхилення заявки про знаходження речей;
- підтвердження заявки про знаходження речей;
- відтворення переліку заявок за втратою;
- заявлення про відповідність знахідки втраті;
- підтвердження відповідності знахідки втраті;
- відхилення відповідності знахідки втраті;
- замовлення доставки власних речей;
- створення зустрічі для повернення речей (рис. 2.1).

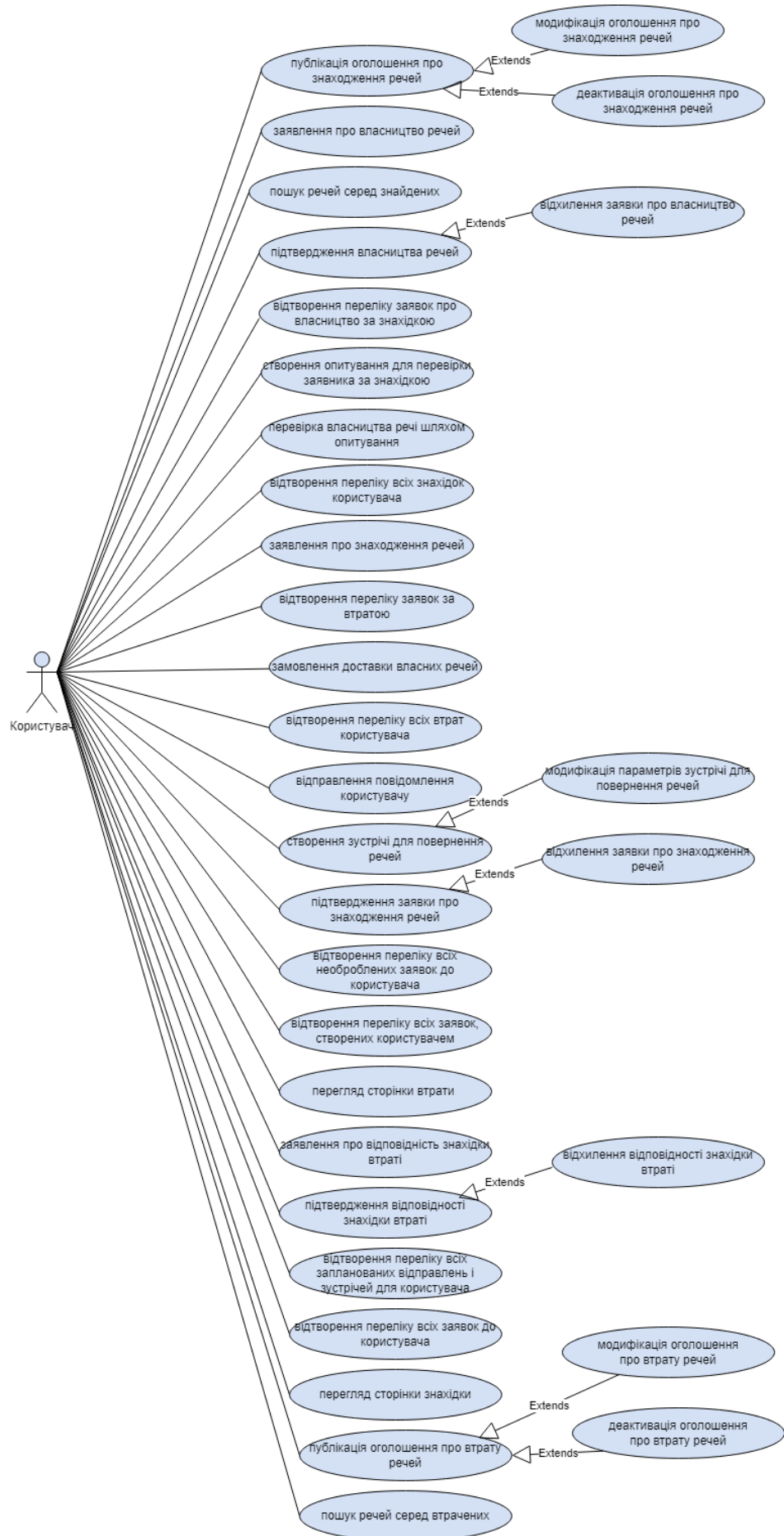


Рисунок 2.1 – Діаграма варіантів використання системи користувачами

До функцій користувача належать також:

- модифікація параметрів зустрічі для повернення речей;
- відтворення переліку всіх заявок до користувача;
- відтворення переліку всіх необроблених заявок до користувача;
- відтворення переліку всіх заявок, створених користувачем;
- відтворення переліку всіх запланованих відправлень і зустрічей для користувача;
- перегляд сторінки знахідки;
- перегляд сторінки втрати.

2.3 Проєктування бази даних

База даних (БД) створена і реалізована під керуванням MySQL [8].

Знахідка вцілому описується оголошенням, яке може включати різні знайдені речі, визначаючи таким чином цілісний випадок. Дані знахідки зберігаються в таблиці Founditemsituation, описуючи її полями:

- id – ідентифікатор знахідки;
- slug – частина Uniform Resource Locator (URL)-адреси, яка унікально ідентифікує знахідку;
- title – загальна назва, що описує суть знахідки, яка буде використовуватися як назва;
- company – підприємство, на території якого було знайдено знахідку, що представляє зв'язок з таблицею Company через цей зовнішній ключ, значення може бути пропущено, якщо знахідка, наприклад, відбулась на вулиці (тоді вказується тільки адреса в наступних полях);
- city – місто, в якому відбулась знахідка, що представляє зв'язок з таблицею City через цей зовнішній ключ;
- order – замовлення, в якому визначено параметри доставки речей знахідки власнику;

- full_address – повна адреса (без назви міста), де відбулась ця знахідка;
- address_long – довгота координати місця, де відбулась ця знахідка;
- address_latt – широта координати місця, де відбулась ця знахідка;
- description – опис загальної знахідки, випадку, що трапився, та того, що було знайдено загалом;
- accountaddress – адреса користувача, за якою зараз знаходяться речі з цієї знахідки, що представляє зв'язок з таблицею Accountaddress через цей зовнішній ключ;
- is_active – активність оголошення про знахідку на даний момент, тобто чи продовжувати пошук далі;
- check – вказує на те, чи потрібно проводити програмну перевірку користувача, який вважає цю знахідку своїми речами;
- check_stop – вказує на те, чи потрібно зупинити перевірку через наплив відхилених запитів, що може свідчити про атаку або якесь шахрайство;
- creation_date – дата і час, коли було створено оголошення про знахідку;
- found_date – дата і час, коли було знайдено речі.

Так само як і знахідка, втрата також може описуватися оголошенням. Тобто господар речей може почати їх шукати, визначаючи оголошення, ще до того, як виконав пошук серед даних про знахідки, або після того, як впевнився, що не знайшов потрібної інформації. Втрата в цілому описується оголошенням, яке може включати різні втрачені речі, визначаючи таким чином цілісний випадок. Дані втрати зберігаються в таблиці Lostitemsituation, описуючи її полями:

- id – ідентифікатор втрати;
- slug – частина Uniform Resource Locator (URL)-адреси, яка унікально ідентифікує втрату;

- title – загальна назва, що описує суть втрати, яка буде використовуватися як назва;
- company – підприємство, на території якого було втрачено речі, що представляє зв'язок з таблицею Company через цей зовнішній ключ, значення може бути пропущено, якщо втрата, наприклад, відбулась на вулиці (тоді вказується тільки адреса в наступних полях);
- order – замовлення, в якому визначено параметри доставки речей втрати власнику;
- city – місто, в якому відбулась втрата, що представляє зв'язок з таблицею City через цей зовнішній ключ;
- full_address – повна адреса (без назви міста), де відбулась ця втрата;
- address_long – довгота координати місця, де відбулась ця втрата;
- address_latt – широта координати місця, де відбулась ця втрата;
- description – опис загальної втрати, випадку, що трапився, та того, що було втрачено загалом;
- account – користувач, який створив оголошення, що представляє зв'язок з таблицею Account через цей зовнішній ключ;
- is_active – активність оголошення про втрату на даний момент, тобто чи продовжувати пошук далі, чи речі було знайдено;
- creation_date – дата і час, коли було створено оголошення про втрату;
- lost_date – дата і час, коли було втрачено речі.

І знахідка, і втрата складається з окремих знайдених або втрачених речей. Фактично і знайдена, і втрачена річ представляються тими самими характеристиками. Загалом, якщо і той, хто втратив, і той, хто знайшов річ, звернулись до програми і ввели інформацію про цю річ, то вона може бути тоді двічі представлена в БД. І як знайдена, і як втрачена. Що є логічним для пошуку. Проте відповідно і для знахідки, і для втрати можна виділити таку сутність як річ, яка буде об'єднувати просто всі характеристики речі, яка в залежності від ситуації може бути і втраченою, і загубленою. Якщо так не

робити, то в БД буде представлено дві сутності: окремо для втраченої речі, і окремо для загубленої. Але в подальшому будь-які сутності, які пов'язані з річчю, мають бути визначені і окремо для знайденої, і для втраченої речі. Це більшу частину сутностей подвоїть і змусить деякі функції в програмі окремо виділяти для одного блоку (втрачених) і для іншого блоку (знайдених) речей. Тому виділено загальну сутність річ, виносячи її в окрему таблицю *Situationitem*, описуючи її полями:

- *id* – ідентифікатор знайденої речі;
- *producer* – виробник знайденої або загубленої речі, якщо він відомий: представляє зв'язок з таблицею *Producer* через цей зовнішній ключ;
- *model* – модель знайденої або загубленої речі;
- *description* – опис знайденої або загубленої речі.

Але для кожної речі потрібно знати ситуації, коли вона є загубленою і ситуації, коли вона є знайденою, пов'язуючи їх з самою знахідкою і самою втратою.

Зв'язок речі зі знахідкою виконується через таблицю *Founditem*, описуючи її полями:

- *founditemsituation* – знахідка, до якої входить річ: представляє зв'язок з таблицею *Founditemsituation* через цей зовнішній ключ;
- *situationitem* – річ, яку було знайдено: представляє зв'язок з таблицею *Situationitem* через цей зовнішній ключ.

Річ зазвичай входить лише в одну знахідку, але знахідка може містити багато таких записів. Зважаючи на зв'язок фактично один-до-одного, його необхідність продиктована тим, що та ж сама річ може бути не знахідкою, а навпаки втратою, як було описано вище.

Зв'язок речі з втратою виконується через таблицю *Lostitem*, описуючи її полями:

- *lostitemsituation* – втрата, до якої входить річ: представляє зв'язок з таблицею *Lostitemsituation* через цей зовнішній ключ;

– situationitem – річ, яку було втрачено: представляє зв'язок з таблицею Situationitem через цей зовнішній ключ.

Для того щоб зрозуміти, що це за річ, важливою є правильна структуризація інформації, надання описових характеристик, адже це спрощує пошук, але в підсумку найкраще мати фотографію. Фотографій речі може бути декілька. Фотографії речі зберігаються в таблиці Itemphoto, описуючи її полями:

- id – ідентифікатор фотографії знайденої або загубленої речі;
- situationitem – річ, яку зображено на фотографії: представляє зв'язок з таблицею Situationitem через цей зовнішній ключ;
- photo – файл фотографії речі;
- description – опис деталей фотографії речі, якщо це необхідно;
- level – фотографія визначає саме цю річ або вона визначає всю знахідку або втрату, тобто зображує всі речі разом.

Користувач програми визначається за своїми акаунтами, а зберігаються ці дані в таблиці Account, описуючи її полями:

- id – ідентифікатор акаунта користувача;
- username – логін акаунта;
- password – пароль акаунта;
- first_name – ім'я користувача;
- last_name – прізвище користувача;
- is_active – може використовуватися акаунт користувача чи ні: необхідно для блокування доступу;
- description – опис користувача про себе;
- photo – фотографія користувача.

Оскільки знайти користувач речі може, знаходячись в різних місцях, то адрес у нього може бути декілька, в тому числі тимчасові. Для надання початкової інформації про те, де можна забрати знайдену річ, використовуються збережені в програмі адреси. Адреси користувача зберігаються в таблиці Accountaddress, описуючи її полями:

- id – ідентифікатор адреси користувача;
- account – користувач, який знаходиться за цією адресою: представляє зв'язок з таблицею Account через цей зовнішній ключ;
- city – місто, в якому знаходиться це місце (адреса): представляє зв'язок з таблицею City через цей зовнішній ключ;
- full_address – повна адреса (без назви міста) знаходження користувача;
- address_long – довгота координати місця знаходження користувача;
- address_latt – широта координати місця знаходження користувача;
- secret – показувати цю повну адреси або тільки наближену інформацію про адресу.

Загальна категорія, до якої може належати річ, є важливим кроком до структуризації, щоб загалом визначити, що саме було знайдено. Глобальні категорії речей зберігаються в таблиці Globalcategory, описуючи її полями:

- id – ідентифікатор загальної категорії речей;
- name – назва загальної категорії речей.

Речі, знайдені або втрачені, можуть належати різним категоріям, а категорія може мати різні знайдені або загублені речі в програмі, тому для реалізації зв'язку потрібна зв'язуюча таблиця SituationitemGlobalcategory з полями:

- situationitem – річ, знайдена або втрачена, яка належить загальній категорії: представляє зв'язок з таблицею Situationitem через цей зовнішній ключ;
- globalcategory – глобальна категорія, якій належить річ: представляє зв'язок з таблицею Globalcategory через цей зовнішній ключ;

Загальна категорія важлива, але її добре було б деталізувати. Тобто знаходження наплічника важливо, але якщо розуміти, що це міський наплічник або що це гірський наплічник, то це значно уточнює загальні дані пошуку. Тому введено локальну категорію речей для цього, що зберігається в таблиці Localcategory, описуючи її полями:

- id – ідентифікатор локальної категорії речей;
- globalcategory – глобальна категорія речей, до якої належить ця локальна: представляє зв'язок з таблицею Globalcategory через цей зовнішній ключ;
- name – назва локальної категорії речей.

Може бути ситуація, коли про знайдену річ відома тільки глобальна категорія, як це було визначено вище, але може бути і ситуація, коли відома локальна категорія. Для цього звичайно людина, яка знайшла річ, має на ній розумітися у відповідній мірі. Тоді також одна річ може належати різним локальним категоріям, а одна локальна категорія може містити декілька речей, тому для реалізації зв'язку потрібна зв'язуюча таблиця SituationitemLocalcategory з полями:

- situationitem – річ, знайдена або втрачена, яка належить локальній категорії: представляє зв'язок з таблицею Situationitem через цей зовнішній ключ;
- localcategory – локальна категорія, якій належить річ: представляє зв'язок з таблицею Localcategory через цей зовнішній ключ.

Знайдені речі можуть бути дуже різними. Це може бути наплічник, а може бути ручка, клавіатура чи ювелірна прикраса. Ці речі мають різні розміри, різні характеристики загалом. Кожна з них має свій власний набір характеристик, а кожна конкретна річ – значення цих характеристик. Частково ці характеристики у різних речей можуть перетинатися, а частково – ні. Тобто загалом це достатньо складна для структуризації інформація. Загальні атрибути знайдених або втрачених речей зберігаються в таблиці Attribute, описуючи її полями:

- id – ідентифікатор атрибуту, який може використовуватися як характеристика знайденої або втраченої речі;
- name – назва атрибуту.

Конкретна знайдена або втрачена річ має конкретні атрибути з цього загального набору, що міститься в таблиці Attribute, при чому для кожного

атрибута встановлюється відповідне значення. Ці дані зберігаються в таблиці `Situationitemattribute`, описуючи її полями:

- `id` – ідентифікатор атрибуту конкретної знайденої або втраченої речі;
- `situationitem` – річ, знайдена або втрачена, яка належить локальній категорії: представляє зв'язок з таблицею `Situationitem` через цей зовнішній ключ;

- `attribute` – атрибут, значення якого встановлено для цієї речі: представляє зв'язок з таблицею `Attribute` через цей зовнішній ключ;

- `attribute_value` – значення атрибуту;

- `secret` – демонструвати значення цього атрибуту відкрито чи ні.

Заявка щодо втрати формується тоді, коли користувач виявив втрату і вважає, що він знайшов саме ці речі. Дані заявки щодо втрати зберігаються в таблиці `LostApplication`, описуючи її полями:

- `id` – ідентифікатор заявки щодо втрати;

- `account` – користувач, який оформив заявку щодо втрати: представляє зв'язок з таблицею `Account` через цей зовнішній ключ;

- `lostitemsituation` – втрата, якої стосується заявка: представляє зв'язок з таблицею `Lostitemsituation` через цей зовнішній ключ;

- `description` – інформація від користувача, який вважає, що він знайшов ці втрачені речі;

- `is_active` – активна заявка чи ні: користувач, який шукав втрачені речі, міг скасувати заявку або визначити, що дійсно це його речі;

- `creation_date` – дата створення заявки щодо втрати.

Заявка щодо знахідки формується тоді, коли користувач виявив знахідку і вважає її своєю річчю. Дані заявки щодо знахідки зберігаються в таблиці `LostApplication`, описуючи її полями:

- `id` – ідентифікатор заявки щодо знахідки;

- `account` – користувач, який оформив заявку щодо знахідки: представляє зв'язок з таблицею `Account` через цей зовнішній ключ;

- founditemsituation – знахідка, якої стосується заявка: представляє зв'язок з таблицею Founditemsituation через цей зовнішній ключ;
- description – інформація від користувача, який вважає себе власником речей;
- is_active – активна заявка чи ні: користувач, який знайшов речі, міг скасувати заявку або визначити, що дійсно цей користувач є власником речей;
- creation_date – дата створення заявки щодо знахідки.

Бувають ситуації, коли користувач знайшов оголошення про знахідку, при цьому маючи власне оголошення про втрату, або навпаки, тоді потрібно вказати, що ця знахідка відповідає втраті. Дані співпадіння зберігаються в таблиці Coincedence, описуючи її полями:

- id – ідентифікатор співпадіння між знахідкою і втратою;
- founditemsituation – знахідка, яка відповідає оголошенню про втрату: представляє зв'язок з таблицею Founditemsituation через цей зовнішній ключ;
- lostitemsituation – втрата, яка відповідає оголошенню про знахідку: представляє зв'язок з таблицею Lostitemsituation через цей зовнішній ключ;
- creator – співпадіння було визначено автором оголошення про втрату або автором оголошення про знахідку;
- description – інформація від користувача, який знайшов співпадіння або вважає, що це співпадіння;
- is_active – активне співпадіння чи ні: користувач міг скасувати заявку або визначити, що дійсно відбулось співпадіння;
- creation_date – дата створення заявки щодо співпадіння.

Користувач може не створювати в програмі оголошення, відповідно не мати фотографій знайденої чи втраченої речі, при цьому одразу знайти оголошення. Тоді подаючи заявку, він може завантажити фотографії до заявки. Оскільки заявка може бути і щодо страти, і щодо знахідки, то створено окремі таблиці для цього. Зокрема таблицю LostApplicationphoto, описуючи її полями:

– id – ідентифікатор фотографії, наданої в підтвердження заявки щодо втрати;

– lostapplication – заявка щодо втрати, для якої користувач бажає продемонструвати наявну в нього річ;

– photo – файл фотографії речі;

– description – опис деталей фотографії речі, якщо це необхідно.

Фотографії для заявок щодо знахідок винесені в таблицю FoundApplicationphoto, описуючи її полями:

– id – ідентифікатор фотографії, наданої в підтвердження заявки щодо знахідки;

– foundapplication – заявка щодо знахідки, для якої користувач бажає продемонструвати загублену ним річ;

– photo – файл фотографії речі;

– description – опис деталей фотографії речі, якщо це необхідно.

За заявкою може бути визначено замовлення для правдивого власника і сформовано деталі подальшої взаємодії (зокрема доставки), що мають бути винесені в таблицю Order, описуючи її полями:

– id – ідентифікатор замовлення доставки речі до власника;

– account – користувач, який володіє річчю, яку було знайдено іншою стороною, або користувач, у якого знаходиться річ, яку загубив власник оголошення про втрату: представляє зв'язок з таблицею Account через цей зовнішній ключ;

– creation_date – дата і час створення замовлення доставки;

– is_approved – підтвердження умов доставки, які визначені власником речі, зі сторони людини, яка знайшла речі;

– sent_date – дата і час відправлення речей власнику;

– delivered_date – дата і час доставки речей власнику;

– deliverystation – пункт доставки, куди має бути доставлено речі;

– phone – номер телефону для доставки.

Підприємства, на території яких було знайдено або загублено речі, вносяться в таблицю Companys, описуючи її полями:

- id – ідентифікатор підприємства (ресторану, готелю, вокзалу, магазину);
- companysphere – сфера, в якій діє підприємство;
- name – назва підприємства;
- city – місто, де знаходиться підприємство: представляє зв'язок з таблицею City через цей зовнішній ключ.
- full_address – повна адреса (без міста), за якою знаходиться підприємство;
- address_long – довгота координати місця, де знаходиться підприємство;
- address_latt – широта координати місця, де знаходиться підприємство;
- description – опис підприємства;
- photo – фотографія підприємства.

Сфери, в яких діють підприємства, зберігаються в таблиці Companysphere, описуючи її полями:

- id – ідентифікатор сфери, в якій діє підприємство;
- name – назва підприємства.

Після визначення власництва може не обов'язково відбуватися доставка. Зокрема якщо це одне й те саме місто, а обидві сторони готові зустрітися, то за замовленням може організовуватися зустріч. Зустрічі зберігаються в таблиці Appointment, описуючи її полями:

- order – замовлення, якого стосується зустріч;
- full_address – повна адреса, за якою має відбутися зустріч;
- address_long – довгота координати місця, де має відбутися зустріч;
- address_latt – широта координати місця, де має відбутися зустріч;
- description – коментар щодо зустрічі.

Для відсіювання хибних заявок автоматично або з перевіркою користувачем, який створив оголошення, за знахідкою можуть створюватися опитування. Запитання опитувань зберігаються в таблиці `Founditemsituationquestion`, описуючи її полями:

- `id` – ідентифікатор запитання опитування;
- `founditemsituation` – знахідка, якої стосується запитання опитування: представляє зв'язок з таблицею `Founditemsituation` через цей зовнішній ключ;
- `question` – запитання, яке потрібно поставити за знахідкою;
- `answer` – відповідь, яку потрібно отримати за запитанням (якщо перевірка відбувається автоматично);
- `check_answer` – необхідність перевірити відповідь на запитання автоматично: в такому випадку відповідь має точно співпадати з вказаною.

Будь-який претендент на власництво за знахідкою, за якою повинно бути проведено опитування, має його пройти, а результати фіксуються в таблиці `Founditemsituationanswer`, описуючи її полями:

- `id` – ідентифікатор опитування заявника за знахідкою;
- `foundapplication` – заявка, за якою було проведено опитування: представляє зв'язок з таблицею `Foundapplication` через цей зовнішній ключ;
- `creation_date` – дата і час проведення опитування;
- `result` – результат опитування (успішні відповіді на всі запитання при автоматичному проведенні, успішне визначення вручну або виявлення недостовірності заявки).

Самі відповіді на запитання під час опитування також фіксуються, для чого виділена таблиця `Questionanswer`, описуючи її полями:

- `id` – ідентифікатор відповіді на запитання з опитування;
- `founditemsituationanswer` – опитування заявника за знахідкою, якого стосується відповідь;
- `founditemsituationquestion` – запитання опитування, на яке було надано цю відповідь: представляє зв'язок з таблицею `Founditemsituationquestion` через цей зовнішній ключ;

– answer – відповідь, отримана від заявника.

Міста, які використовувались вже вище при визначенні адрес, зберігаються окремо в таблиці City, описуючи її полями:

– id – ідентифікатор міста, з яким працює програма;

– name – назва міста, з яким працює програма;

– country – країна, в якій знаходиться місто: представляє зв'язок з таблицею Country через цей зовнішній ключ.

Відповідно країни зберігаються в таблиці Country, описуючи її полями:

– id – ідентифікатор країни, з якою працює програма;

– name – назва країни, з якою працює програма.

Повідомлення між користувачами щодо обговорення знайдених речей зберігаються в таблиці Accountmessage, описуючи її полями:

– id – ідентифікатор повідомлення між користувачами;

– account_from – користувач, який відправив повідомлення: представляє зв'язок з таблицею Account через цей зовнішній ключ;

– account_to – користувач, який отримав повідомлення: представляє зв'язок з таблицею Account через цей зовнішній ключ;

– message_text – текст повідомлення;

– message_date – дата і час відправлення цього повідомлення.

Знахідка або втрата не завжди є статичною подією. Можна загубити річ в потязі. Тоді не дуже суттєвим є те, в якому місті річ була загублена. Важливим є те, що на всьому подальшому маршруті потягу річ могла бути знайдена. Це стосується, наприклад, і міжміських автобусів. Збереження даних про маршрут, в русі яким було знайдено речі, відбувається в таблиці Founditemroute, описуючи її полями:

– id – ідентифікатор пункту маршруту;

– founditemsituation – знахідка, до якої відноситься цей пункт маршруту: представляє зв'язок з таблицею Founditemsituation через цей зовнішній ключ;

– city – місто, яке є пунктом маршруту: представляє зв'язок з таблицею City через цей зовнішній ключ.

Збереження даних про маршрут, в русі яким було загублено речі, відбувається в таблиці Lostroute, описуючи її полями:

– id – ідентифікатор пункту маршруту;

– lostitemsituation – втрата, до якої відноситься цей пункт маршруту: представляє зв'язок з таблицею Lostitemsituation через цей зовнішній ключ;

– city – місто, яке є пунктом маршруту: представляє зв'язок з таблицею City через цей зовнішній ключ.

Виробники речей зберігаються в таблиці Producer, описуючи її полями:

– id – ідентифікатор виробника речей;

– name – назва виробника.

Сервіси доставки зберігаються в таблиці Delivery, описуючи її полями:

– id – ідентифікатор сервісу доставки;

– name – назва сервісу доставки.

Пункти доставки відповідного сервісу зберігаються в таблиці Deliverystation, описуючи її полями:

– id – ідентифікатор пункту доставки;

– delivery – сервіс доставки, до якого відноситься пункт: представляє зв'язок з таблицею Delivery через цей зовнішній ключ;

– number – номер пункту доставки;

– specialization – особливості спеціалізації пункту доставки, якщо вони є;

– city – місто, в якому знаходиться пункт доставки: представляє зв'язок з таблицею City через цей зовнішній ключ;

– address – повна адреса (без міста), за якою знаходиться пункт доставки;

- address_long – довгота координати місця, де розташований пункт доставки;
- address_latt – широта координати місця, де розташований пункт доставки (рис. 2.2).

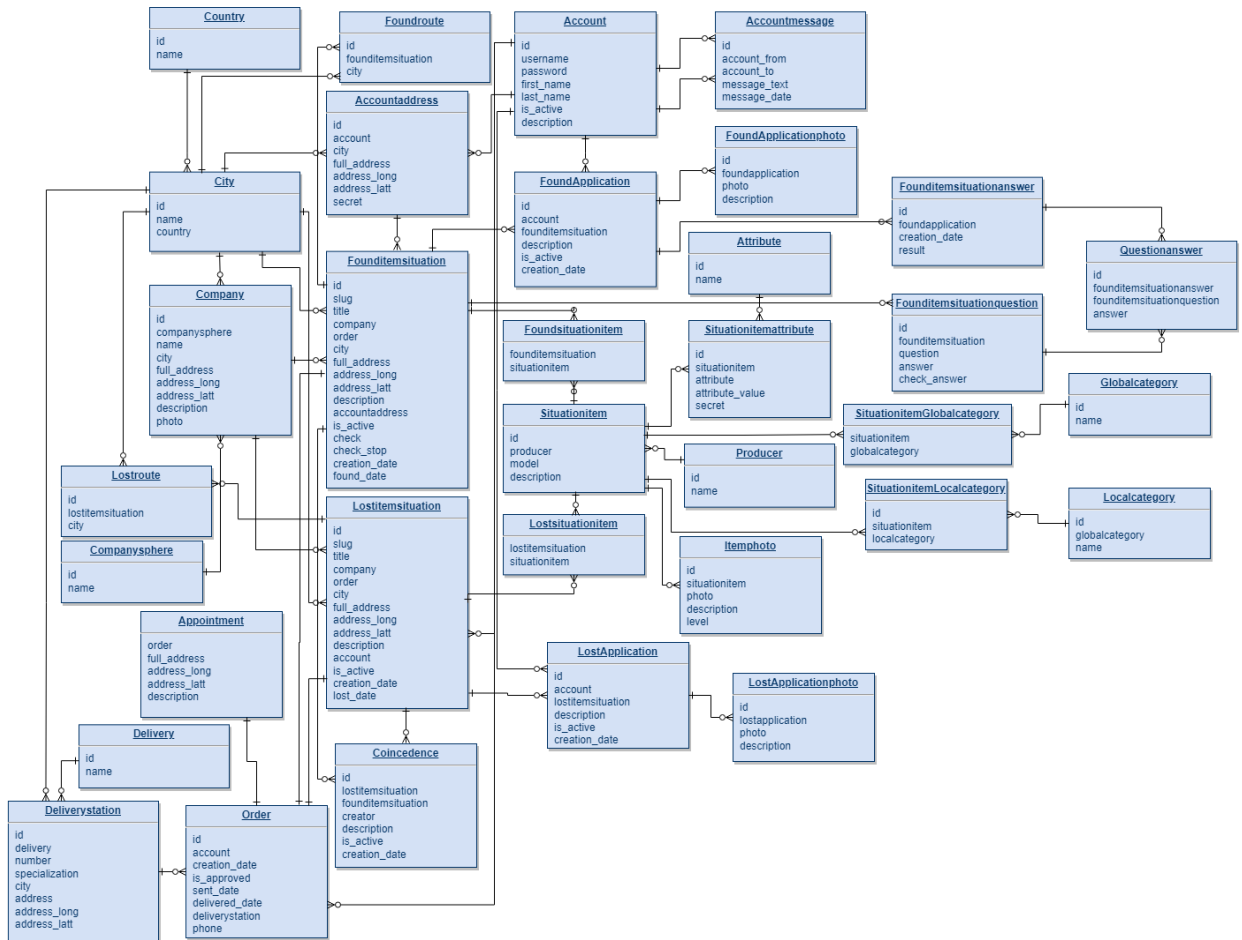


Рисунок 2.2 – Схема БД

2.4 Висновки за розділом 2

У розділі повністю досягнуто виконання завдання проєктування програмного забезпечення пошуку господарів знайдених речей, включаючи вибір мови програмування Python, моделювання вимог до програми на основі діаграми варіантів використання програми, визначення структури роботи з даними на основі БД.

3 РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1 Порядок виконання пошуку господарів знайдених речей

Для програмного забезпечення пошуку господарів знайдених речей звичайно основною є процедура самого пошуку. Вона передбачає виставлення ряду параметрів, а потім гнучке виконання пошуку серед всіх знахідок за визначеними параметрами з виділенням в підсумку тієї підмножини знахідок, які відповідають параметрам пошуку (рис. 3.1).

Спочатку реалізується визначення параметрів підприємства, де було загублено речі, щоб визначити за ним ті знахідки, які були виконані саме на території підприємства (ресторану, готелю і т.д.). Якщо підприємство не визначено, то відбувається визначення іншого варіанту пошуку за локацією. Цим варіантом є безпосередньо вказання міста, де загублено речі. Якщо місто вказано напряду, то відбувається відбір знахідок на території міста. Якщо місто не вказано, то реалізується перехід до іншого варіанта визначення локації. Для цього визначається маршрут руху транспорту, в якому було загублено речі. Якщо певні міста було вказано, то відбираються ті оголошення, які стосуються цього маршруту руху. Якщо і цей варіант не використано, то відбувається визначення точки і радіусу пошуку. За нею відбираються всі знахідки, які були виконані в заданому радіусу відносно вказаної точки. Після цього реалізується перехід, як і після будь-якого виконаного раніше варіанта відбору, до визначення періоду дат втрати. Якщо вони визначені, то відбувається відбір знахідок за періодом знайдення, що має відповідати встановленому періоду дат втрати. І після успішного визначення, і після відсутності визначення виконується перехід до визначення речей зі знахідки. Якщо встановлено конкретні речі, то відбувається пошук всіх знахідок, які містять всі вказані речі у своєму складі. Якщо речі не визначено, то визначаються виробники речей, а за ними відбираються всі знахідки з речима цих виробників (рис. 3.1).

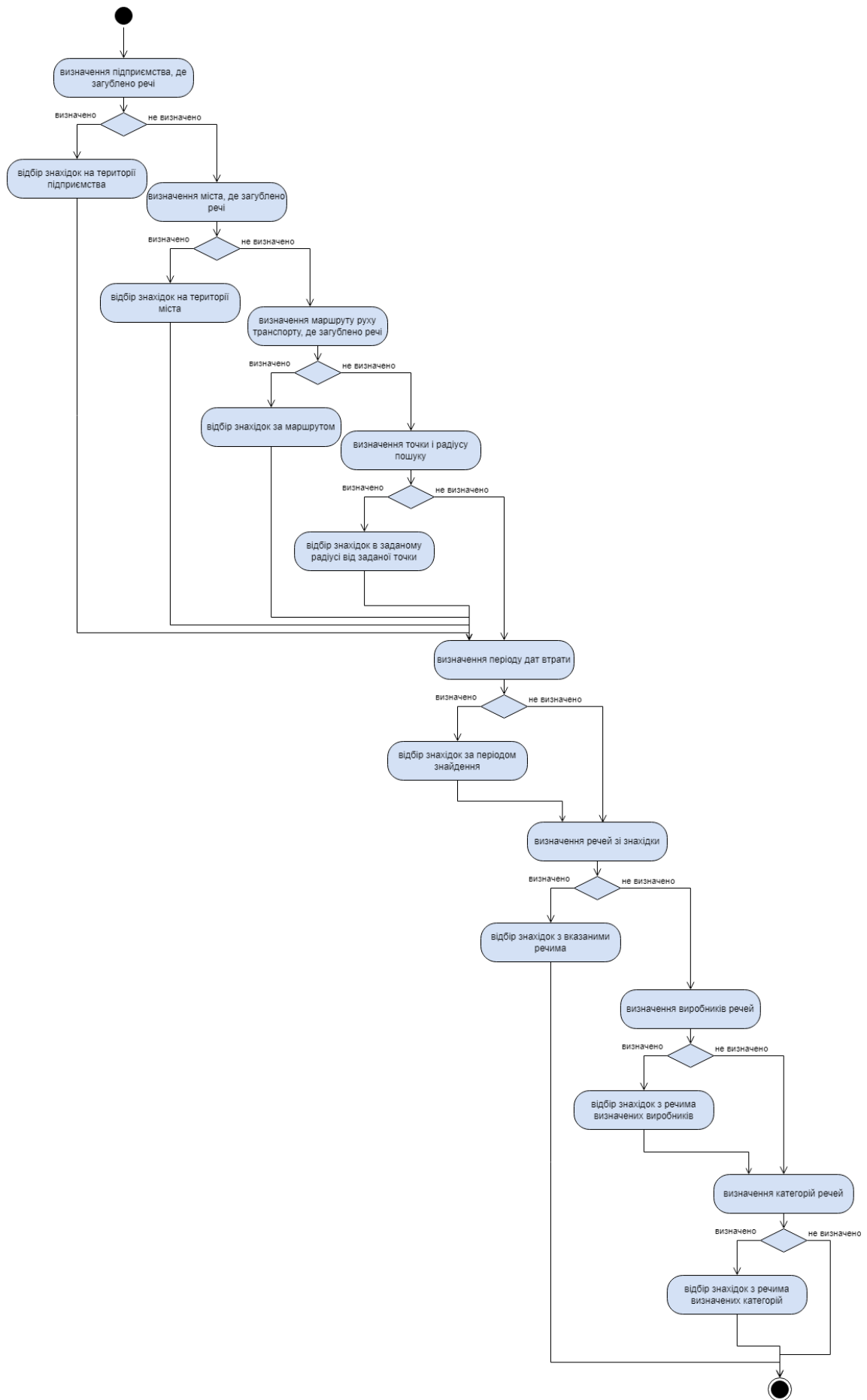


Рисунок 3.1 – Діаграма діяльності пошуку за знахідками

Далі виконується визначення категорій речей. Якщо вони встановлені, то відбираються всі оголошення, які містять речі вказаних категорій. Після цього відбору виконання пошуку завершується.

3.2 Програмна реалізація вебсистеми

Для перегляду сторінки знахідки використовується функція `founditemsituation` (`request`, `foundsituation_slug` = `None`), що окрім базового параметра має параметр `foundsituation_slug`, який визначає частину URL-адреси, яка унікально ідентифікує знахідку, оголошення про яку представляється в результаті. У результаті роботи ця функція представлення наповнює шаблон вебсторінки `founditemsituation.html`, створений з підтримкою безкоштовного шаблону [9], словником, що містить ключі:

- `founditemsituation` – об'єкт знахідки з ідентифікатором `foundsituation_slug` з повними даними не тільки про сам об'єкт, але і підприємство, місто, місто підприємства, адресу акаунта, місто адреси акаунта, акаунт з адреси акаунта, сферу роботи підприємства, маршрут, пов'язаний зі знахідкою;
- `situationitems` – об'єкти знайдених речей моделі `Situationitem`, що відповідають цій знахідці, включаючи також фотографії речей, глобальні категорії, локальні категорії, атрибути зі значеннями та іменами, дані виробника;
- `questions` – запитання, за якими проводиться опитування: цей ключ присутній тільки в тому випадку, якщо авторизований користувач є автором оголошення про знахідку.

Для перегляду сторінки втрати використовується функція `lostitemsituation` (`request`, `lostitemsituation_slug` = `None`), що окрім базового параметра має параметр `lostitemsituation_slug`, який визначає частину URL-адреси, яка унікально ідентифікує втрату, оголошення про яку представляється в результаті. У результаті роботи ця функція представлення

наповнює шаблон вебсторінки `lostitemsituation.html` словником, що містить ключі:

- `lostitemsituation` – об'єкт втрати з ідентифікатором `lostitemsituation_slug` з повними даними не тільки про сам об'єкт, але і підприємство, місто, місто підприємства, акаунт, сферу роботи підприємства, маршрут, пов'язаний з втратою;

- `situationitems` – об'єкти втрачених речей моделі `Situationitem`, що відповідають цій втраті, включаючи також фотографії речей, глобальні категорії, локальні категорії, атрибути зі значеннями та іменами, дані виробника.

Для пошуку речей серед знайдених використовується функція `search_found_items(request)`. Робота функції організована за описаною в підрозділі 3.1 логікою.

Фактично спочатку витягається значення параметра GET-запиту `company`, якщо воно встановлено, за ним витягається відповідний об'єкт підприємства, а за цим підприємством витягуються знахідки, оголошення яких є активними (`is_active`).

За кожним оголошенням витягуються речі з цієї знахідки, фотографії речей, глобальні категорії, локальні категорії, атрибути зі значеннями та іменами, дані маршруту, підприємства, міста підприємства, сфери діяльності підприємства.

Далі витягається значення параметра GET-запиту `city`, за ним витягається відповідний об'єкт міста, а за цим містом відфільтровуються всі вже відібрані знахідки. З параметра `cities` відбираються ідентифікатори кожного міста, а за ними створюється перелік об'єктів відповідних міст. Усі відібрані знахідки відфільтровуються за входженням в маршрут одного з заданих у переліку міст.

Зі значень параметрів `found_from` і `found_to` визначається інтервал, коли могли бути знайдені речі. Відповідно відфільтровуються всі знахідки, які були здійснені в цьому інтервалі часу.

Далі з параметра `producers` виділяються виробники, речі яких мають бути в знахідці. За параметром витягається відповідні об'єкти виробників, а за цим виробником відфільтровуються всі вже відібрані знахідки. З параметра `producers` відбираються ідентифікатори кожного виробника, а за ними створюється перелік об'єктів відповідних виробників.

З параметра `local` відбираються ідентифікатори кожної локальної категорії, а за ними створюється перелік об'єктів відповідних локальних категорій. Усі відібрані знахідки відфільтровуються за тим, що до них входять речі, які належать одній з цих категорій.

З параметра `global` відбираються ідентифікатори кожної глобальної категорії, а за ними створюється перелік об'єктів відповідних глобальних категорій. Усі відібрані знахідки відфільтровуються за тим, що до них входять речі, які належать одній з цих категорій.

У параметрах `address_long`, `address_latt`, `distance` встановлюються довгота і широта місця, де як вважає користувач, були загублені речі та можливий радіус відстані від цієї точки, де можна було б знайти речі. За цими параметрами, якщо вони були встановлені, перевіряється за кожною знахідкою, на скільки далеко вона була здійснена від цієї точки. Якщо відстань між цими точками не потрапляє в задану відстань, то знахідка вилучається з результатів.

На останньому етапі перевіряються всі конкретно задані речі з параметра `items`. Даний параметр розшифровується з виділенням повної інформації про речі, включаючи наступні позиції, відділені між собою комами:

- символи «l-» на початку рядка з розділеними крапками з комами ідентифікаторами локальних категорій;
- символи «g-» на початку рядка з розділеними крапками з комами ідентифікаторами глобальних категорій;
- символи «m-» на початку рядка з подальшою назвою моделі речі;

- символи «р-» на початку рядка з подальшим ідентифікатором виробника;
- пари з ідентифікатора атрибута та значення атрибута, відділених дефісом.

Після розшифрування описаним чином даного параметра виконується пошук знахідок, які містять відповідні речі. Для цього перевіряються всі відфільтровані за минулими кроками знахідки. Для кожної знахідки перевіряється кількість речей зі знахідки, які відповідають заданим наборам параметрів речей (виробник, модель, атрибути зі значеннями, локальні та глобальні категорії). Якщо вона не менше ніж задана кількість наборів параметрів, то така знахідка залишається в результатах.

Після цього шаблон сторінки `search_found_items.html` заповнюється словником, що містить ключі:

- `situations` – основні результати пошуку, що включають відібрані знахідки;
- `globalcategories` – перелік всіх відомих глобальних категорій, які потрібні для визначення параметрів пошуку;
- `localcategories` – перелік всіх відомих локальних категорій, які потрібні для визначення параметрів пошуку;
- `cities` – перелік всіх відомих міст, які потрібні для визначення параметрів пошуку;
- `attributes` – перелік всіх відомих атрибутів, які потрібні для визначення параметрів пошуку;
- `producers` – перелік всіх відомих виробників, які потрібні для визначення параметрів пошуку.

Для пошуку речей серед загублених використовується функція `search_lost_items (request)`. Спочатку витягається значення параметра `GET-запиту company`, якщо воно встановлено, за ним витягається відповідний об'єкт підприємства, а за цим підприємством витягуються знахідки.

За кожним оголошенням витягуються речі з цієї втрати, фотографії речей, глобальні категорії, локальні категорії, атрибути зі значеннями та іменами, дані маршруту, підприємства, міста підприємства, сфери діяльності підприємства.

Далі витягається значення параметра GET-запиту city, за ним витягається відповідний об'єкт міста, а за цим містом відфільтровуються всі вже відібрані втрати. З параметра cities відбираються ідентифікатори кожного міста, а за ними створюється перелік об'єктів відповідних міст. Усі відібрані втрати відфільтровуються за входженням в маршрут одного з заданих у переліку міст.

За значенням параметра lost_from визначається найбільш рання дата, коли могли бути втрачені речі. Звичайно, що вона не може бути ранішою за ту дату, коли речі були знайдені користувачем, який виконує пошук. Відповідно відфільтровуються всі втрати за цією датою.

Далі з параметра producers виділяються виробники, речі яких мають бути у оголошенні про втрату. За параметром витягається відповідні об'єкти виробників, а за цим виробником відфільтровуються всі вже відібрані втрати. З параметра producers відбираються ідентифікатори кожного виробника, а за ними створюється перелік об'єктів відповідних виробників.

З параметра local відбираються ідентифікатори кожної локальної категорії, а за ними створюється перелік об'єктів відповідних локальних категорій. Усі відібрані втрати відфільтровуються за тим, що до них входять речі, які належать одній з цих категорій.

З параметра global відбираються ідентифікатори кожної глобальної категорії, а за ними створюється перелік об'єктів відповідних глобальних категорій. Усі відібрані втрати відфільтровуються за тим, що до них входять речі, які належать одній з цих категорій.

У параметрах address_long, address_latt, distance встановлюються довгота і широта місця, де як вважає користувач, були знайдені речі та можливий радіус відстані від цієї точки, де можна було б втратити речі. За

цими параметрами, якщо вони були встановлені, перевіряється за кожною втратою, на скільки далеко вона була здійснена від цієї точки. Якщо відстань між цими точками не потрапляє в задану відстань, то оголошення про втрату вилучається з результатів.

На останньому етапі перевіряються всі конкретно задані речі з параметра `items`. Даний параметр розшифровується з виділенням повної інформації про речі аналогічно тому, як це було описано в пошуку знахідок.

Після розшифрування даного параметра виконується пошук втрат, які містять відповідні речі. Для цього перевіряються всі відфільтровані за минулими кроками втрати. Для кожної втрати перевіряється кількість речей зі втрати, які відповідають заданим наборам параметрів речей (виробник, модель, атрибути зі значеннями, локальні та глобальні категорії). Якщо вона не менше ніж задана кількість наборів параметрів, то така втрата залишається в результатах.

Після цього шаблон сторінки `search_lost_items.html` заповнюється словником, що містить ключі:

- `situations` – основні результати пошуку, що включають відібрані втрати;
- `globalcategories` – перелік всіх відомих глобальних категорій, які потрібні для визначення параметрів пошуку;
- `localcategories` – перелік всіх відомих локальних категорій, які потрібні для визначення параметрів пошуку;
- `cities` – перелік всіх відомих міст, які потрібні для визначення параметрів пошуку;
- `attributes` – перелік всіх відомих атрибутів, які потрібні для визначення параметрів пошуку;
- `producers` – перелік всіх відомих виробників, які потрібні для визначення параметрів пошуку.

Для відтворення переліку всіх знахідок користувача використовується функція `founditemsituations (request, user_account = None)`, параметр якої

`user_account` визначає логін користувача, а якщо він не заданий, то дані користувача визначаються через дані його авторизації в параметрі `request`. За даними користувача відбираються всі оголошення про знахідки, автором яких є цей користувач, розташовуючи спочатку останні. За знахідкою також перевіряється наявність створеного замовлення, що говорить про завершення пошуку.

Для відтворення переліку всіх втрат користувача використовується функція `lostitemsituations` (`request`, `user_account = None`), параметр якої `user_account` визначає логін користувача, а якщо він не заданий, то дані користувача визначаються через дані його авторизації в параметрі `request`. За даними користувача відбираються всі оголошення про втрати, автором яких є цей користувач, розташовуючи спочатку останні. За втратою також перевіряється наявність створеного замовлення, що говорить про завершення пошуку.

Для модифікації оголошення про знаходження речей використовується функція `edit_founditemsituation` (`request`, `situation_slug`), яка на основі форми `FounditemsituationForm` виконує керування процесом модифікації даних знахідки. При цьому зміна відбувається саме параметрів основного оголошення, без модифікації даних самих речей, які були знайдені. В результаті в шаблон сторінки `edit_founditemsituation.html` повертається словник з ключами: `form`, що містить саму форму з даними оголошення, яка у випадку звернення за методом GET заповнюється даними відповідного оголошення, а у випадку звернення за методом POST зберігає внесені дані до БД, а також ключами `situation`, що містить дані самої знахідки, та `items`, що містить дані речей з оголошення. Речі на сторінці відображаються, але модифікація їх відбувається окремо.

Для модифікації оголошення про знаходження речей використовується функція `edit_founditemsituation` (`request`, `situation_slug`), яка на основі форми `FounditemsituationForm` виконує керування процесом модифікації даних знахідки. При цьому зміна відбувається саме параметрів

основного оголошення, без модифікації даних самих речей, які були знайдені. В результаті в шаблон сторінки `edit_founditemsituation.html` повертається словник з ключами: `form`, що містить саму форму з даними оголошення, яка у випадку звернення за методом GET заповнюється даними відповідного оголошення, а у випадку звернення за методом POST зберігає внесені дані до БД, а також ключами `situation`, що містить дані самої знахідки, та `items`, що містить дані речей з оголошення. Речі на сторінці відображаються, але модифікація їх відбувається окремо.

Для модифікації оголошення про втрату речей використовується функція `edit_lostitemsituation(request, situation_slug)`, яка на основі форми `LostitemsituationForm` виконує керування процесом модифікації даних втрати. При цьому зміна відбувається саме параметрів основного оголошення, без модифікації даних самих речей, які були знайдені. В результаті в шаблон сторінки `edit_lostitemsituation.html` повертається словник з ключами: `form`, що містить саму форму з даними оголошення, яка у випадку звернення за методом GET заповнюється даними відповідного оголошення, а у випадку звернення за методом POST зберігає внесені дані до БД, а також ключами `situation`, що містить дані самої втрати, та `items`, що містить дані речей з оголошення. Речі на сторінці відображаються, але модифікація їх відбувається окремо.

Відповідно для двох останніх представлень потрібно додатково виконувати модифікацію даних про річ з оголошення, для чого побудована функція представлення `edit_situationitem(request, item_id)` з параметром `item_id`, який визначає ідентифікатор речі, модифікація даних якої виконується. При реалізації функції виконується перевірка прав. Для цього визначається, чи було задано у відповідному об'єкті речі атрибут `lostitemsituation`. Якщо так, то дані вибираються з відповідного оголошення про втрати, перевіряючи, що автором цього оголошення є користувач, який авторизувався. Якщо ця річ належить до оголошення про знахідку, то перевіряється, що вказана у оголошенні адреса належить користувачу, який

авторизувався. Тільки після перевірки відбувається доступ до форми `SituationitemForm`. Якщо доступ до представлення відбувається методом `GET`, то форма заповнюється даними відповідного оголошення, а у випадку звернення за методом `POST` форма зберігає внесені дані до БД.

Функція `publish_situationitem (request, lost = 0, situation_slug)` реалізує додавання фактично додаткової речі до оголошення. Оскільки це може бути або оголошення про знахідку, або про втрату, то параметр `lost` визначає варіант оголошення, до якого додається річ. Нульове значення говорить про оголошення про втрату, відповідно ідентифікатор з URL-адреси цього оголошення визначається параметром `situation_slug`. Інші значення вказують на оголошення про знахідку. За цими даними витягається об'єкт відповідного оголошення і вноситься в відповідний атрибут об'єкта речі з оголошення. Робота цього представлення підтримується також формою `SituationitemForm`, адже модифікація і додавання фактично мають ті самі параметри.

Для деактивації оголошення про знаходження речей використовується функція `cancel_lostitemsituation (request, situation_slug)`, яка для оголошення про знахідку з ідентифікатором URL-адреси `situation_slug` змінює значення атрибута `is_active` на `False`, після чого знахідка вже не з'являється в результатах пошуку.

Для деактивації оголошення про втрату речей використовується функція `cancel_founditemsituation (request, situation_slug)`, яка для оголошення про втрату з ідентифікатором URL-адреси `situation_slug` змінює значення атрибута `is_active` на `False`, після чого втрата вже не з'являється в результатах пошуку.

Публікація оголошення про знаходження речей реалізується функцією `publish_founditemsituation`, в основі якої лежать одразу дві форми. Це форма створення оголошення про знахідку `FounditemsituationForm` та форма додавання речі до оголошення `SituationitemForm`. При збереженні за звернення за методом `POST` для речі виконується її зв'язування з новоствореним оголошенням. Для того щоб додати інші речі, потрібно в

готовому оголошенні викликати представлення додавання речі до оголошення, описану вище.

Публікація оголошення про втрату речей реалізується функцією `publish_lostitemsituation`, в основі якої лежать одразу дві форми. Це форма створення оголошення про втрату `Lostitemsituation` та форма додавання речі до оголошення `SituationitemForm`. При збереженні за звернення за методом `POST` для речі виконується її зв'язування з новоствореним оголошенням.

Для відтворення переліку заявок про власництво за знахідкою використовується функція `founditemsituation_applications` (`request`, `founditemsituation_slug = None`), що має параметр `founditemsituation_slug`, який визначає частину URL-адреси, яка унікально ідентифікує знахідку. У результаті роботи ця функція представлення наповнює шаблон вебсторінки `founditemsituation_applications.html` переліком тих заявок, які стосуються цієї знахідки як об'єкт моделі `FoundApplication` або `Coincedence`, починаючи з останньої поданої. Якщо за заявкою є створене замовлення, тобто замовлення стосується того ж користувача, яким була створена заявка, то додається в полі за цією заявкою замовлення.

Для відтворення переліку всіх заявок до користувача використовується функція `account_applications` (`request`), що не має параметрів, а формує перелік заявок з трьох моделей (знахідок, втрат і співпадінь), які стосуються знахідок і втрат, автором оголошень про які є авторизований користувач. За заявками також перевіряється наявність створеного замовлення, що говорить про підтвердження заявки. Дані замовлення повертаються у відповідному полі такої заявки.

Для відтворення переліку всіх необроблених заявок до користувача використовується функція `undone_account_applications` (`request`), що не має параметрів, а формує перелік заявок з трьох моделей (знахідок, втрат і співпадінь), які стосуються знахідок і втрат, автором оголошень про які є авторизований користувач і які при цьому є активними.

Для відтворення переліку заявок за втратою використовується функція `lostitemsituation_applications` (`request`, `lostitemsituation_slug = None`), що має параметр `lostitemsituation_slug`, який визначає частину URL-адреси, яка унікально ідентифікує втрату. У результаті роботи ця функція представлення наповнює шаблон вебсторінки `lostitemsituation_applications.html` переліком тих заявок, які стосуються цієї втрати та співпадінь за цією втратою, починаючи з останньої поданої. За заявками також перевіряється наявність створеного замовлення, що говорить про підтвердження заявки. Дані замовлення повертаються у відповідному полі такої заявки.

Для відтворення переліку всіх заявок, створених користувачем, використовується функція `applications` (`request`), що не має параметрів, а формує перелік заявок з трьох моделей (знахідок, втрат і співпадінь), які створені авторизованим користувачем. За заявками також перевіряється наявність створеного замовлення, що говорить про підтвердження заявки. Дані замовлення повертаються у відповідному полі такої заявки.

Заявлення про власництво речей реалізується функцією `publish_foundapplication` (`request`, `founditemsituation_slug`), що на основі форми `FoundApplicationForm` створює відповідну заявку, зв'язуючи її з оголошенням про знахідку з ідентифікатором URL `founditemsituation_slug`. Якщо в полі `check_stop` вказано хибне значення, то запускається перевірка власництва речі, тобто відкривається сторінка з опитуванням.

Заявлення про знаходження речей реалізується функцією `publish_lostapplication` (`request`, `lostitemsituation_slug`), що на основі форми `LostApplicationForm` створює відповідну заявку, зв'язуючи її з оголошенням про втрату з ідентифікатором URL `lostitemsituation_slug`.

Для відхилення заявки про власництво речей використовується функція `cancel_foundapplication` (`request`, `application`), яка для заявки з ідентифікатором `application` змінює значення атрибута `is_active` на `False`. При відхиленні перевіряється кількість заявок користувача як про власництво, так

і про знаходження речей, які не мають подальшого замовлення за заявкою. Якщо виявляється, що кількість заявок більше 10 і відсоток відхилених заявок більше 70 %, то доступ користувача блокується, тобто відповідний акаунт стає неактивним, зберігаючи результати в БД.

Для підтвердження власництва речей використовується функція `approve_foundapplication (request, application)`, яка для заявки з ідентифікатором `application` змінює значення атрибута `is_active` на `False`, але при цьому на відміну від відхилення заявки одразу створює в БД об'єкт моделі `Order`, за яким реалізується доставка.

Для відхилення заявки про знаходження речей використовується функція `cancel_lostapplication (request, application)`, яка для заявки з ідентифікатором `application` змінює значення атрибута `is_active` на `False`. Аналогічна перевірка виконується в даному випадку, як було вказано для відхилення заявки про власництво речей.

Для підтвердження заявки про знаходження речей використовується функція `approve_lostapplication (request, application)`, яка для заявки з ідентифікатором `application` змінює значення атрибута `is_active` на `False`, але при цьому на відміну від відхилення заявки одразу створює в БД об'єкт моделі `Order`, за яким реалізується доставка.

Відправлення повідомлення користувачу реалізується функцією `publish_accountmessage (request, account)`, що на основі форми `AccountmessageForm` створює відповідне повідомлення, адресуючи його користувачу `account`, але окрім того вона створює перелік зі всіх користувачів, яким відправляв або які відправляли авторизованому користувачу повідомлення. Перелік містить словник на кожній позиції. Словник включає логін користувача та перелік повідомлень між авторизованим користувачем і даним користувачем.

Заявлення про відповідність знахідки втраті реалізується функцією `publish_coincedence (request, founditemsituation_slug, lostitemsituation_slug)`, що на основі форми `CoincedenceForm` створює відповідну заявку, зв'язуючи її

зі знахідкою `founditemsituation_slug` та з втратою `lostitemsituation_slug`, при цьому витягаючи об'єкти відповідних оголошень та перевіряючи, що одне з них належить авторизованому користувачу.

Для відхилення відповідності знахідки втраті використовується функція `cancel_coinedence (request, coinedence)`, яка для заявки з ідентифікатором `coinedence` змінює значення атрибута `is_active` на `False`. Перевіряється, що авторизований користувач є саме другою стороною заявки, тобто він не створив заявку.

Для підтвердження відповідності знахідки втраті використовується функція `approve_coinedence (request, coinedence)`, яка для заявки з ідентифікатором `coinedence` змінює значення атрибута `is_active` на `False`, але при цьому на відміну від відхилення одразу створює в БД об'єкт моделі `Order`, за яким реалізується доставка. Перевіряється, що авторизований користувач є саме другою стороною заявки, тобто він не створив заявку.

Замовлення доставки власних речей реалізується функцією представлення `edit_order (request, order)` з параметром `order`, який визначає ідентифікатор замовлення, до якого потрібно внести дані про доставку. Фактично при цьому виконується модифікація даних, адже замовлення на цей момент вже створено під час підтвердження відповідної заявки іншою стороною. При реалізації функції виконується перевірка прав: авторизований користувач має бути вказаний у замовленні в атрибуті `account`.

Створення зустрічі для повернення речей реалізується функцією `publish_appointment (request, order)`, що на основі форми `AppointmentForm` створює відповідну зустріч, зв'язуючи її з замовленням з ідентифікатором `order`, при цьому перевіряючи, що саме авторизований користувач вказаний у замовленні.

Модифікація параметрів зустрічі для повернення речей реалізується функцією `edit_appointment (request, appointment)`, яка на основі форми `AppointmentForm` вносить зміни в раніше збережений у БД об'єкт зустрічі,

перевіряючи при цьому, що воно стосується замовлення, в якому вказано авторизованого користувача.

Перевірка власництва речі шляхом опитування реалізується функцією представлення `publish_founditemsituationanswer` (`request`, `foundapplication`), де для заявки про власництво речей з ідентифікатором `foundapplication` створюється об'єкт моделі `Founditemsituationanswer`, до якого на основі множини форм `QuestionanswerForm`, що відповідають кількості запитань стосовно цього оголошення про знахідку, прив'язується відповідна кількість відповідей авторизованого користувача. Якщо у запитанні автором зазначено необхідність перевірки (поле `check_answer` істинне), то в кінці відбувається перевірка результатів, а загальний результат вноситься в створений об'єкт моделі `Founditemsituationanswer`. Після отримання результату відбувається перевірка, описана вище, щодо частоти відмов стосовно заявок користувача з можливістю блокування акаунта в результаті, якщо перевірка була автоматичною. Загалом дане представлення запускається автоматично у випадку створення заявки користувачем стосовно знахідки, за якою визначена автоматизована перевірка, тобто не заблокована автоматична перевірка, а також є наявні запитання.

Відтворення переліку всіх запланованих відправлень і зустрічей для користувача реалізується функцією представлення `account_appointments` (`request`), яка для авторизованого користувача вибирає об'єкти моделей `Appointment` і `Order`, які стосуються замовлень або є замовленнями, в яких визначений авторизований користувач.

Створення опитування для перевірки заявника за знахідкою реалізується за допомогою функції представлення `publish_founditemsituationquestion` (`request`, `founditemsituation_slug`), яка для оголошення про знахідку з ідентифікатором URL `founditemsituation_slug` на основі форми `Founditemsituationquestion` створює відповідне запитання на зберігає його до БД.

3.3 Висновки за розділом 3

Програму реалізовано: описано порядок виконання і представлено функції, які реалізують функціональні вимоги до програми.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для розв’язання ситуацій, коли людина знайшла загублені кимось речі та бажає знайти власника, або коли вона навпаки загубила свої речі та бажає їх повернути. Для цього мають надаватися можливості виконати пошук відповідних речей, створити власне оголошення про знахідку або втрату та засоби перевірки того, чи точно заявник може бути власником речей, щоб позбутися шахрайства.

4.2 Умови виконання програми

Обов’язково для виконання програми на сервері має бути встановлено інтерпретатор мови Python, систему для керування БД MySQL при підтримці наступної апаратної частини: процесор, мінімальна конфігурація: Intel Core 2 або подібний, оперативна пам’ять з доступом не менше ніж до 4 Гб, жорсткий диск з доступним обсягом не менше ніж 100 Гб.

4.3 Виконання і тестування програми

Для тестування було виконано перевірку функціональних вимог за забезпечення вказаного рівня апаратних вимог та з програмною підтримкою клієнтської сторони на основі браузерів Chrome і Microsoft Edge. Це дозволило не тільки встановити повноцінну реалізацію функцій програми, але і перевірити правильність відображення елементів сторінок, які були сформовані на основі створених шаблонів.

При зверненні з клієнтської сторони будь-який користувач отримує один і той же загальний інтерфейс. При цьому якщо для виконання функції потрібна авторизація, а користувач не авторизований, то автоматично відбувається перенаправлення на сторінку авторизації.

Меню розбито на пункти, що включають для всіх користувачів “Знахідки” для перегляду всіх оголошень про знахідки, створених авторизованим користувачем, “Втрати” для перегляду всіх оголошень про втрати, створених авторизованим користувачем, “Заявки” для роботи з поданими заявками про власництво, “Переписка” для перегляду і відправлення повідомлень користувачам, які поставили питання або висловити бажання встановити контакт, “Вхід” або “Вихід” для того щоб увійти до власного облікового запису або навпаки і засоби пошуку серед знахідок та втрат відповідно.

Сторінка оголошення про знахідку як і сторінка оголошення про втрату має верхню частину з загальною інформацією (рис. 4.1), кнопками для оформлення заявки про власництво та можливістю написати повідомлення.

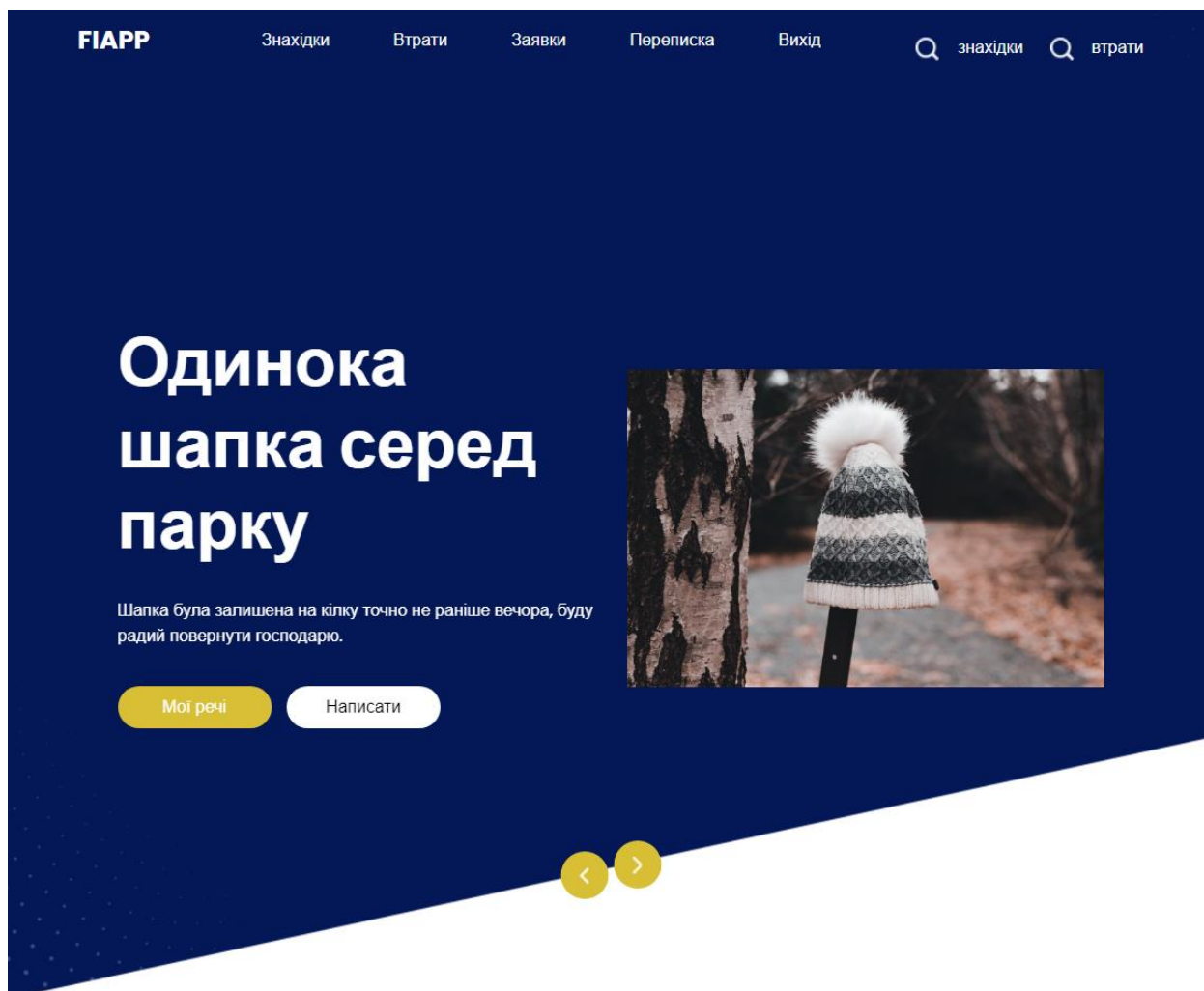


Рисунок 4.1 – Оголошення про знахідку

У верхній частині сторінки демонструються фотографії всіх речей, які входять до знахідки, з переходом між ними. Нижче (рис. 4.2) детально виводиться інформація про те, чи проводиться після оформлення заявки про власництво перевірка шляхом опитування. Якщо проводиться, то одразу після заповнення форми відкривається сторінка проведення опитування з запитаннями і полями для відповідей.

Оголошення активне ●

передбачена перевірка власності

Київ. Come and Stay

Тип організації: кав'ярня
Адреса: Велика Васильківська, 23В

Знайдено 24.05.2023

Створено оголошення 25.05.2023

Сумки / Ділова сумка

Виробник: Vittorio Safino
Сумка з секретом.
Колір: коричневий

Відобразити

Акcesуари / Гаманець

Виробник: Wittchen
Гаманець з деякими речима, але без документів. Має характерні сліди.
Колір: коричневий

Відображено



<

>

Рисунок 4.2 – Деталізація знахідки

За оголошенням надаються дані про те, де були знайдені речі, а внизу сторінки ця адреса відображається на мапі. Якщо знахідка була в транспорті, то відображається деталізований маршрут. Також виводяться дані про дату знахідки, створення оголошення. Приводяться дані про кожну річ. За кожною річчю виводяться всі атрибути зі значеннями, які були встановлені і які

можна відображати. При натисканні на кнопку “Відобразити” у правій частині відображається фотографія речі з можливістю переходу між речами.

При виконанні пошуку потрібно задати параметри, а далі відображаються результати (рис. 4.3). За кожним результатом виводиться назва оголошення, опис, перша фотографія першої речі, адреса, де було знайдено речі. Кнопка “Відкрити” дозволяє відкрити сторінку оголошення з деталізацією, описану вище. Кнопка “Мої речі” дозволяє одразу оформити заявку на власництво. Якщо користувач має оголошення про втрату речі, а знайшов оголошення про знахідку, то він може зі сторінки заявки перейти за посиланням на сторінку заявки про співпадіння.

Результати: знайдені речі ●



Сумка з речами і секретом ●

Сумка були залишена на столику в одному з закладів торгового центру.

📍 Lavina Mall / 22.05.2023
Торгово-розважальний центр
Київ, вулиця Берковецька, 6Д

Відкрити
Мої речі



Загублена дорожня сумка на вокзалі ●

Сумка порожня, не знаю чи в ній щось було. Буду рада повернути.

📍 Київ Центральний / 22.05.2023
Залізничний вокзал
Київ, Вокзальна площа, 1

Відкрити
Мої речі

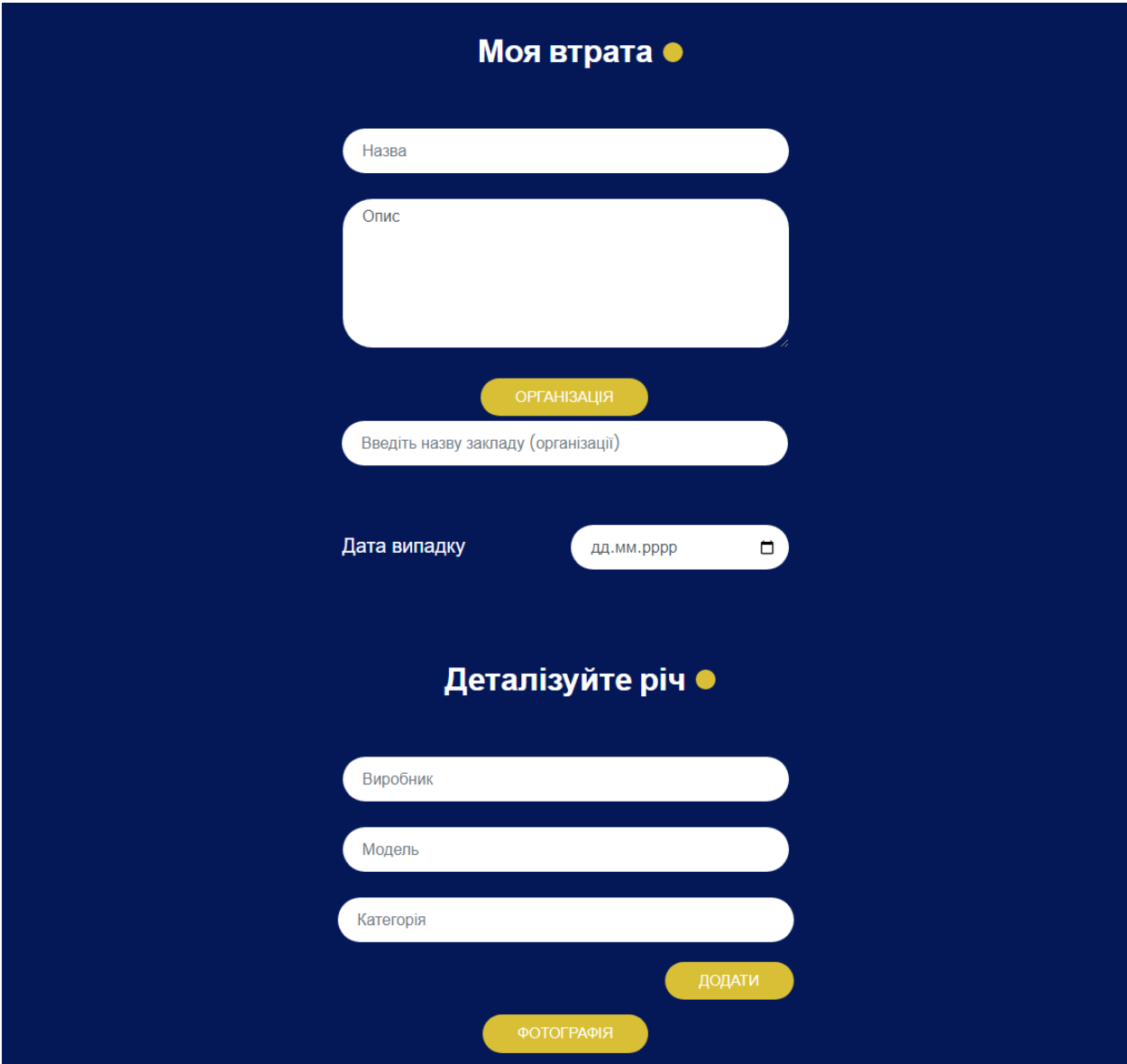


Персоналізована знахідка ●

Містить деяку інформацію про власника, але на жаль не повну, тому відшукати поки що не вдалось.

Рисунок 4.3 – Пошук серед знахідок

Перейти до публікації оголошення про втрату (рис. 4.4) або знахідку можна зі сторінки перегляду своїх відповідних оголошень, де міститься кнопка для публікації нової знахідки або втрати. На цій сторінці потрібно вказати загальну інформацію про втрату або знахідку. Для втрати це назва, опис. Кнопка “Організація” дозволяє ввести в поле пошуку назву ресторану, торгового центру. При натисканні на неї вона змінює назву на “Мапа” і відкривається мапа, де можна обрати потрібну точку, де втрачено речі. При наступному натисканні кнопка змінює назву на “Місто” і потрібно в полі пошуку ввести назву міста. І так далі назва буде змінюватися по колу.



Моя втрата ●

Назва

Опис

ОРГАНІЗАЦІЯ

Введіть назву закладу (організації)

Дата випадку

дд.мм.рррр

Деталізуйте річ ●

Виробник

Модель

Категорія

ДОДАТИ

ФОТОГРАФІЯ

Рисунок 4.4 – Публікація оголошення про втрату

Після визначення загальних даних потрібно задати одну річ, яку було втрачено. Поле виробника працює за принципом пошуку з випадającym переліком при введенні. Поле категорії працює так само, але в кінці потрібно натиснути на кнопку “Додати”, тоді категорія відобразиться в переліку нижче, до якого далі можна додати наступну категорію. Так само нижче можна задати атрибути зі значеннями і вказати, чи вони відкриті. Для завантаження фотографії з вибором відповідного файлу використовується кнопка “Фотографія”. Далі оголошення потрібно опублікувати. Після цього зі сторінки оголошення можна натиснути на кнопку для додавання речей і внести інформацію про ще одну річ.

4.4 Висновки за розділом 4

Програмне забезпечення розроблено у загальному режимі, який використовується для всіх користувачів, у тому числі неавторизованих, що сприяє розумінню можливостей і простоті доступу. Основні варіанти роботи для пошуку господарів знайдених речей і своїх речей господарями описані.

ВИСНОВКИ

Виконано виділення програмних засобів, які зараз можуть використовуватися для пошуку господарів знайдених речей: iLost, Lost and found networks, LostAndFound. Проаналізовано наявні та відсутні у них можливості. Запропоновано в програмному додатку, що було розроблено в підсумку, представити засоби визначення маршрутів для знахідок у транспорті, перевірки шахрайства, що є важливим для людей, які знайшли речі, та готові повернути їх господарю, але не готові витратити час на хибні заявки, кількість яких може збільшитися значно, якщо речі є цінними. Таким чином досягнуто виконання завдання аналізу програмних засобів пошуку господарів знайдених речей і визначення концепції програми.

Досягнуто виконання завдання проектування програмного забезпечення пошуку господарів знайдених речей, включаючи вибір мови програмування Python, моделювання вимог до програми на основі діаграми варіантів використання, визначення структури роботи з даними на основі БД.

Існування простого і зручного у використанні інструмента для пошуку господарів знайдених речей на практиці може сприяти тому, щоб люди активніше намагалися допомогти тим, хто загубив власні речі, які можуть бути цінними взагалі тільки господарю або навпаки викликати інтерес шахраїв. Тому наявність такої платформи пошуку може сприяти позитивному соціальному ефекту, довірі в суспільстві в певній мірі. Внесення чіткої інформації про виявлену знахідку, підтримка авторів оголошень щодо виявлення шахраїв, визначення повного ланцюжка переміщень при виявленні знахідки в транспорті та врахування цього ланцюжка в пошуку потенційно мають важливий ефект для досягнення результативності всього пошуку.

Досягнуто виконання завдання реалізації вебдодатку пошуку господарів знайдених речей, в результаті чого описано порядок виконання і представлено функції, які реалізують всі функціональні вимоги до програми.

Усі отримані результати призвели до досягнення мети розробити програмне забезпечення пошуку господарів знайдених речей для сприяння підвищенню результативності даного процесу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. The search engine for lost and found. iLost [Electronic resource]. – Access mode : <https://ilost.co/>.
2. Lost & Found – Global Lost & Found Search Engine to report lost [Electronic resource]. – Access mode : <https://lostandfoundnetworks.com/>.
3. LostAndFound.com [Electronic resource] : The Internet Lost and Found. – Access mode : <https://www.lostandfound.com/>.
4. Our Documentation. Python.org [Electronic resource]. – Access mode : <https://www.python.org/doc/>.
5. Python in a Nutshell [Text] : A Desktop Quick Reference / Alex Martelli, Anna Ravenscroft, Steve Holden et al. – Massachusetts : O'Reilly Media, 2023. – 735 p.
6. Hartl, M. Ruby on Rails Tutorial [Text] / H. Hartl. – Boston : Addison-Wesley Professional, 2020. – 848 p.
7. Web Development with Django [Text] : Learn to build modern web applications with a Python-based framework / Ben Shaw, Saurabh Badhwar, Andrew Bird et al. – Birmingham : Packt Publishing, 2021. – 826 p.
8. Botros, S. High Performance MySQL [Text] : Proven Strategies for Operating at Scale / S. Botros, J. Tinley. – Massachusetts : O'Reilly Media, 2021. – 386 p.
9. Seotech Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page289/seotech>.

ДОДАТОК А
Технічне завдання

Вступ

Якщо людина загубила свої речі, це ще не означає, що вона втратила їх назавжди. Однак у випадках, коли речі не були вкрадені, а просто десь залишені та потім кимось підібрані, не так просто повернути їх власнику. Для цього потрібен програмний інструмент, який дозволить знайти людині, яка загубила речі, людину, яка їх знайшла. Це передбачає потужні інструменти пошуку, а також перевірку власності. Саме для надання таких інструментів має бути створено розробку в роботі. Вона зокрема має враховувати специфічні ситуації, коли це може трапитися і за яких передумов цей пошук має виконуватися. Наприклад, однією з таких передумов є шахрайські задуми, коли людина, побачивши речі в оголошенні, вирішує неправомірно ними заволодіти. Ці проблеми мають бути вирішені в роботі.

A.1 Підстави для розробки

Завдання на дипломну кваліфікаційну роботу визначено на підставі наказу № 87 від 4 квітня 2023 р. за Національним університетом «Запорізька політехніка» за темою «Програмне забезпечення пошуку господарів знайдених речей».

A.2 Призначення розробки

Програма призначена для розв'язання ситуацій, коли людина знайшла загублені кимось речі та бажає знайти власника, або коли вона навпаки загубила свої речі та бажає їх повернути. Для цього мають надаватися можливості виконати пошук відповідних речей, створити власне оголошення про знахідку або втрату та засоби перевірки того, чи точно заявник може бути власником речей, щоб позбутися шахрайства.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для підтримки описаного призначення програма повинна надавати користувачам наступний набір функцій для роботи:

- публікація оголошення про знаходження речей;
- модифікація оголошення про знаходження речей;
- деактивація оголошення про знаходження речей;
- публікація оголошення про втрату речей;
- модифікація оголошення про втрату речей;
- деактивація оголошення про втрату речей;
- пошук речей серед знайдених;
- пошук речей серед втрачених;
- заявлення про власництво речей;
- відхилення заявки про власництво речей;
- підтвердження власництва речей;
- відтворення переліку заявок про власництво за знахідкою;
- створення опитування для перевірки заявника за знахідкою;
- перевірка власництва речі шляхом опитування;
- блокування доступу користувача при великій кількості відхилень запитів;
- відтворення переліку всіх знахідок користувача;
- відтворення переліку всіх втрат користувача;
- відправлення повідомлення користувачу;
- заявлення про знаходження речей;
- відхилення заявки про знаходження речей;
- підтвердження заявки про знаходження речей;
- відтворення переліку заявок за втратою;
- заявлення про відповідність знахідки втраті;
- підтвердження відповідності знахідки втраті;

- відхилення відповідності знахідки втраті;
- замовлення доставки власних речей;
- створення зустрічі для повернення речей;
- модифікація параметрів зустрічі для повернення речей;
- відтворення переліку всіх заявок до користувача;
- відтворення переліку всіх необроблених заявок до користувача;
- відтворення переліку всіх заявок, створених користувачем;
- відтворення переліку всіх запланованих відправлень і зустрічей для користувача;
- перегляд сторінки знахідки;
- перегляд сторінки втрати.

A.3.2 Вимоги до складу та параметрів технічних засобів

Апаратні вимоги до серверної сторони:

- процесор у мінімальній конфігурації Intel Core 2 або еквівалент;
 - оперативна пам'ять з доступом не менше ніж до 4 Гб;
 - жорсткий диск з бажаним доступним обсягом не менше ніж 100 Гб,
- а у випадку меншого обсягу контролювати обсяг даних у БД при внесенні фотографій речей, при цьому в будь-якому випадку потрібний обсяг може і значно збільшуватися, що залежить від кількості користувачів, кількості та якості фотографій, які вони завантажують.

A.4 Порядок контролю та приймання

Основою щодо реалізації контролю та приймання дипломної кваліфікаційної роботи є завдання на роботу і безпосередньо дане технічне завдання. За успішного виконання цих завдань у встановлені строки робота після проходження всіх підготовчих процедур виноситься на захист перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

    from .models import Situationitem, Founditemsituation,
    Lostitemsituation, Company, City, Producer, Localcategory,
    Globalcategory, Founditemsituationquestion, Attribute, Account,
    Situationitemattribute
    from .forms import FounditemsituationForm,
    SituationitemForm, LostitemsituationForm
    from django.shortcuts import redirect
    import datetime
    import geopy.distance

    def founditemsituation(request, foundsituation_slug =
None):
        founditemsituation = None
        situationitems = []
        if foundsituation_slug:
            try:
                founditemsituation =
Founditemsituation.objects.select_related("company").select_rela
ted("city").select_related("company__city").select_related("acco
untaddress").select_related("accountaddress__city").select_relat
ed("accountaddress__account").select_related("company__companysp
here").prefetch_related("foundroutes").get(slug =
foundsituation_slug)
            except Founditemsituation.DoesNotExist:
                founditemsituation = None
            if founditemsituation:
                situationitems =
Situationitem.objects.filter(founditemsituation =
founditemsituation).prefetch_related("itemphotos").prefetch_rela
ted("globalcategories").prefetch_related("localcategories").pref
etch_related("situationitemattributes").select_related("situatio
nitemattributes__attribute").select_related("producer").order_by
("id")
                if founditemsituation.accountaddress.account ==
request.user:
                    questions =
Founditemsituationquestion.objects.filter(founditemsituation =
founditemsituation)
            else:
                questions = []
            return render(request, "fiapp/founditemsituation.html",
{"founditemsituation": founditemsituation, "situationitems":
situationitems, "questions": questions)

    def lostitemsituation(request, lostitemsituation_slug =
None):

```

```

        lostitemsituation = None
        situationitems = []
        if lostitemsituation_slug:
            lostitemsituation =
Lostitemsituation.objects.select_related("company").select_relat
ed("city").select_related("company__city").select_related("accou
nt").select_related("company__companysphere").prefetch_related("
lostroutes").get(slug = lostitemsituation_slug, None)
            if lostitemsituation:
                situationitems =
Situationitem.objects.filter(lostitemsituation =
lostitemsituation).prefetch_related("itemphotos").prefetch_relat
ed("globalcategories").prefetch_related("localcategories").prefe
tch_related("situationitemattributes").select_related("situation
itemattributes__attribute").select_related("producer").order_by(
"id")
                return render(request, "fiapp/lostitemsituation.html",
{"lostitemsituation": lostitemsituation, "situationitems":
situationitems})

```

```

def search_found_items(request):
    situations = []
    searched = False
    det = False
    company = self.request.GET.get('company', None)
    if company:
        try:
            company = Company.objects.get(id =
int(company))
        except Company.DoesNotExist:
            company = None
        if company:
            situations =
Founditemsituation.objects.filter(company = company, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_photos').prefetch_related('situationitems_attributes')
.prefetch_related('situationitems_globalcategories').prefetch_re
lated('situationitems_localcategories').prefetch_related('foundr
outes').select_related("company").select_related("city").select_
related("company__city").select_related("company__companysphere"
)
            searched = True
            city = self.request.GET.get('city', None)
            if city:
                try:
                    city = City.objects.get(id = int(city))

```

```

        except City.DoesNotExist:
            city = None
        if city and not searched:
            det = True
            situations =
Founditemsituation.objects.filter(city = city, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r
elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

            searched = True
        elif city:
            situations = situations.filter(city = city)
        cities = self.request.GET.get('cities', None)
        cities_id = []
        cities_list = cities.split(',')
        found_items = self.request.GET.get('items', None)
        for city in cities_list:
            try:
                city = City.objects.get(id = int(city))
            except City.DoesNotExist:
                city = None
            if city:
                cities_id.append(city)
        if cities_id and not searched and not det:
            det = True
            situations =
Founditemsituation.objects.filter(foundroutes__in = cities_id,
is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r
elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

            searched = True
        elif cities_id and not det:
            det = True
            situations = situations.filter(foundroutes__in =
cities_id)
        found_from = self.request.GET.get('found_from', None)
        found_from = found_from.split('.')

```

```

        found_from = datetime.datetime(int(found_from[2]),
int(found_from[1]), int(found_from[0]))
        found_to = self.request.GET.get('found_to', None)
        found_to = found_to.split('.')
        found_to = datetime.datetime(int(found_to[2]),
int(found_to[1]), int(found_to[0]))
        if searched and (found_from or found_to):
            situations = situations.filter(found_date__gte =
found_from, found_date__lte = found_to)
        elif not searched and (found_from or found_to):
            situations =
Founditemsituation.objects.filter(found_date__gte = found_from,
found_date__lte = found_to, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r
elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

        searched = True
        producers = self.request.GET.get('producers', None)
        producers_id = []
        producers_list = producers.split(',')
        for producer in producers_list:
            try:
                producer = Producer.objects.get(id =
int(producer))
            except Producer.DoesNotExist:
                producer = None
            if producer:
                producers_id.append(producer)
        det2 = False
        if producers_id and searched and not found_items:
            det2 = True
            situations =
situations.filter(situationitems__producer__in = producers_id)
        elif producers_id and not found_items:
            det2 = True
            situations =
Founditemsituation.objects.filter(situationitems__producer__in =
producers_id, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r

```

```

elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

        searched = True
        localcategories = self.request.GET.get('local', None)
        localcategories_id = []
        localcategories_list = localcategories.split(',')
        for localcategory in localcategories_list:
            try:
                localcategory = Localcategory.objects.get(id =
int(localcategory))
            except Localcategory.DoesNotExist:
                localcategory = None
            if localcategories:
                localcategories_id.append(localcategory)
        if localcategories_id and searched and not found_items:
            det2 = True
            situations =
situations.filter(situationitems__localcategories__in =
localcategories_id)
            elif localcategories_id:
                det2 = True
                situations =
Founditemsituation.objects.filter(situationitems__localcategor
ies__in = localcategories_id, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r
elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

        searched = True
        globalcategories = self.request.GET.get('global', None)
        globalcategories_id = []
        globalcategories_list = globalcategories.split(',')
        for globalcategory in globalcategories_list:
            try:
                globalcategory = Globalcategory.objects.get(id
= int(globalcategory))
            except Globalcategory.DoesNotExist:
                globalcategory = None
            if globalcategory:
                globalcategories_id.append(globalcategory)
        if globalcategories_id and searched and not
found_items:

```

```

        det2 = True
        situations =
situations.filter(situationitems__globalcategories__in =
globalcategories_id)
        elif globalcategories_id and not found_items:
            det2 = True
            situations =
Founditemsituation.objects.filter(situationitems__globalcategori
es__in = globalcategories_id, is_active =
True).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('foundroutes').select_related("company").select_r
elated("city").select_related("company__city").select_related("c
ompany__companysphere").prefetch_related('situationitems_photos'
)

        searched = True
        address_long = self.request.GET.get('address_long',
None)

        address_long = float(address_long)
        address_latt = self.request.GET.get('address_latt',
None)

        address_latt = float(address_latt)
        distance = self.request.GET.get('distance', None)
        search_distance = int(distance)
        if not searched:
            situations =
Founditemsituation.objects.all(is_active = True)
            if address_long and not det:
                situations_dist = []
                search_address = (address_latt, address_long)
                for situation in situations:
                    situation_address = (situation.address_latt,
situation.address_long)
                    distance =
geopy.distance.geodesic(situation_address, search_address).km
                    if distance.km <= search_distance:
                        situations_dist.append(situation)
                situations = situations_dist
            elif not address_long and not found_items:
                globalcategories = Globalcategory.objects.all()
                cities =
City.objects.all().select_related("country")
                localcategories = Localcategory.objects.all()
                attributes = Attribute.objects.all()
                producers = Producer.objects.all()

```

```

        return render(request,
"fiapp/search_found_items.html", {"situations": [],
"globalcategories": globalcategories, "localcategories":
localcategories, "cities": cities, "attributes": attributes,
"producers": producers })
        found_items_list = found_items.split(',')
        item_parameters = []
        if not det2:
            for found_item in found_items_list:
                item_parameter = {}
                if found_item[0] == 'l' and found_item[0] == '-
':
                    parameter = []
                    found_item = found_item[2:]
                    locals = found_item.split(';')
                    for locali in locals:
                        try:
                            loc = Localcategory.objects.get(id
= locali)

                        except Localcategory.DoesNotExist:
                            loc = None
                        if loc:
                            parameter.append(loc)
                    item_parameter['local'] = parameter
                elif found_item[0] == 'g' and found_item[0] ==
'-':
                    parameter = []
                    found_item = found_item[2:]
                    globals = found_item.split(';')
                    for glob in globals:
                        try:
                            glob =
Globalcategory.objects.get(id = glob)
                        except Globalcategory.DoesNotExist:
                            glob = None
                        if glob:
                            parameter.append(glob)
                    item_parameter['global'] = parameter
                elif found_item[0] == 'm' and found_item[0] ==
'-':
                    item_parameter['model'] = found_item[2:]
                elif found_item[0] == 'p' and found_item[0] ==
'-':
                    producer = found_item[2:]
                    try:

```

```

        producer = Producer.objects.get(id =
producer)

        except Producer.DoesNotExist:
            producer = None
        if producer:
            item_parameter['producer'] = producer
    else:
        attributes = found_item.split('.')
        attributevalues = []
        for attribute in attributes:
            attribute_id = attribute.split('-')[0]
            attribute_value = attribute.split('-
' ) [1]

            try:
                attribute =
Attribute.objects.get(id = attribute_id)
            except Attribute.DoesNotExist:
                attribute = None
            if attribute:
                attribute_valuer = []
                attribute_valuer.append(attribute)

attribute_valuer.append(attribute_value)

attributevalues.append(attribute_valuer)
            item_parameter['attributes'] =
attributevalues

            item_parameters.append(item_parameter)
        foundsituations = []
        search_items = []
        if item_parameter and searched:
            search_items = situations
        else:
            search_items =
Founditemsituation.objects.all(is_active = True)
            if not found_items:
                foundsituations = situations:
            else:
                for situation in situations:
                    coincidence_items = 0
                    for item in situation.situationitems:
                        coincidence_global = False
                        for item_parameter in item_parameters:
                            coincidence = True

```

```

        if item.model !=
item_parameter['model'] or item.producer !=
item_parameter['producer']:
            coincidence = False
        else:
            for loc in
item_parameter['local']:
                if loc not in
item.localcategories:
                    coincidence = False
                if coincidence:
                    for loc in
item_parameter['global']:
                        if loc not in
item.globalcategories:
                            coincidence = False
                        if coincidence:
                            for loc in
item_parameter['attributes']:
                                attribute =
Situationitemattribute(situationitem = item, attribute = loc[0],
attribute_value = loc[1], secret = False)
                                if attribute not in
item.attributes:
                                    coincidence = False
                                if coincidence:
                                    coincidence_global = True
                                if coincidence_global:
                                    coincidence_items =
coincidence_items + 1
                                    if coincidence_items >=
len(item_parameters):
                                        foundsituations.append(situation)
                                        globalcategories = Globalcategory.objects.all()
                                        cities = City.objects.all().select_related("country")
                                        localcategories = Localcategory.objects.all()
                                        attributes = Attribute.objects.all()
                                        producers = Producer.objects.all()
                                        return render(request, "fiapp/search_found_items.html",
{"situations": foundsituations, "globalcategories":
globalcategories, "localcategories": localcategories, "cities":
cities, "attributes": attributes, "producers": producers })

def search_lost_items(request):
    situations = []
    searched = False

```

```

        company = self.request.GET.get('company', None)
        if company:
            try:
                company = Company.objects.get(id =
int(company))
            except Company.DoesNotExist:
                company = None
            if company:
                situations =
Lostitemsituation.objects.filter(company =
company).prefetch_related('situationitems').prefetch_related('si
tuationitems_attributes').prefetch_related('situationitems_globa
lcategories').prefetch_related('situationitems_localcategories')
.prefetch_related('lostroutes').select_related("company").select_
related("city").select_related("company__city").select_related(
"company__companysphere").prefetch_related('situationitems_photo
s')

                searched = True
        city = self.request.GET.get('city', None)
        if city:
            try:
                city = City.objects.get(id = int(city))
            except City.DoesNotExist:
                city = None
            if city and not searched:
                situations =
Lostitemsituation.objects.filter(city =
city).prefetch_related('situationitems').prefetch_related('situa
tionitems_attributes').prefetch_related('situationitems_globalca
tegories').prefetch_related('situationitems_localcategories').pr
efetch_related('lostroutes').select_related("company").select_re
lated("city").select_related("company__city").select_related("co
mpany__companysphere").prefetch_related('situationitems_photos')
                searched = True
            elif city:
                situations = situations.filter(city = city)
        cities = self.request.GET.get('cities', None)
        cities_id = []
        cities_list = cities.split(',')
        for city in cities_list:
            try:
                city = City.objects.get(id = int(city))
            except City.DoesNotExist:
                city = None
            if city:
                cities_id.append(city)

```

```

        if cities_id and not searched:
            situations =
Lostitemsituation.objects.filter(foundroutes__in =
cities_id).prefetch_related('situationitems').prefetch_related('
situationitems_attributes').prefetch_related('situationitems_glo
balcategories').prefetch_related('situationitems_localcategories
').prefetch_related('lostroutes').select_related("company").sele
ct_related("city").select_related("company__city").select_relate
d("company__companysphere").prefetch_related('situationitems_pho
tos')

            searched = True
        elif cities_id:
            situations = situations.filter(lostroutes__in =
cities_id)
            lost_from = self.request.GET.get('lost_from', None)
            lost_from = lost_from.split('.')
            lost_from = datetime.datetime(int(lost_from[2]),
int(lost_from[1]), int(lost_from[0]))
            if searched and lost_from:
                situations = situations.filter(lost_date__lte =
lost_from)
            elif not searched and lost_from:
                situations =
Lostitemsituation.objects.filter(lost_date__lte =
lost_from).prefetch_related('situationitems').prefetch_related('
situationitems_attributes').prefetch_related('situationitems_glo
balcategories').prefetch_related('situationitems_localcategories
').prefetch_related('lostroutes').select_related("company").sele
ct_related("city").select_related("company__city").select_relate
d("company__companysphere").prefetch_related('situationitems_pho
tos')

                searched = True
                producers = self.request.GET.get('producers', None)
                producers_id = []
                producers_list = producers.split(',')
                for producer in producers_list:
                    try:
                        producer = Producer.objects.get(id =
int(producer))
                    except Producer.DoesNotExist:
                        producer = None
                    if producer:
                        producers_id.append(producer)
                if producers_id and searched:
                    situations =
situations.filter(situationitems__producer__in = producers_id)

```



```

        globalcategory = None
        if globalcategory:
            globalcategories_id.append(globalcategory)
        if globalcategories_id and searched:
            situations =
situations.filter(situationitems__globalcategories__in =
globalcategories_id)
            elif globalcategories_id:
                situations =
Lostitemsituation.objects.filter(situationitems__globalcategorie
s__in =
globalcategories_id).prefetch_related('situationitems').prefetch
_related('situationitems_attributes').prefetch_related('situatio
nitems_globalcategories').prefetch_related('situationitems_local
categories').prefetch_related('lostroutes').select_related("comp
any").select_related("city").select_related("company__city").sel
ect_related("company__companysphere").prefetch_related('situatio
nitems_photos')
                searched = True
                address_long = self.request.GET.get('address_long',
None)
                address_long = float(address_long)
                address_latt = self.request.GET.get('address_latt',
None)
                address_latt = float(address_latt)
                distance = self.request.GET.get('distance', None)
                search_distance = int(distance)
                lost_items = self.request.GET.get('items', None)
                if not searched:
                    situations = Lostitemsituation.objects.all()
                if address_long:
                    situations_dist = []
                    search_address = (address_latt, address_long)
                    for situation in situations:
                        situation_address = (situation.address_latt,
situation.address_long)
                        distance =
geopy.distance.geodesic(situation_address, search_address).km
                        if distance.km <= search_distance:
                            situations_dist.append(situation)
                    situations = situations_dist
                elif not address_long and not lost_items:
                    globalcategories = Globalcategory.objects.all()
                    cities =
City.objects.all().select_related("country")
                    localcategories = Localcategory.objects.all()

```

```

        attributes = Attribute.objects.all()
        producers = Producer.objects.all()
        return render(request,
"fiapp/search_lost_items.html", {"situations": [],
"globalcategories": globalcategories, "localcategories":
localcategories, "cities": cities, "attributes": attributes,
"producers": producers })
        lost_items_list = lost_items.split(',')
        item_parameters = []
        for lost_item in lost_items_list:
            item_parameter = {}
            if lost_item[0] == 'l' and lost_item[0] == '-':
                parameter = []
                lost_item = lost_item[2:]
                locals = lost_item.split('.')
                for locali in locals:
                    try:
                        loc = Localcategory.objects.get(id =
locali)

                        except Localcategory.DoesNotExist:
                            loc = None
                        if loc:
                            parameter.append(loc)
                    item_parameter['local'] = parameter
            elif lost_item[0] == 'g' and lost_item[0] == '-':
                parameter = []
                lost_item = lost_item[2:]
                globals = lost_item.split('.')
                for glob in globals:
                    try:
                        glob = Globalcategory.objects.get(id =
glob)

                        except Globalcategory.DoesNotExist:
                            glob = None
                        if glob:
                            parameter.append(glob)
                    item_parameter['global'] = parameter
            elif lost_item[0] == 'm' and lost_item[0] == '-':
                item_parameter['model'] = lost_item[2:]
            elif lost_item[0] == 'p' and lost_item[0] == '-':
                producer = lost_item[2:]
                try:
                    producer = Producer.objects.get(id =
producer)

                    except Producer.DoesNotExist:
                        producer = None

```

```

        if producer:
            item_parameter['producer'] = producer
    else:
        attributes = lost_item.split('.')
        attributevalues = []
        for attribute in attributes:
            attribute_id = attribute.split('-')[0]
            attribute_value = attribute.split('-')[1]
            try:
                attribute = Attribute.objects.get(id =
attribute_id)

            except Attribute.DoesNotExist:
                attribute = None
            if attribute:
                attribute_valuer = []
                attribute_valuer.append(attribute)

attribute_valuer.append(attribute_value)
                attributevalues.append(attribute_valuer)
            item_parameter['attributes'] = attributevalues
            item_parameters.append(item_parameter)
        lostsituations = []
        search_items = []
        if item_parameter and searched:
            search_items = situations
        else:
            search_items = Lostitemsituation.objects.all()
        if not lost_items:
            lostsituations = situations
        else:
            for situation in situations:
                coincidence_items = 0
                for item in situation.situationitems:
                    coincidence_global = False
                    for item_parameter in item_parameters:
                        coincidence = True
                        if item.producer !=
item_parameter['producer']:
                            coincidence = False
                        else:
                            for loc in item_parameter['local']:
                                if loc not in
item.localcategories:
                                    coincidence = False
                                if coincidence:

```

```

                                for loc in
item_parameter['global']:
                                if loc not in
item.globalcategories:
                                coincidence = False
                                if coincidence:
                                    for loc in
item_parameter['attributes']:
                                attribute =
Situationitemattribute(situationitem = item, attribute = loc[0],
attribute_value = loc[1], secret = False)
                                if attribute not in
item.attributes:
                                coincidence = False
                                if coincidence:
                                    coincidence_global = True
                                if coincidence_global:
                                    coincidence_items = coincidence_items +
1
                                if coincidence_items == len(item_parameters):
                                    lostsituations.append(situation)
                                globalcategories = Globalcategory.objects.all()
                                cities = City.objects.all().select_related("country")
                                localcategories = Localcategory.objects.all()
                                attributes = Attribute.objects.all()
                                producers = Producer.objects.all()
                                return render(request, "fiapp/search_lost_items.html",
{"situations": lostsituations, "globalcategories":
globalcategories, "localcategories": localcategories, "cities":
cities, "attributes": attributes, "producers": producers })

def founditemsituations(request, user_account = None):
    if not user_account
        account = request.user
    else:
        try:
            account = Account.objects.get(username =
user_account)
        except Account.DoesNotExist:
            account = None
    if account:
        founditemsituations =
Founditemsituation.objects.filter(accountaddress__account =
account).order_by("-found_date")
    else:
        founditemsituations = []

```

```

        return render(request,
"fiapp/founditemsituations.html", {"situations":
founditemsituations })

    def founditemsituations(request, user_account = None):
        if not user_account
            account = request.user
        else:
            try:
                account = Account.objects.get(username =
user_account)
            except Account.DoesNotExist:
                account = None
        if account:
            founditemsituations =
Founditemsituation.objects.filter(accountaddress__account =
account).order_by("-found_date")
        else:
            founditemsituations = []
        return render(request,
"fiapp/founditemsituations.html", {"situations":
founditemsituations })

    def lostitemsituations(request, user_account = None):
        if not user_account
            account = request.user
        else:
            try:
                account = Account.objects.get(username =
user_account)
            except Account.DoesNotExist:
                account = None
        if account:
            lostitemsituations =
Lostitemsituation.objects.filter(accountaddress__account =
account).order_by("-lost_date")
        else:
            lostitemsituations = []
        return render(request, "fiapp/lostitemsituations.html",
{"situations": lostitemsituations })

    def edit_founditemsituation(request, situation_slug):
        try:
            founditemsituation =
Founditemsituation.objects.select_related("company").select_rela
ted("city").select_related("company__city").select_related("acco

```

```

untaddress").select_related("accountaddress__city").select_relat
ed("accountaddress__account").select_related("company__companysp
here").prefetch_related("foundroutes").get(slug =
situation_slug)
    except Founditemsituation.DoesNotExist:
        founditemsituation = None
        account = request.user
        if founditemsituation and account.is_active and
founditemsituation.accountaddress.account == account:
            if request.method == "POST":
                founditemsituation_form =
FounditemsituationForm(request.POST, instance =
founditemsituation)
                if founditemsituation_form.is_valid():
                    founditemsituation_form.save()
                    founditemsituation_form.save_m2m()
                    return redirect("founditemsituation",
situation_slug)
            else:
                founditemsituation_form =
FounditemsituationForm(None, instance = founditemsituation)
            else:
                founditemsituation_form = None
                situationitems =
Situationitem.objects.filter(founditemsituation =
founditemsituation).prefetch_related("itemphotos").prefetch_rela
ted("globalcategories").prefetch_related("localcategories").pref
etch_related("situationitemattributes").select_related("situatio
nitemattributes__attribute").select_related("producer").order_by
("id")
                return render(request,
"fiapp/edit_founditemsituation.html", {"form":
founditemsituation_form, "situation": founditemsituation,
"items": situationitems })

def edit_lostitemsituation(request, situation_slug):
    try:
        lostitemsituation = Lostitemsituation.objects.
select_related("company").select_related("city").select_related(
"company__city").select_related("account").select_related("compa
ny__companysphere").prefetch_related("lostroutes").get(slug =
situation_slug)
    except Lostitemsituation.DoesNotExist:
        lostitemsituation = None
        account = request.user
        if lostitemsituation:

```

```

        situationitems =
        Situationitem.objects.filter(lostitemsituation =
        lostitemsituation).prefetch_related("itemphotos").prefetch_relat
        ed("globalcategories").prefetch_related("localcategories").prefe
        tch_related("situationitemattributes").select_related("situation
        itemattributes__attribute").select_related("producer").order_by(
        "id")

        if lostitemsituation and account.is_active and
        lostitemsituation.account == account:
            if request.method == "POST":
                lostitemsituation_form =
                LostitemsituationForm(request.POST, instance =
                lostitemsituation)
                if lostitemsituation_form.is_valid():
                    lostitemsituation_form.save()
                    lostitemsituation_form.save_m2m()
                    return redirect("lostitemsituation",
situation_slug)
                else:
                    lostitemsituation_form =
                    LostitemsituationForm(None, instance = lostitemsituation)
            else:
                lostitemsituation_form = None
            return render(request,
"fiapp/edit_lostitemsituation.html", {"form":
lostitemsituation_form, "situation": lostitemsituation, "items":
situationitems })

```

```

def edit_situationitem(request, item_id):
    try:
        situationitem = Situationitem.objects.get(id =
item_id)
    except Situationitem.DoesNotExist:
        situationitem = None
    account = request.user
    lost = situationitem.lostitemsituation
    if not lost:
        found = situationitem.founditemsituation
        check = found.accountaddress.account == account
        number = found.slug
    else:
        check = lost.account == account
        number = lost.slug
    if situationitem and account.is_active and check:
        if request.method == "POST":

```

```

        situationitem_form =
SituationitemForm(request.POST, instance = situationitem)
        if situationitem_form.is_valid():
            situationitem_form.save()
            situationitem_form.save_m2m()
            if not lost:
                return redirect("lostitemsituation",
number)
            else:
                return redirect("founditemsituation",
number)
        else:
            situationitem_form = SituationitemForm(None,
instance = situationitem)
        else:
            situationitem_form = None
            return render(request, "fiapp/edit_situationitem.html",
{"form": situationitem_form })

def publish_situationitem (request, lost = 0,
situation_slug):
    check = False
    if not lost:
        try:
            lostitemsituation =
Lostitemsituation.objects.get(slug = situation_slug)
            check =
lostitemsituation.accountaddress.account == request.user
        except Lostitemsituation.DoesNotExist:
            lostitemsituation = None
    else:
        try:
            founditemsituation =
Founditemsituation.objects.get(slug = situation_slug)
            check = founditemsituation.account ==
request.user
        except Lostitemsituation.DoesNotExist:
            founditemsituation = None
    if request.user.is_active and check:
        if request.method == "POST":
            situationitem_form =
SituationitemForm(request.POST)
            if situationitem.is_valid():
                situationitem =
situationitem_form.save(commit=False)
                if not lost:

```

```

        situationitem.lostitemsituation =
lostitemsituation
    else:
        situationitem.founditemsituation =
founditemsituation
        situationitem.save()
        situationitem.save_m2m()
        if not lost:
            return redirect("lostitemsituation",
lostitemsituation.id)
        else:
            return redirect("founditemsituation",
founditemsituation.id)
    else:
        situationitem_form = SituationitemForm()
    else:
        situationitem_form = None
    return render(request, "fiapp/edit_situationitem.html",
{"form": situationitem_form })

def cancel_lostitemsituation(request, situation_slug):
    try:
        lostitemsituation =
Lostitemsituation.objects.get(slug = situation_slug)
    except Lostitemsituation.DoesNotExist:
        lostitemsituation = None
    if lostitemsituation and
lostitemsituation.accountaddress.account == request.user:
        lostitemsituation.is_active = False
        lostitemsituation.save()
        return redirect("lostitemsituation",
lostitemsituation.slug)

def cancel_founditemsituation(request, situation_slug):
    try:
        founditemsituation =
Founditemsituation.objects.get(slug = situation_slug)
    except Founditemsituation.DoesNotExist:
        founditemsituation = None
    if founditemsituation and founditemsituation.account ==
request.user:
        founditemsituation.is_active = False
        founditemsituation.save()
        return redirect("founditemsituation",
founditemsituation.slug)

```

```

def publish_founditemsituation(request):
    account = request.user
    if account.is_active:
        if request.method == "POST":
            data_post = request.POST.copy()
            accountaddress = data_post["accountaddress"]
            founditemsituation_form =
FounditemsituationForm(request.POST)
            situationitem_form =
SituationitemForm(request.POST)
            if founditemsituation_form.is_valid() and
accountaddress.account = account and
situationitem_form.is_valid():
                founditemsituation =
founditemsituation_form.save(commit=False)
                founditemsituation.is_active = True
                founditemsituation.save()
                situationitem =
situationitem_form.save(commit=False)
                situationitem.founditemsituation =
founditemsituation
                situationitem.save()
                redirect("founditemsituation",
founditemsituation.slug)
            else:
                founditemsituation_form =
FounditemsituationForm()
                situationitem_form = SituationitemForm()
                return render(request, "fiapp/publish_found.html",
{"found_form": founditemsituation_form, "item_form":
situationitem_form })

```

```

def publish_lostitemsituation(request):
    account = request.user
    if account.is_active:
        if request.method == "POST":
            data_post = request.POST.copy()
            accountaddress = data_post["accountaddress"]
            lostitemsituation_form =
Lostitemsituation(request.POST)
            situationitem_form =
SituationitemForm(request.POST)
            if lostitemsituation_form.is_valid() and
accountaddress.account = account and
situationitem_form.is_valid():

```

```

        lostitemsituation =
lostitemsituation_form.save(commit=False)
        lostitemsituation.is_active = True
        lostitemsituation.save()
        situationitem =
situationitem_form.save(commit=False)
        situationitem.lostitemsituation =
lostitemsituation

        situationitem.save()
        redirect("lostitemsituation",
lostitemsituation.slug)
    else:
        lostitemsituation_form =
LostitemsituationForm()
        situationitem_form = SituationitemForm()
        return render(request, "fiapp/publish_lost.html",
{"lost_form": lostitemsituation_form, "item_form":
situationitem_form })

```

ДОДАТОК В
Слайди презентації

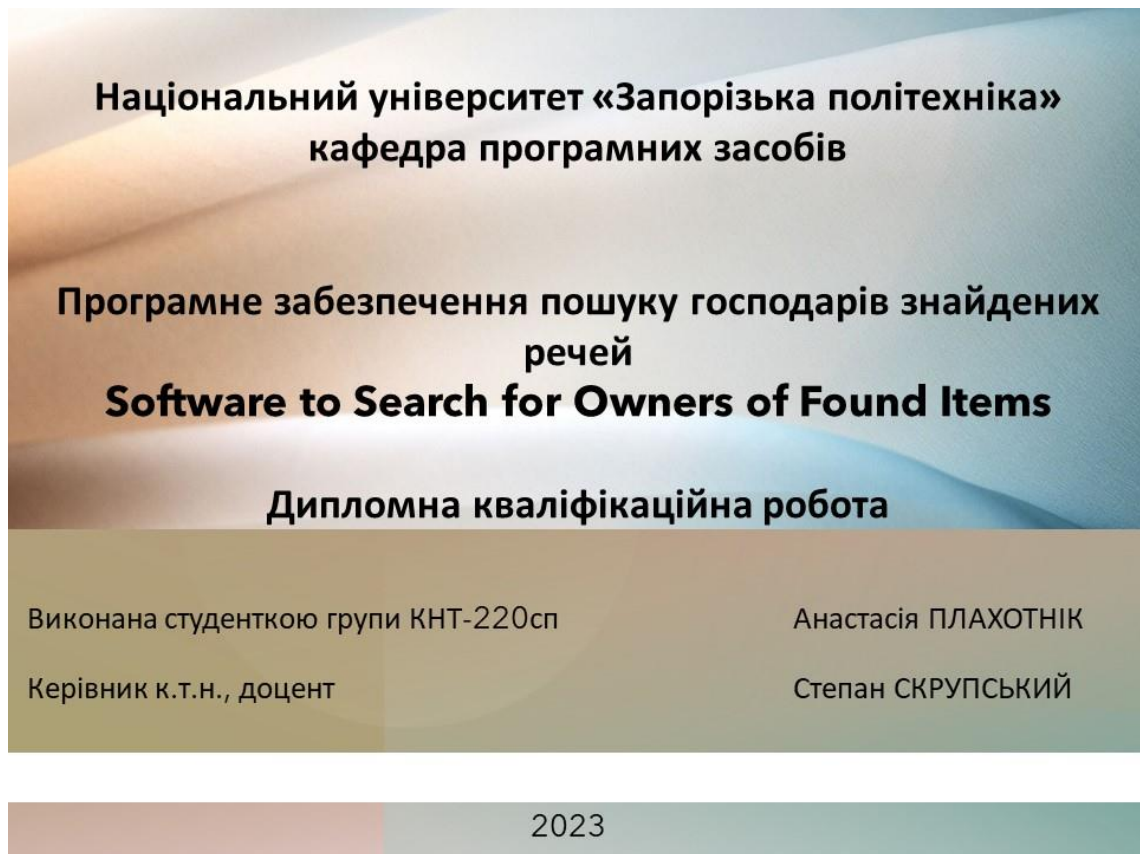


Рисунок В.1 – Титульний слайд

ОБ'ЄКТ, ПРЕДМЕТ ТА МЕТА РОБОТИ

- Об'єктом роботи є процес пошуку господарів знайдених речей і господарями втрачених речей.
- Предметом роботи є програмне забезпечення пошуку господарів знайдених речей.
- Мета роботи – розробити програмне забезпечення пошуку господарів знайдених речей для сприяння підвищенню результативності даного процесу.



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

ЗАВДАННЯ РОБОТИ

- аналіз програмних засобів пошуку господарів знайдених речей і визначення концепції програми;
- проектування програмного забезпечення пошуку господарів знайдених речей;
- реалізація вебдодатку пошуку господарів знайдених речей.

3

Рисунок В.3 – Завдання роботи

ПОРІВНЯННЯ ПРОГРАМНИХ ЗАСОБІВ ПОШУКУ ГОСПОДАРІВ ЗНАЙДЕНИХ РЕЧЕЙ

Характеристика	iLost	Lost and found networks	LostAndFound	FIAPP
Об'єднання декількох речей (оголошень) в одне оголошення	-	-	-	+
Пошук втрат і знахідок	-	+	+	+
Маршрути для знахідок у транспорті	-	-	-	+
Підтримка визначення доставки	+	-	-	+
Перевірка шахрайства	-	-	-	+

4

Рисунок В.4 – Порівняння програмних засобів пошуку господарів знайдених речей

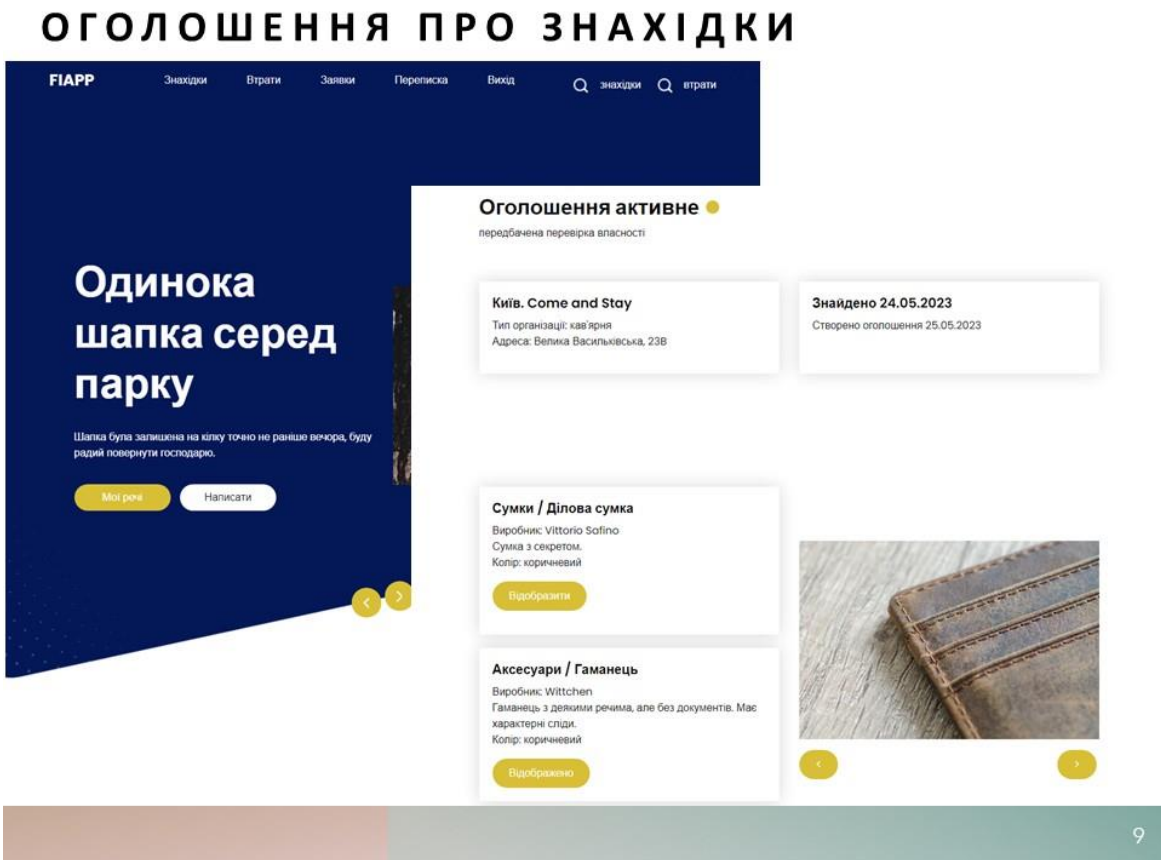


Рисунок В.9 – Оголошення про знахідки

ПОШУК СЕРЕД ЗНАХІДОК

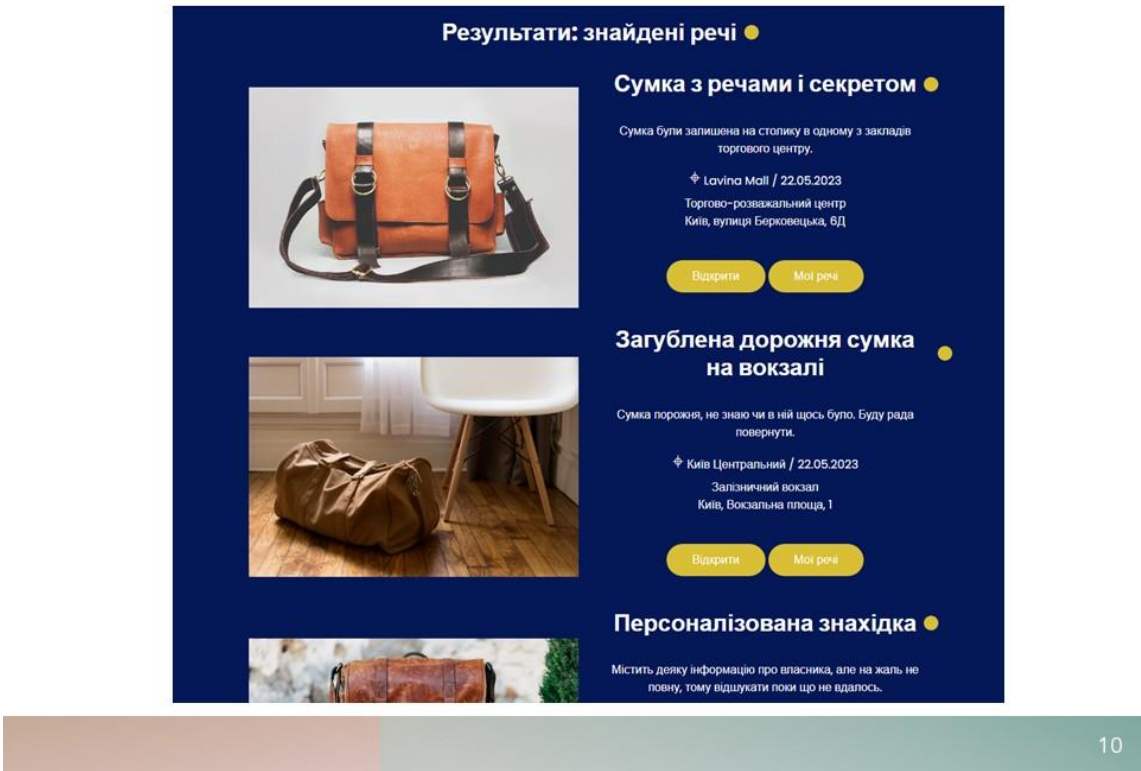


Рисунок В.10 – Пошук серед знахідок

ПУБЛІКАЦІЯ ОГолоШЕННЯ ПРО ВТРАТУ

Моя втрата

Назва

Опис

Організація

Введіть назву закладу (організації)

Дата випадку

дд.мм.рррр

Деталізуйте річ

Виробник

Модель

Категорія

Додати

Всі результати

11

Рисунок В.11 – Публікація оголошення про втрату

ВИСНОВКИ

- Виконано виділення програмних засобів, які зараз можуть використовуватися для пошуку господарів знайдених речей: iLost, Lost and found networks, LostAndFound. Проаналізовано наявні та відсутні у них можливості. Запропоновано функції програмному додатку, що було розроблено в підсумку, за результатами аналізу.
- Досягнуто виконання завдання проєктування програмного забезпечення пошуку господарів знайдених речей, включаючи вибір мови програмування Python, моделювання вимог до програми на основі діаграми варіантів використання, визначення структури роботи з даними на основі бази даних.
- Досягнуто виконання завдання реалізації вебдодатку пошуку господарів знайдених речей, в результаті чого описано порядок виконання і представлено функції, які реалізують всі функціональні вимоги до програми.
- Усі отримані результати призвели до досягнення мети розробити програмне забезпечення пошуку господарів знайдених речей для сприяння підвищенню результативності даного процесу.

12

Рисунок В.12 – Висновки