**Форма № 24**

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
**Національний університет «Запорізька політехніка»**

Факультет комп'ютерних наук і технологій
(повне найменування факультету)

Кафедра програмних засобів
(повне найменування кафедри )

# Пояснювальна записка

до дипломного проєкту (роботи)
бакалавр
(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ТУРИСТИЧНИХ ПОЇЗДОК
SOFTWARE FOR ORGANIZING GROUP TOURIST TRIPS

Виконав(ла): студент(ка) 4 курсу, групи КНТ-112
Спеціальності 121 Інженерія програмного
(код і найменування спеціальності)
забезпечення

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

БРАУН В.І.
(ПРІЗВИЩЕ та ініціали)
Керівник ЛЬОВКІН В.М.
(ПРІЗВИЩЕ та ініціали)
Рецензент ЗЕЛІК О.В.
(ПРІЗВИЩЕ та ініціали)

2026

**Форма № 25**

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

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

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри ПЗ, д.т.н, проф.**
**_____Сергій СУББОТІН**
“____” _____2026 року

# З А В Д А Н Н Я

## НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

БРАУНА Владислава Ігоровича
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення організації колективних туристичних поїздок. Software for Organizing Group Tourist Trips

керівник проєкту (роботи) к.т.н., доцент, ЛЬОВКІН Валерій Миколайович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “ 07 ” квітня 2026 року № 139
2. Строк подання студентом проєкту (роботи) 10 червня 2026 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. Проєктування програми. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

6. Консультанти розділів проєкту (роботи)

| Розділ | ПРІЗВИЩЕ, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | прийняв виконане завдання |
| 1-4 Основна частина | ЛЬОВКІН В.М., доцент | | |
| Нормоконтроль | БЄЛОВА А.В., асистент | | |

7. Дата видачі завдання « 20 » квітня 2026 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи) | Строк виконання етапів проєкту ( роботи ) | Примітка |
|---|---|---|---|
| 1 | Постановка завдання роботи. | 1 тиждень | Завдання, ТЗ |
| 2 | Аналіз предметної області. | 1 тиждень | Розділ 1 |
| 3 | Вибір мови програмування та інших технологій | 2 тиждень | Розділ 2 |
| | розробки. | | |
| 4 | Розробка архітектури програми. | 2 тиждень | Розділ 2 |
| 5 | Розробка програми. | 3-4 тижні | Розділ 3, 4 |
| 6 | Тестування та експериментальне дослідження | 5 тиждень | Розділ 4 |
| | програмного забезпечення. | | |
| 7 | Оформлення пояснювальної записки та документів | 6 тиждень | Додатки |
| | до неї. | | |
| 8 | Нормоконтроль та рецензування. | 7 тиждень | |
| 9 | Захист роботи. | 8 тиждень | |

**Студент(ка)** _____ Владислав БРАУН
( підпис ) (Ім'я ПРІЗВИЩЕ)

**Керівник проєкту (роботи)** _____ Валерій ЛЬОВКІН
( підпис ) (Ім'я ПРІЗВИЩЕ)

# РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 89 с., 3 табл., 27 рис., 3 дод., 10 джерел.

ТУРИСТИЧНА ПОЇЗДКА, ПЕРЕВІЗНИК, ПАМ’ЯТКА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ВЕБЗАСТОСУНОК.

Об’єкт роботи – процес розробки програмного забезпечення організації колективних туристичних поїздок. Предмет роботи – програмне забезпечення організації колективних туристичних поїздок. Мета роботи – зменшення витрат на організацію персональних подорожей за рахунок формування групи туристів на основі розробки програмного забезпечення організації колективних туристичних поїздок.

Матеріали, методи та технічні засоби: мова Python як основна структуруюча одиниця розробки загалом та фреймворк Django як основна структуруюча одиниця веброзробки, система MySQL для організації роботи з даними.

Результати. Програма створена з виділенням основних прийнятих рішень для реалізації програмного забезпечення організації колективних туристичних поїздок, включаючи етапи аналізу проблеми, проєктування і реалізації програми, та описом цих рішень у роботі.

Висновки. Програмне забезпечення організації колективних туристичних поїздок призначене для користувачів, які мають бажання відвідати музеї або пам’ятки в різних містах, але при цьому не мають власного транспорту або бажають скористатися перевагами відвідування у складі групи, та для перевізників, які готові забезпечити підтримку таких поїздок на основі маршрутів, сформованих самими туристами.

Галузь використання. Підтримка організації туристичних подорожей на індивідуальній основі. Програма призначена для її використання туристами, перевізниками.

# ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 3 tables, 27 figures, 3 appendixes, 10 sources.

TOURIST TRIP, TRANSPORTER, SIGHTSEEING, SOFTWARE, WEB APPLICATION.

The object of the work is the process of developing software for organizing group tourist trips. The subject of the work is software for organizing group tourist trips. The purpose of the work is to reduce the costs of organizing personal trips by forming a group of tourists based on software for organizing group tourist trips.

Materials, methods and technical means: Python language as the main structuring unit of development in general and Django framework as the main structuring unit of web development, MySQL system for organizing data work.

Results. The software was created with the allocation of the main decisions made for the implementation of software for organizing group tourist trips, including the stages of problem analysis, design and implementation of the software, and a description of these decisions in the work.

Conclusions. The software for organizing group tourist trips is intended for users who want to visit museums or attractions in different cities, but do not have their own transport or want to take advantage of the benefits of visiting as part of a group, and for carriers who are ready to provide support for such trips based on routes formed by tourists themselves.

Field of use. Support for organizing tourist trips on an individual basis. The program is intended for use by tourists, carriers.

# ЗМІСТ

С.

# ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

БД – база даних;

СКБД – система керування базами даних.

# ВСТУП

Колективні туристичні поїздки можуть бути організовані не тільки для існуючої групи, яка формується туристичною компанією, але і для колективу, який формується віддалено на основі використання програмного забезпечення. У такому випадку колективна туристична поїздка передбачає необхідність формування маршруту, доєднання до нового колективу, визначення перевізника. Саме це має бути забезпечено головним чином програмою організації колективних туристичних поїздок, створення якої передбачено в цій роботі.

Туристичні поїздки зазвичай організовуються або туристичними компаніями, або окремими організаторами, таким чином сприяючи шаблонності процесу. Відповідно є певний організатор, який бере на себе відповідальність за організацію всього процесу, а туристи тільки підтверджують свою участь в поїздці. Використання ж концепції, яку запропоновано в роботі, сприяє тому, щоб самі туристи стали більш активними учасниками самого процесу організації туристичної поїздки. Тоді вони можуть визначати маршрут, самі шукати перевізника, визначати дати поїздки. З іншого боку це їм надає можливості застосування переваг саме колективної поїздки. Адже саме колектив може знизити витрати на поїздку, але для цього план поїздки має бути узгоджений між учасниками, щоб таку перевагу фактично застосувати. Тому програма, яка розробляється, повинна враховувати це.

Мета роботи сформульована таким чином: зменшення витрат на організацію персональних подорожей за рахунок формування групи туристів на основі розробки програмного забезпечення організації колективних туристичних поїздок. Мета має бути досягнута через завдання:

– аналіз проблеми розробки програми організації колективних туристичних поїздок;

– проєктування програми організації колективних туристичних поїздок;

– розробка програми організації колективних туристичних поїздок.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Огляд програмних засобів організації колективних туристичних поїздок

Програма Howbout (рис. 1.1) дозволяє планувати колективні туристичні поїздки, але саме між друзями [1].

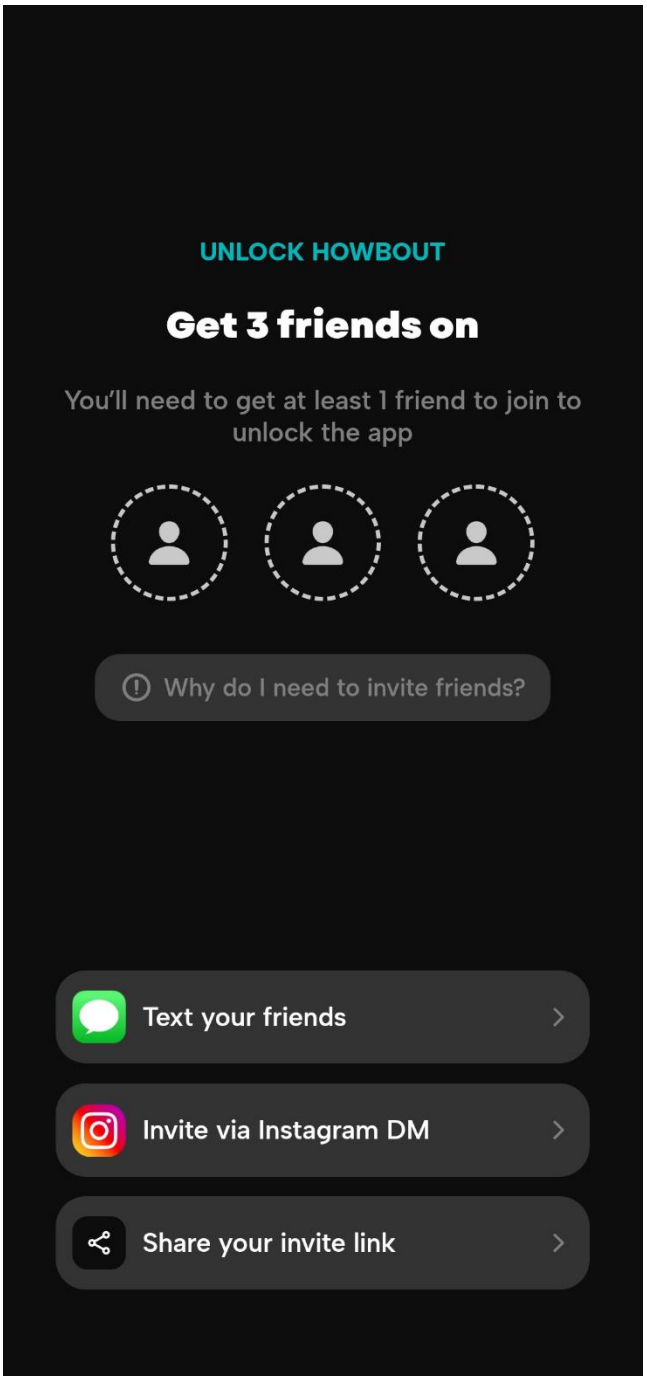


Рисунок 1.1 – Початок роботи з програмою Howbout

Якщо користувач не має таких друзів, тобто він фактично не має свого власного готового переліку учасників поїздки, то він не може навіть розпочати роботу з програмою, що звужує її використання або навіть унеможливлює у випадку, коли такого готового списку немає. Окрім того програма не надає можливості пошуку додаткових засобів, зокрема перевізників, які могли б підтримувати таку поїздку. Тобто фактично вона дозволяє створювати план поїздки та ділитись ним між власними друзями, але не організовувати саму колективну туристичну поїздку з незнайомими іншими туристами.

Таким чином, програма Howbout має своє призначення, проте в контексті організації колективних туристичних поїздок дуже обмежена або не може повноцінно використовуватись для цієї мети. До того ж програма може використовуватися тільки як застосунок для мобільних операційних систем. Тому отримати до неї доступ більш універсально, зокрема не вказуючи номер мобільного телефону, неможливо.

Програма Group Tour (рис. 1.2) також має те саме обмеження стосовно платформи, що і Howbout [2].

Group Tour дозволяє виконувати пошук можливих колективних туристичних поїздок, однак джерелом організації таких туристичних поїздок є туристичні агентства, що не зовсім відповідає меті цієї роботи. Тобто зниження витрат за рахунок наявності колективу досягається, проте самі туристичні поїздки не організовуються через програмне забезпечення, а лиш пропонуються готові варіанти, які підготовлені за заздалегідь готовим шаблоном. Програма є тільки агрегатором  таких пропозицій, через який зручно виконувати пошук, але цей пошук ведеться саме серед готових варіантів.

Використання програми безпосередньо з України на даний момент достатньо обмежено, оскільки підтримуваний номер телефону неможливо задати.

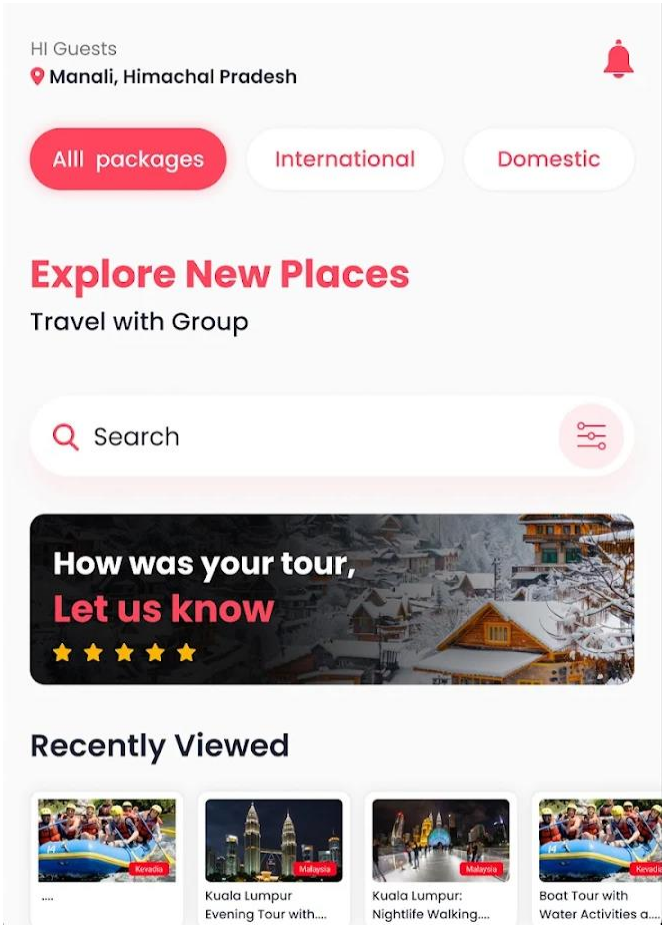


Рисунок 1.2 – Програма Group Tour

Окрім того було розглянуто ще ряд програмних засобів, які можуть використовуватися для задач, споріднених з колективними туристичними поїздками. Зокрема Splitwise, що є застосунком, який дозволяє розподіляти витрати між колективом поїздки [3]. Проте це рішення також не може бути застосовано для самого формування колективної туристичної поїздки.

## 1.2 Порівняльний аналіз програмного забезпечення організації колективних туристичних поїздок

Для порівняння програмних засобів організації колективних туристичних поїздок було виділено Howbout та Group Tour як аналоги застосунку Landmarktouringer, який розробляється в роботі (табл. 1.1).

Таблиця 1.1 – Порівняння програмних засобів організації колективних туристичних поїздок

| Критерій | Howbout | Group Tour | Landmark-touringer |
|---|---|---|---|
| Формування маршруту туристичної поїздки | + | – | + |
| Джерело організації туристичної поїздки | Знайомі, друзі | Агентство | Все коло бажаючих користувачів |
| Пошук перевізника для підтримки поїздки | – | – | + |
| Платформа | Мобільний застосунок | Мобільний застосунок | Вебрішення |
| Обов’язкова ідентифікація за номером телефону | + | + | – |

Landmarktouringer, розробка якого виконується в цій роботі, дозволяє формувати сам маршрут туристичної поїздки, який складається з музеїв та пам’яток, що доступно також і в Howbout. Проте формувати такий маршрут у Howbout можна тільки зі знайомими, за якими відомий номер телефону, тобто склад учасників має бути сталим та відомим ще до використання програми. У Landmarktouringer будь-який користувач за бажання може долучитись до поїздки. Окрім того аналоги не забезпечують пошук перевізника, можуть бути використані тільки на мобільній платформі та потребують обов’язкової ідентифікації користувача за номером телефону.

### 1.3 Висновки за розділом 1

Результати роботи над даним розділом включають:

– огляд програмних засобів організації колективних туристичних поїздок, серед яких головним чином виділено Howbout та Group Tour;

– порівняльний аналіз програмного забезпечення організації колективних туристичних поїздок, де виділено відмінності програми, яка розробляється та має назву Landmarktouringer, порівняно з аналогами.

# 2 ПРОЄКТУВАННЯ ПРОГРАМИ

## 2.1 Вибір засобів проєктування і розробки

Мова програмування Python є одним з базових рішень для створення вебзастосунків як через те, що сприяє прискоренню розробки, спрощенню написання коду, так і через ефективність у цьому процесі супутніх рішень для веброзробки. Такі рішення, фреймворки, були розглянуті детальніше під час порівняння FastAPI [4] та Django [5] для вибору того, який буде використовуватися в роботі (табл. 2.1).

Таблиця 2.1 – Порівняння фреймворків для розробки програми організації колективних туристичних поїздок

| Критерій | FastAPI | Django |
|---|---|---|
| Концепція фреймворку | Мікрофреймворк | Все головне вбудовано |
| Продуктивність | Дуже висока для мікросервісів | Висока для великих рішень |
| Розширюваність | Готових розширень значно менше | Багато підтримуваних розширень |
| Опис, документування, підтримка | Значно нижчі | Високі |

Для роботи обрано фреймворк Django, оскільки він серед вбудованих засобів має об’єктно-реляційне відображення, адміністративну панель і систему авторизації, на що не потрібно витрачати додатковий час для розробки або на пошук рішення для підключення, має високий ступінь опису, документування і підтримки, має багато готових розширень.

Роботу з даними, яка має забезпечуватися через використання бази даних (БД), вирішено реалізувати через систему керування базами даних (СКБД) MySQL [6].

## 2.2 Моделювання роботи з програмою організації колективних туристичних поїздок

Коло сценаріїв неавторизованого користувача включає в основному потоці:

– відтворення інформації про музей або пам'ятку;

– відтворення музеїв та пам'яток за містом;

– відтворення музеїв та пам'яток за країною;

– пошук серед музеїв та пам'яток;

– авторизація туристів;

– авторизація перевізників;

– авторизація менеджерів;

– відтворення рейтингувань музею або пам'ятки (рис. 2.1).

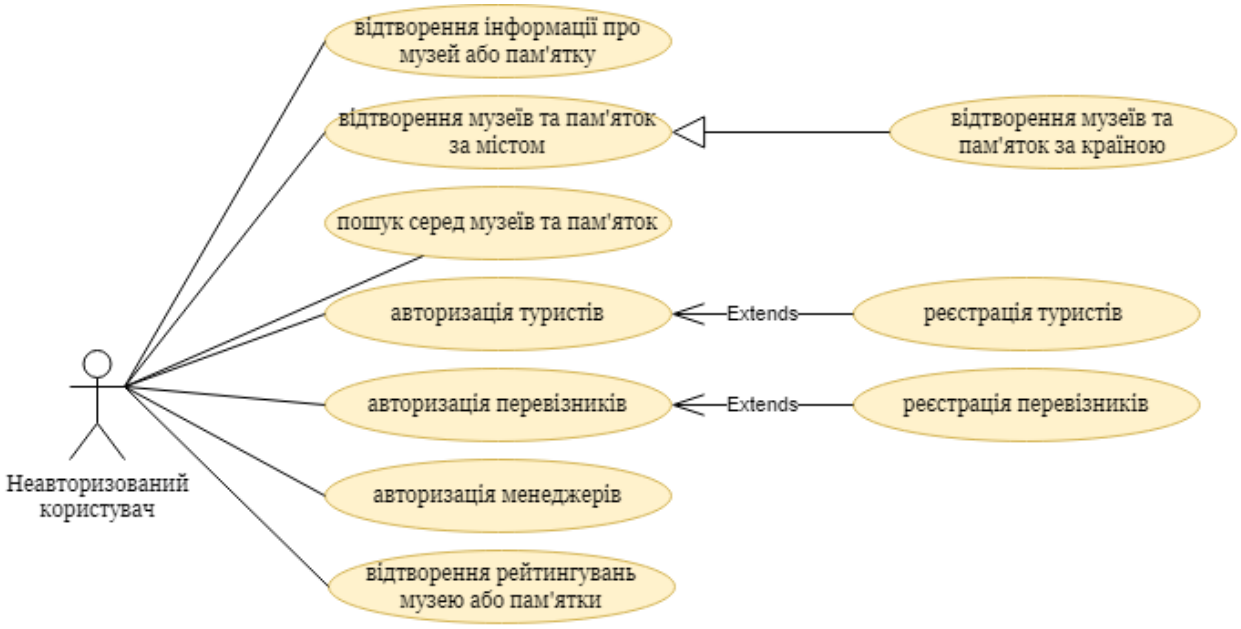


Рисунок 2.1 – Діаграма прецедентів неавторизованого користувача програми організації колективних туристичних поїздок

Коло сценаріїв неавторизованого користувача включає в альтернативному потоці:

– реєстрація туристів альтернативно до авторизації туристів, коли

турист не має акаунту;

– реєстрація перевізників до авторизації перевізників, коли перевізник не має акаунту.

Коло сценаріїв туриста додатково до неавторизованого користувача включає в основному потоці:

– закладання музею або пам'ятки;

– відтворення закладених музеїв та пам'яток;

– висловлення бажання відвідування музею або пам'ятки;

– відтворення бажаних для відвідування користувачем музеїв або пам'яток;

– формування пропозиції для відвідування музею або пам'ятки;

– відтворення власних пропозицій для відвідування музею або пам'ятки;

– відтворення задіяних пропозицій для відвідування музею або пам'ятки;

– приєднання до існуючої пропозиції відвідування музею або пам'ятки;

– визначення параметрів додавання музеїв або пам'яток до пропозиції;

– додавання музею або пам'ятки до існуючої пропозиції;

– голосування за музей або пам'ятку в існуючій пропозиції;

– адресування пропозиції перевізнику за колективною туристичною поїздкою;

– відтворення інформації пропозиції для відвідування музею або пам'ятки;

– рейтингування перевізника за результатами колективної туристичної поїздки;

– рейтингування музею або пам'ятки;

– відтворення своїх рейтингувань перевізників;

– відтворення запланованих відвідувань музеїв або пам'яток;

– відтворення рейтингувань перевізника;

– відтворення інформації про перевізника;

– відтворення даних туриста;

– відтворення своїх рейтингувань музеїв або пам'яток (рис. 2.2).

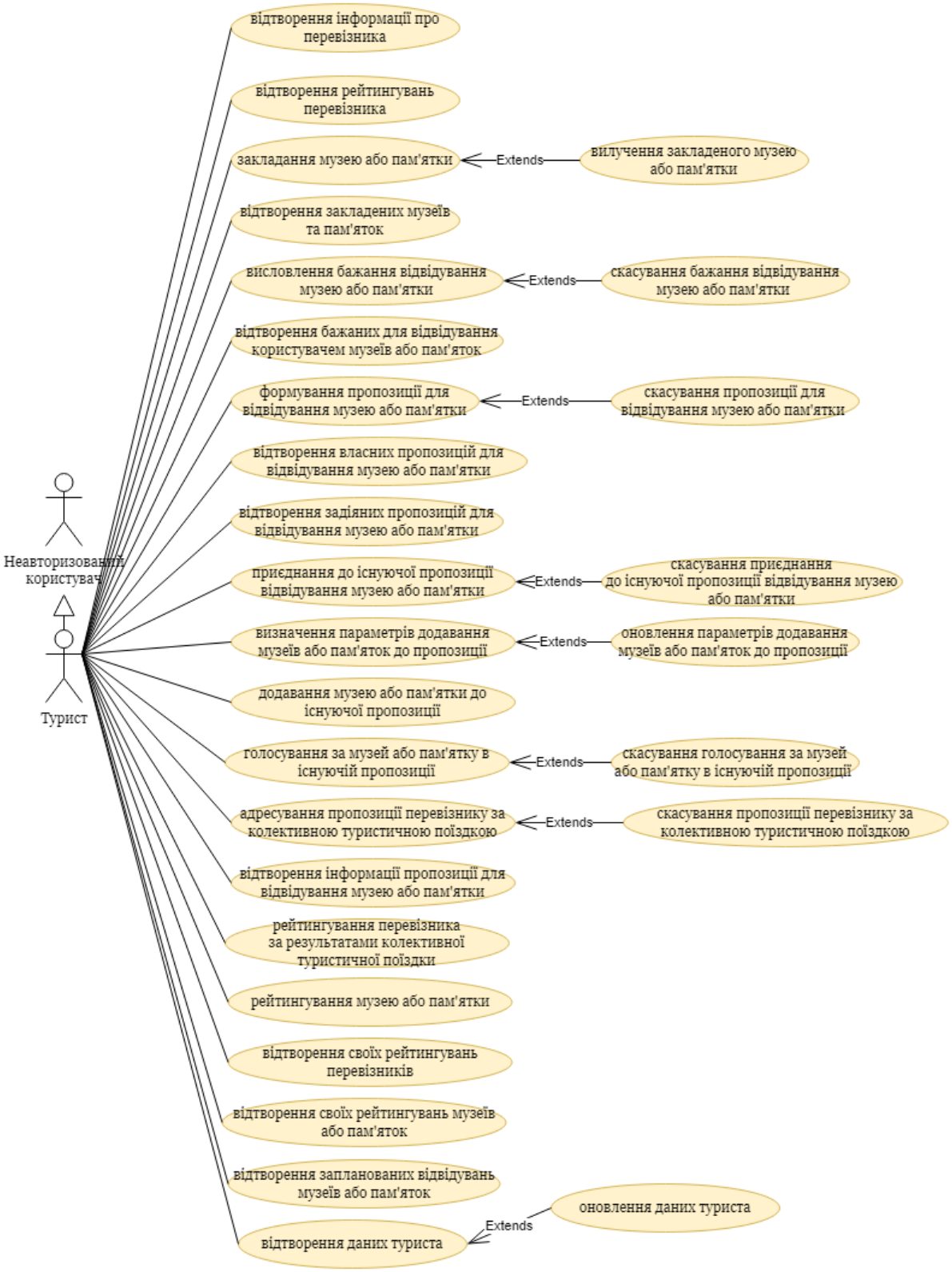


Рисунок 2.2 – Діаграма прецедентів туриста в програмі організації колективних туристичних поїздок

Коло сценаріїв туриста включає в альтернативному потоці:

– оновлення даних туриста альтернативно до відтворення даних туриста, коли дані мають бути змінені самим туристом через їх неактуальність;

– вилучення закладеного музею або пам'ятки альтернативно до закладання музею або пам'ятки, коли закладення музею або пам’ятки вже реалізоване, але більше в тому немає потреби;

– скасування бажання відвідування музею або пам'ятки альтернативно до висловлення бажання відвідування музею або пам'ятки, коли бажання відвідування музею або пам'ятки висловлено, але більше воно не відповідає дійсності;

– скасування пропозиції для відвідування музею або пам'ятки альтернативно до формування пропозиції для відвідування музею або пам'ятки, коли пропозиція для відвідування музею або пам’ятки була створена автором помилково;

– скасування приєднання до існуючої пропозиції відвідування музею або пам'ятки альтернативно до приєднання до існуючої пропозиції відвідування музею або пам'ятки, коли приєднання до існуючої пропозиції відвідування музею або пам'ятки було виконано помилково;

– оновлення параметрів додавання музеїв або пам'яток до пропозиції альтернативно до визначення параметрів додавання музеїв або пам'яток до пропозиції, коли параметри додавання музеїв або пам'яток до пропозиції автором встановлені, але вже не є актуальними;

– скасування голосування за музей або пам'ятку в існуючій пропозиції альтернативно до голосування за музей або пам'ятку в існуючій пропозиції, коли голосування за музей або пам'ятку в існуючій пропозиції вже не відповідає інтересам туриста, який віддав цей голос;

– скасування пропозиції перевізнику за колективною туристичною поїздкою альтернативно до адресування пропозиції перевізнику за

колективною туристичною поїздкою, коли така пропозиція перевізнику за колективною туристичною поїздкою була створена туристом помилково.

Коло сценаріїв перевізника включає додатково до неавторизованого користувача в основному потоці:

– заповнення даних перевізника;

– відтворення інформації пропозиції для відвідування музею або пам'ятки;

– відтворення непідтверджених пропозицій перевізнику за колективними туристичними поїздками;

– відтворення скасованих пропозицій перевізнику за колективними туристичними поїздками;

– відтворення підтверджених пропозицій перевізнику за колективними туристичними поїздками;

– відтворення рейтингувань перевізника;

– відтворення інформації про перевізника;

– підтвердження пропозиції перевізнику за колективною туристичною поїздкою (рис. 2.3).

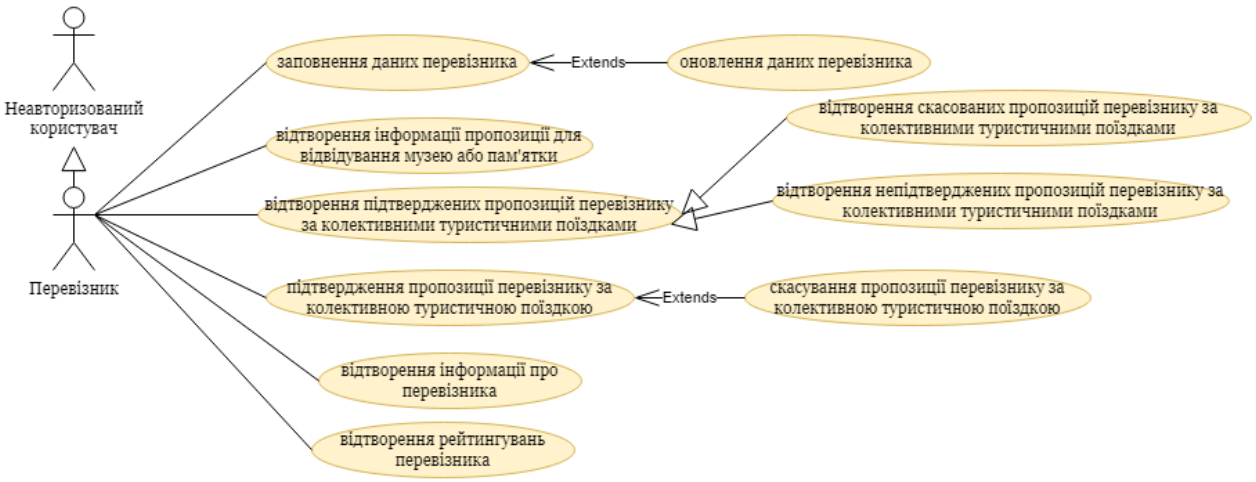


Рисунок 2.3 – Діаграма прецедентів перевізника в програмі організації колективних туристичних поїздок

Коло сценаріїв перевізника включає в альтернативному потоці:

– оновлення даних перевізника альтернативно до заповнення даних перевізника, коли дані, внесені перевізником, вже не актуальні;

– скасування пропозиції перевізнику за колективною туристичною поїздкою альтернативно до підтвердження пропозиції перевізнику за колективною туристичною поїздкою, коли перевізник не бажає виконувати таке перевезення.

Коло сценаріїв менеджера включає в основному потоці додатково до неавторизованого користувача додавання екземпляру музею або пам'ятки.

Коло сценаріїв неавторизованого користувача включає в альтернативному потоці:

– оновлення екземпляру музею або пам'ятки альтернативно до додавання екземпляру музею або пам'ятки, коли такий екземпляр музею або пам'ятки містить некоректні дані;

– вилучення екземпляру музею або пам'ятки альтернативно до додавання екземпляру музею або пам'ятки, коли такий екземпляр музею або пам'ятки був доданий помилково (рис. 2.4).

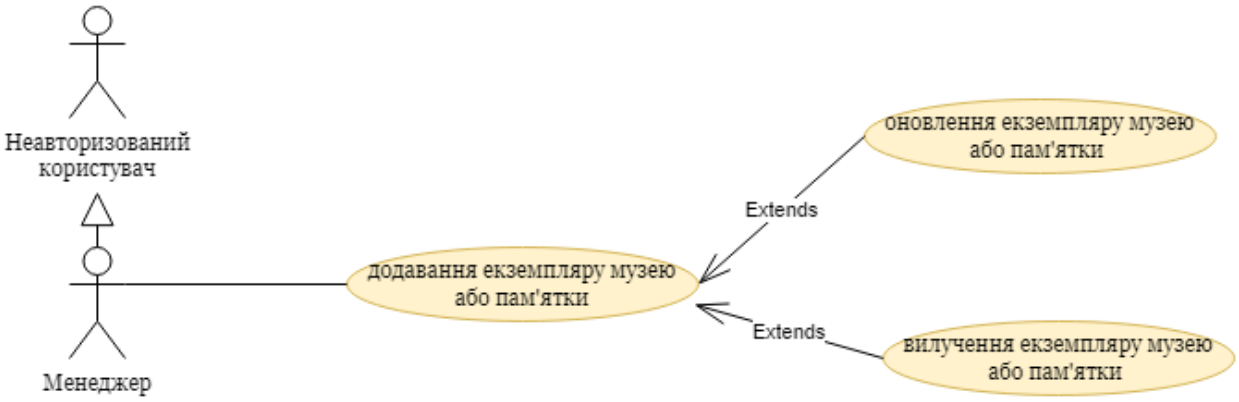


Рисунок 2.4 – Діаграма прецедентів менеджера в програмі організації колективних туристичних поїздок

## 2.3 Визначення структури бази даних програми організації колективних туристичних поїздок

Робота з БД підтримується через використання СКБД MySQL.

У БД таблиця Touristcountry представляє країни, в яких можливі колективні туристичні поїздки через програму, через поля:

– id – ідентифікатор країни, в яку можлива колективна туристична поїздка;

– countryoffname – назва країни, в яку можлива колективна туристична поїздка.

У БД таблиця Touristcity представляє міста, в яких можливі колективні туристичні поїздки через програму, через поля:

– id – ідентифікатор міста, в яке можлива колективна туристична поїздка;

– touristcountry_id – країна, де знаходиться можливе для колективної туристичної поїздки місто: зв’язок з таблицею Touristcountry через зовнішній ключ (один-до-багатьох);

– cityoffname – назва міста, в яке можлива колективна туристична поїздка.

У БД таблиця Landmarkcategory представляє категорії туристичних пам’яток, доступних для колективних туристичних поїздок, через поля:

– id – ідентифікатор категорії туристичних пам’яток, доступних для колективних туристичних поїздок;

– landmarkcategname – назва категорії туристичних пам’яток, доступних для колективних туристичних поїздок.

У БД таблиця Touristlandmark представляє музеї та пам’ятки, доступні для колективних туристичних поїздок, через поля:

– id – ідентифікатор музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarktitle – назва музею або пам’ятки, доступної для колективних туристичних поїздок;

– touristcity_id – місто, в якому знаходиться музей або пам’ятка, доступна для колективної туристичної поїздки: зв’язок з таблицею Touristcity через зовнішній ключ (один-до-багатьох);

– landmarklat – широта положення музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarklong – довгота положення музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarkaddress – адреса музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarksite – вебадреса музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarkcategory_id – категорія пам’ятки (або музей), доступної для колективних туристичних поїздок: зв’язок з таблицею Landmarkcategory через зовнішній ключ (один-до-багатьох);

– landmarkinfo – текстове представлення музею або пам’ятки, доступної для колективних туристичних поїздок;

– landmarkrating – рейтинг музею або пам’ятки, доступної для колективних туристичних поїздок, виставлений відвідувачами за результатами поїздок.

У БД таблиця Touristlandmarkphoto представляє фотографії музею або пам’ятки, доступної для колективної туристичної поїздки, через поля:

– id – ідентифікатор фотографії музею або пам’ятки, доступної для колективної туристичної поїздки;

– touristlandmark_id – музей або пам’ятка, доступна для колективної туристичної поїздки: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– photofile – файл фотографії музею або пам’ятки, доступної для колективної туристичної поїздки;

– photolog – коментар до фотографії музею або пам’ятки, доступної для колективної туристичної поїздки;

– basephoto – позначка про використання фотографії музею або пам’ятки, доступної для колективної туристичної поїздки, в якості головного зображення.

У БД таблиця Tourist представляє акаунт туриста через поля:

– id – ідентифікатор акаунта туриста;

– djappuser_id – акаунт туриста в Django: зв’язок з таблицею User через зовнішній ключ (один-до-одного);

– touristphoto – фотографія туриста;

– touristinfo – текстова інформація туриста;

– touristcity_id – місто, в якому проживає турист, що використовується як базова точка для орієнтації під час початку туристичної подорожі: зв’язок з таблицею Touristcity через зовнішній ключ (один-до-багатьох).

У БД таблиця Touristlandmarkbook представляє закладання музею або пам’ятки через поля:

– id – ідентифікатор закладання музею або пам’ятки туристом;

– touristlandmark_id – музей або пам’ятка, яку закладено туристом: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який зазначив закладку на цьому музеї або пам’ятці: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох).

У БД таблиця Touristlandmarkwish представляє бажання відвідування музею або пам'яткичерез поля:

– id – ідентифікатор бажання відвідування музею або пам'ятки туристом;

– touristlandmark_id – музей або пам’ятка, яку бажає відвідати турист: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який бажає відвідати цей музей або пам’ятку: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох).

У БД таблиця Touristlandmarkproposition представляє пропозицію для відвідування музею або пам'ятки у складі колективної туристичної поїздки через поля:

– id – ідентифікатор пропозиції для відвідування музею або пам'ятки у складі колективної туристичної поїздки;

– tourist_id – турист, який створив таку пропозицію для відвідування музею або пам'ятки у складі колективної туристичної поїздки: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох);

– touristlandmark_id – музей або пам’ятка, пропозицію відвідування якої створив турист у складі колективної туристичної поїздки: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– touristvisittimestart – час початку перебування в самому музеї або пам’ятці за пропозицією відвідування музею або пам’ятки у складі колективної туристичної поїздки;

– touristvisittimeend – час завершення перебування в самому музеї або пам’ятці за пропозицією відвідування музею або пам’ятки у складі колективної туристичної поїздки;

– touristlandmarkminvotes – мінімальна кількість голосів, яку необхідно отримати пропозиції відвідування музею або пам’ятки у складі колективної туристичної поїздки для можливості адресування її перевізнику або додавання інших музеїв або пам’яток;

– minvisitinterval – мінімальний час, який може тривати відвідування музею або пам’ятки, які додатково внесені до пропозиції;

– maxvisitinterval – максимальний час, який може тривати відвідування музею або пам’ятки, які додатково внесені до пропозиції;

– touristvisitfeature – коментар щодо особливостей відвідування музею або пам’ятки у складі колективної туристичної поїздки;

– touristpropositionstate – стан пропозиції для відвідування музею або пам'ятки у складі колективної туристичної поїздки.

У БД таблиця Touristlandmarkpropositionguest представляє приєднання туристів до існуючої пропозиції відвідування музею або пам'ятки через поля:

– id – ідентифікатор приєднання туриста до існуючої пропозиції відвідування музею або пам'ятки;

– touristlandmarkproposition_id – пропозиція для відвідування музею або пам'ятки у складі колективної туристичної поїздки, до якої приєднався турист: зв’язок з таблицею Touristlandmarkproposition через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який приєднався до існуючої пропозиції відвідування музею або пам'ятки: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох).

У БД таблиця Touristlandmarkpropositionpoint представляє додані музеї або пам'ятки до існуючої пропозиції відвідування музею або пам'ятки через поля:

– id – ідентифікатор доданого музею або пам'ятки до існуючої пропозиції відвідування музею або пам'ятки;

– touristlandmarkproposition_id – існуюча пропозиція відвідування музею або пам'ятки, до якої додано музей або пам’ятку: зв’язок з таблицею Touristlandmarkproposition через зовнішній ключ (один-до-багатьох);

– touristlandmark_id – музей або пам’ятка, додана до існуючої пропозиції відвідування музею або пам'ятки: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– touristvisittimestart – час пропонованого початку відвідування конкретно цього музею або пам'ятки, доданого до існуючої пропозиції відвідування музею або пам'ятки;

– touristvisittimeend – час пропонованого завершення відвідування конкретно цього музею або пам'ятки, доданого до існуючої пропозиції відвідування музею або пам'ятки;

– touristpointstate – стан додавання музею або пам’ятки до існуючої пропозиції відвідування музею або пам'ятки.

У БД таблиця Touristlandmarkpropositionpointvote представляє голосування за доданий музей або пам'ятку до існуючої пропозиції відвідування музею або пам'ятки через поля:

– id – ідентифікатор голосування за доданий музей або пам'ятку до існуючої пропозиції відвідування музею або пам'ятки;

– touristlandmarkpropositionpoint_id – доданий музей або пам'ятка до існуючої пропозиції відвідування музею або пам'ятки, за який віддано голос: зв’язок з таблицею Touristlandmarkpropositionpoint через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який віддав свій голос за додавання цього музею або пам’ятки до існуючої пропозиції відвідування музею або пам'ятки: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох).

У БД таблиця Transporter представляє перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, через поля:

– id – ідентифікатор перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки;

– djappuser_id – акаунт перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, в Django: зв’язок з таблицею User через зовнішній ключ (один-до-одного);

– transporterphoto – фотографія для графічного представлення перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки;

– transporterinfo – текстове представлення перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки;

– transportercity_id – місто, з якого веде свою діяльність перевізник, який може підтримувати колективну поїздку для відвідування музею або пам’ятки: зв’язок з таблицею Touristcity через зовнішній ключ (один-до-багатьох);

– transporterradius – радіус в кілометрах, на якому перевізник готовий підтримувати колективну поїздку для відвідування музею або пам’ятки, відносно міста, з якого він веде свою діяльність;

– transporterrating – рейтинг перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки (рис. 2.5).

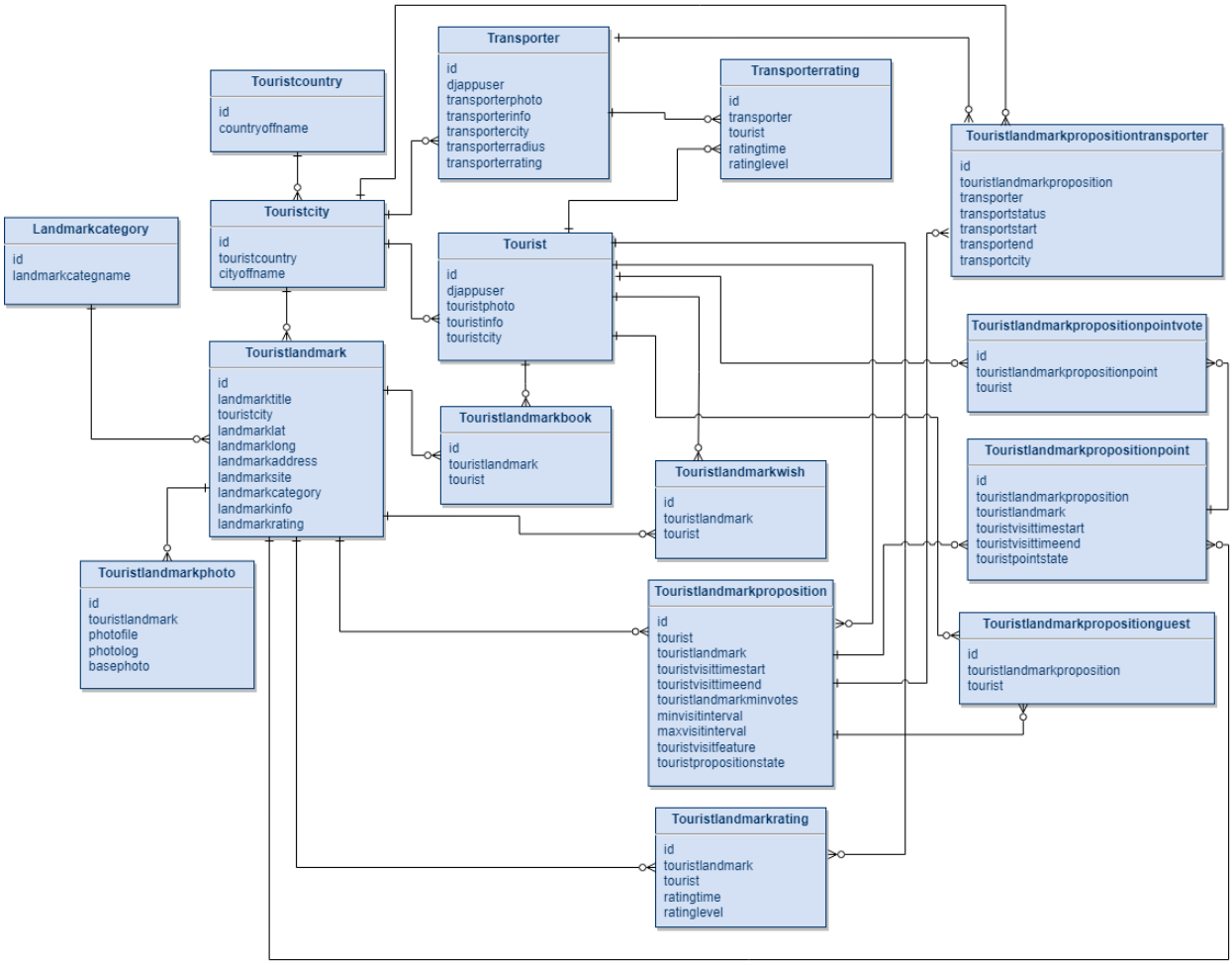


Рисунок 2.5 – Схема БД програми організації колективних туристичних поїздок

У БД таблиця Transporterrating представляє рейтингування перевізника через поля:

– id – ідентифікатор рейтингування перевізника;

– transporter_id – перевізник, який отримав цей рейтинг: зв’язок з таблицею Transporter через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який виставив цей рейтинг: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох);

– ratingtime – час рейтингування перевізника;

– ratinglevel – рейтингування перевізника в балах.

У БД таблиця Touristlandmarkrating представляє рейтингування музею або пам’ятки через поля:

– id – ідентифікатор рейтингування музею або пам’ятки;

– touristlandmark_id – музей або пам’ятка, рейтингування якої виконано: зв’язок з таблицею Touristlandmark через зовнішній ключ (один-до-багатьох);

– tourist_id – турист, який виставив цей рейтинг: зв’язок з таблицею Tourist через зовнішній ключ (один-до-багатьох);

– ratingtime – час рейтингування музею або пам’ятки;

– ratinglevel – рейтингування музею або пам’ятки в балах.

У БД таблиця Touristlandmarkpropositiontransporter представляє пропозицію перевізнику за колективною туристичною поїздкою через поля:

– id – ідентифікатор пропозиції перевізнику за колективною туристичною поїздкою;

– touristlandmarkproposition_id – існуюча пропозиція відвідування музею або пам'ятки, за якою сформована ця пропозиція перевізнику: зв’язок з таблицею Touristlandmarkproposition через зовнішній ключ (один-до-багатьох);

– transporter_id – перевізник, якому сформовано пропозицію за колективною туристичною поїздкою: зв’язок з таблицею Transporter через зовнішній ключ (один-до-багатьох);

– transportstatus – стан пропозиції перевізнику за колективною туристичною поїздкою;

– transportcity_id – місто, з якого має бути розпочато колективну туристичну поїздку за цією пропозицією за колективною туристичною

поїздкою: зв’язок з таблицею Touristcity через зовнішній ключ (один-до-багатьох);

– transportstart – час виїзду з початкового міста за цією пропозицією перевізнику за колективною туристичною поїздкою;

– transportend – час повернення до початкового міста за цією пропозицією перевізнику за колективною туристичною поїздкою.

## 2.4 Висновки за розділом 2

Результати роботи над даним розділом включають:

– порівняння альтернатив та вибір мови Python як основної структуруючої одиниці розробки загалом та фреймворку Django як основної структуруючої одиниці веброзробки, системи MySQL для організації роботи з даними;

– моделювання роботи з програмою організації колективних туристичних поїздок для менеджера, туриста, неавторизованого користувача, перевізника на основі діаграм прецедентів;

– визначення структури БД програми організації колективних туристичних поїздок з її детальним описом.

# 3 РОЗРОБКА ПРОГРАМИ

## 3.1 Послідовність роботи з програмою організації колективних туристичних поїздок

Програма організації колективних туристичних поїздок впорядкована таким чином, що з нею можна розпочати роботу з вибору одного з режимів, доступних всім користувачам, вибираючи відповідну адресу або пункт меню, а далі розширюючи набір варіантів зокрема через авторизацію (рис. 3.1).

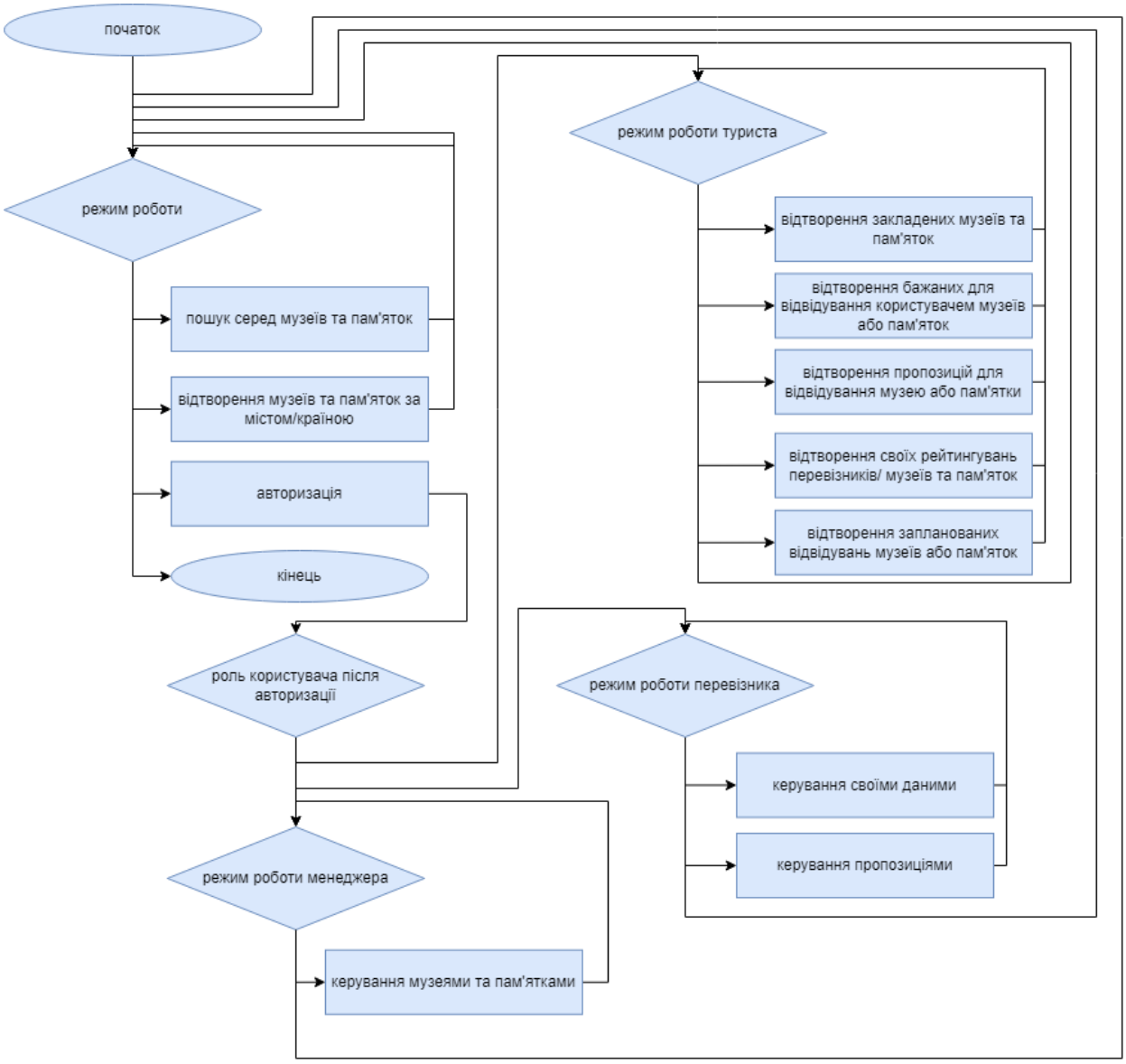


Рисунок 3.1 – Схема функціонування програми організації колективних туристичних поїздок

## 3.2 Рішення з реалізації програми організації колективних туристичних поїздок

Опис реалізації програми організації колективних туристичних поїздок, розробленої у відповідності зі структурою Django-застосунку [7], розпочато з опису створених класів моделей програми [8] для взаємодії з СКБД MySQL [6].

Клас Touristcountry представляє модель країн, в яких можливі колективні туристичні поїздки через програму.

Поле countryoffname реалізує назву країни, в яку можлива колективна туристична поїздка, на основі класу CharField.

Клас Touristcity представляє модель міст, в яких можливі колективні туристичні поїздки через програму.

Поле touristcountry реалізує країну, де знаходиться можливе для колективної туристичної поїздки місто, на основі класу ForeignKey зі зв’язком з моделлю Touristcountry.

Поле cityoffname реалізує назву міста, в яке можлива колективна туристична поїздка, на основі класу CharField.

Клас Landmarkcategory представляє модель категорії туристичних пам’яток, доступних для колективних туристичних поїздок.

Поле landmarkcategname реалізує назву категорії туристичних пам’яток, доступних для колективних туристичних поїздок на основі класу CharField.

Клас Touristlandmark представляє модель музеїв та пам’яток, доступних для колективних туристичних поїздок.

Поле landmarktitle реалізує назву музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу CharField.

Поле touristcity реалізує місто, в якому знаходиться музей або пам’ятка, доступна для колективної туристичної поїздки, на основі класу ForeignKey зі зв’язком з моделлю Touristcity.

Поле landmarklat реалізує широту положення музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу DecimalField.

Поле landmarklong реалізує довготу положення музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу DecimalField.

Поле landmarkaddress реалізує адресу музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу CharField.

Поле landmarksite реалізує вебадресу музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу URLField.

Поле landmarkcategory реалізує категорію пам’ятки (або музей), доступної для колективних туристичних поїздок, на основі класу ForeignKey зі зв’язком з моделлю Landmarkcategory.

Поле landmarkinfo реалізує текстове представлення музею або пам’ятки, доступної для колективних туристичних поїздок, на основі класу CharField.

Поле landmarkrating реалізує рейтинг музею або пам’ятки, доступної для колективних туристичних поїздок, виставлений відвідувачами за результатами поїздок, на основі класу PositiveSmallIntegerField.

Клас Touristlandmarkphoto представляє модель фотографії музею або пам’ятки, доступної для колективної туристичної поїздки.

Поле touristlandmark реалізує музей або пам’ятку, доступну для колективної туристичної поїздки, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmark.

Поле photofile реалізує файл фотографії музею або пам’ятки, доступної для колективної туристичної поїздки, на основі класу ImageField.

Поле photolog реалізує коментар до фотографії музею або пам’ятки, доступної для колективної туристичної поїздки, на основі класу CharField.

Поле basephoto реалізує позначку про використання фотографії музею або пам’ятки, доступної для колективної туристичної поїздки, в якості головного зображення, на основі класу BooleanField.

Клас Tourist представляє модель акаунта туриста.

Поле djappuser реалізує акаунт туриста в Django на основі класу OneToOneField зі зв’язком з моделлю User.

Поле touristphoto реалізує фотографію туриста на основі класу ImageField.

Поле touristinfo реалізує текстову інформацію туриста на основі класу CharField.

Поле touristcity реалізує місто, в якому проживає турист, що використовується як базова точка для орієнтації під час початку туристичної подорожі, на основі класу ForeignKey зі зв’язком з моделлю Touristcity.

Поле touristlandmarkbooks реалізує закладання музеїв або пам’яток, виконані туристом, на основі класу ManyToManyField зі зв’язком з моделлю Touristlandmark.

Поле touristlandmarkwishes реалізує бажання відвідування музею або пам'ятки, висловлені туристом, на основі класу ManyToManyField зі зв’язком з моделлю Touristlandmark.

Клас Touristlandmarkproposition представляє модель пропозиції для відвідування музею або пам'ятки у складі колективної туристичної поїздки.

Поле tourist реалізує туриста, який створив таку пропозицію для відвідування музею або пам'ятки у складі колективної туристичної поїздки, на основі класу ForeignKey зі зв’язком з моделлю Tourist.

Поле touristlandmark реалізує музей або пам’ятку, пропозицію відвідування якої створив турист у складі колективної туристичної поїздки, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmark.

Поле touristvisittimestart реалізує час початку перебування в самому музеї або пам’ятці за пропозицією відвідування музею або пам’ятки у складі колективної туристичної поїздки на основі класу DateTimeField.

Поле touristvisittimeend реалізує час завершення перебування в самому музеї або пам’ятці за пропозицією відвідування музею або пам’ятки у складі колективної туристичної поїздки на основі класу DateTimeField.

Поле touristlandmarkminvotes реалізує мінімальну кількість голосів, яку необхідно отримати пропозиції відвідування музею або пам’ятки у складі колективної туристичної поїздки для можливості адресування її перевізнику або додавання інших музеїв або пам’яток, на основі класу PositiveSmallIntegerField.

Поле minvisitinterval реалізує мінімальний час, який може тривати відвідування музею або пам’ятки, які додатково внесені до пропозиції, на основі класу TimeField.

Поле maxvisitinterval реалізує максимальний час, який може тривати відвідування музею або пам’ятки, які додатково внесені до пропозиції, на основі класу TimeField.

Поле touristvisitfeature реалізує коментар щодо особливостей відвідування музею або пам’ятки у складі колективної туристичної поїздки на основі класу CharField.

Поле touristpropositionstate реалізує стан пропозиції для відвідування музею або пам'ятки у складі колективної туристичної поїздки на основі класу CharField.

Поле touristlandmarkpropositionguests реалізує приєднання туристів до існуючої пропозиції відвідування музею або пам'ятки на основі класу ForeignKey зі зв’язком з моделлю Tourist.

Клас Touristlandmarkpropositionpoint представляє модель доданого музею або пам'ятки до існуючої пропозиції відвідування музею або пам'ятки.

Поле touristlandmarkproposition реалізує існуючу пропозицію відвідування музею або пам'ятки, до якої додано музей або пам’ятку, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmarkproposition.

Поле touristlandmark реалізує музей або пам’ятку, додану до існуючої пропозиції відвідування музею або пам'ятки, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmark.

Поле touristvisittimestart реалізує час пропонованого початку відвідування конкретно цього музею або пам'ятки, доданого до існуючої пропозиції відвідування музею або пам'ятки, на основі класу DateTimeField.

Поле touristvisittimeend реалізує час пропонованого завершення відвідування конкретно цього музею або пам'ятки, доданого до існуючої пропозиції відвідування музею або пам'ятки, на основі класу DateTimeField.

Поле touristpointstate реалізує стан додавання музею або пам’ятки до існуючої пропозиції відвідування музею або пам'ятки, на основі класу CharField.

Поле touristlandmarkpropositionpointvotes реалізує голосування за доданий музей або пам'ятку до існуючої пропозиції відвідування музею або пам'ятки на основі класу ManyToManyField зі зв’язком з моделлю Tourist.

Клас Transporter представляє модель перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки.

Поле djappuser реалізує акаунт перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, в Django на основі класу OneToOneField зі зв’язком з моделлю User.

Поле transporterphoto реалізує фотографію для графічного представлення перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, на основі класу ImageField.

Поле transporterinfo реалізує текстове представлення перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, на основі класу CharField.

Поле transportercity реалізує місто, з якого веде свою діяльність перевізник, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, на основі класу ForeignKey зі зв’язком з моделлю Touristcity.

Поле transporterradius реалізує радіус в кілометрах, на якому перевізник готовий підтримувати колективну поїздку для відвідування музею

або пам’ятки, відносно міста, з якого він веде свою діяльність, на основі класу DecimalField.

Поле transporterrating реалізує рейтинг перевізника, який може підтримувати колективну поїздку для відвідування музею або пам’ятки, на основі класу DecimalField.

Клас Transporterrating представляє модель рейтингування перевізника.

Поле transporter реалізує перевізника, який отримав цей рейтинг, на основі класу ForeignKey зі зв’язком з моделлю Transporter.

Поле tourist реалізує туриста, який виставив цей рейтинг, на основі класу ForeignKey зі зв’язком з моделлю Tourist.

Поле ratingtime реалізує час рейтингування перевізника на основі класу DateTimeField.

Поле ratinglevel реалізує рейтингування перевізника в балах на основі класу PositiveSmallIntegerField.

Клас Touristlandmarkrating представляє модель рейтингування музею або пам’ятки.

Поле touristlandmark реалізує музей або пам’ятку, рейтингування якої виконано, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmark.

Поле tourist реалізує туриста, який виставив цей рейтинг, на основі класу ForeignKey зі зв’язком з моделлю Tourist.

Поле ratingtime реалізує час рейтингування музею або пам’ятки на основі класу DateTimeField.

Поле ratinglevel реалізує рейтингування музею або пам’ятки в балах на основі класу PositiveSmallIntegerField.

Клас Touristlandmarkpropositiontransporter представляє модель пропозиції перевізнику за колективною туристичною поїздкою.

Поле touristlandmarkproposition реалізує існуючу пропозицію відвідування музею або пам'ятки, за якою сформована ця пропозиція перевізнику, на основі класу ForeignKey зі зв’язком з моделлю Touristlandmarkproposition.

Поле transporter реалізує перевізника, якому сформовано пропозицію за колективною туристичною поїздкою, на основі класу ForeignKey зі зв’язком з моделлю Transporter.

Поле transportstatus реалізує стан пропозиції перевізнику за колективною туристичною поїздкою на основі класу CharField.

Поле transportcity реалізує місто, з якого має бути розпочато колективну туристичну поїздку за цією пропозицією за колективною туристичною поїздкою, на основі класу ForeignKey зі зв’язком з моделлю Touristcity.

Поле transportstart реалізує час виїзду з початкового міста за цією пропозицією перевізнику за колективною туристичною поїздкою на основі класу DateTimeField.

Поле transportend реалізує час повернення до початкового міста за цією пропозицією перевізнику за колективною туристичною поїздкою на основі класу DateTimeField.

Класи моделей програми використовуються в представленнях програми організації колективних туристичних поїздок для забезпечення визначених функціональних можливостей.

Представлення відтворення інформації про музей або пам'ятку утворено через функцію touristlandmark_view зі створенням шаблону вебсторінки tourlandmark.html, утвореної з застосуванням вільного шаблону [9]. Цей шаблон вебсторінки заповнюється даними через словник, який включає ключі touristlandmark з даними самого музею або пам’ятки, lmarkmainnphoto – з даними основної фотографії музею або пам’ятки, lmarkphotos – зі всім набором фотографій музею або пам’ятки, ratings – з 10 останніми рейтингуваннями цього музею або пам’ятки. Функція має параметри:

– touristlandmark_id, що задає ідентифікатор музею або пам’ятки;

– proposition_period_start – дата періоду початку пропозицій, які будуть відображатися за цим музеєм або пам’яткою;

– proposition_period_end – дата періоду завершення пропозицій, які будуть відображатися за цим музеєм або пам’яткою.

Представлення відтворення музеїв та пам'яток за містом утворено через функцію touristlandmark_by_city_view з параметром touristcity_id, який визначає ідентифікатор міста, в результаті шаблон вебсторінки landmarksbycity.html наповнюється даними самих музеїв та пам’яток, даними обраного міста та переліком міст.

Представлення відтворення музеїв та пам'яток за країною утворено через функцію touristlandmark_by_country_view з параметром touristcountry_id, який визначає ідентифікатор країни, в результаті шаблон вебсторінки landmarksbycountry.html наповнюється даними самих музеїв та пам’яток, даними обраної країни та переліком країн.

Представлення пошуку серед музеїв та пам'яток утворено через функцію touristlandmark_search_view. Параметри пошуку серед музеїв та пам'яток регулюються параметрами GET-запиту, включаючи city для визначення міста розташування музеїв та пам'яток, country для визначення країни розташування музеїв та пам'яток, category для визначення категорії пам’яток, sort для визначення напрямку сортування, page для визначення сторінки відтворення результатів пошуку музеїв та пам'яток порціями по 10.

Представлення відтворення рейтингувань музею або пам'ятки утворено через функцію touristlandmark_ratings_view з параметром touristlandmark_id, який задає ідентифікатор музею або пам’ятки.

Представлення відтворення закладених музеїв та пам'яток утворено через функцію touristlandmark_books_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста.

Представлення відтворення бажаних для відвідування користувачем музеїв або пам'яток утворено через функцію touristlandmark_wishes_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста.

Представлення відтворення власних пропозицій для відвідування музею або пам'ятки утворено через функцію tourist_landmarkpropositions_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста.

Представлення відтворення задіяних пропозицій для відвідування музею або пам'ятки утворено через функцію tourist_participated_landmarkpropositions_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста. Задіяними пропозиціями для відвідування музею або пам'ятки є ті, до яких турист долучився після того, як автор створив таку пропозицію.

Представлення відтворення інформації пропозиції для відвідування музею або пам'ятки утворено через функцію landmarkproposition_view з параметром landmarkproposition_id, який задає ідентифікатор пропозиції для відвідування музею або пам'ятки.

Представлення відтворення своїх рейтингувань музеїв або пам'яток утворено через функцію own_touristlandmark_ratings_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста.

Представлення відтворення своїх рейтингувань перевізників утворено через функцію own_transporter_ratings_view, яка не має параметрів, оскільки вони визначаються авторизацією користувача-туриста.

Представлення відтворення рейтингувань перевізника утворено через функцію transporter_ratings_view з параметром transporter_id, який задає ідентифікатор перевізника.

Представлення відтворення інформації про перевізника утворено через функцію transporter_view з параметром transporter_id, який задає ідентифікатор перевізника.

Представлення закладання музею або пам'ятки утворено через функцію touristlandmark_make_book_view з параметром touristlandmark_id, що задає ідентифікатор музею або пам’ятки, а визначення туриста реалізується за авторизацією, при цьому відбувається перевірка щодо закладання цього музею

або пам’ятки цим туристом раніше, а якщо це виявлено, то реалізується вилучення закладеного музею або пам'ятки.

Представлення висловлення бажання відвідування музею або пам'ятки утворено через функцію touristlandmark_make_wish_view з параметром touristlandmark_id, що задає ідентифікатор музею або пам’ятки, а визначення туриста реалізується за авторизацією, при цьому відбувається перевірка щодо висловлення бажання відвідування цього музею або пам’ятки цим туристом раніше, а якщо це виявлено, то реалізується скасування бажання відвідування музею або пам'ятки.

Представлення формування пропозиції для відвідування музею або пам'ятки утворено через функцію touristlandmark_proposition_create_view з параметром touristlandmark, що задає ідентифікатор музею або пам'ятки, та класом форми TouristlandmarkpropositionForm в основі, утвореного на основі правил [10].

Представлення приєднання до існуючої пропозиції відвідування музею або пам'ятки утворено через функцію touristlandmark_proposition_guest_create_view з параметром touristlandmarkproposition_id, який задає ідентифікатор пропозиції відвідування музею або пам'ятки. Визначення туриста реалізується за авторизацією, при цьому відбувається перевірка щодо приєднання до цієї існуючої пропозиції відвідування музею або пам'ятки цим туристом раніше, а якщо це виявлено, то реалізується скасування приєднання до існуючої пропозиції відвідування музею або пам'ятки.

Представлення голосування за музей або пам'ятку в існуючій пропозиції утворено через функцію touristlandmark_proposition_point_vote_view з параметром touristlandmarkpropositionpoint_id, який задає ідентифікатор музею або пам’ятки у складі пропозиції відвідування музею або пам'ятки. Визначення туриста реалізується за авторизацією, при цьому відбувається перевірка щодо голосування за цей музей або пам'ятку в цій існуючій пропозиції цим туристом

раніше, а якщо це виявлено, то реалізується скасування голосування за музей або пам'ятку в існуючій пропозиції.

Представлення додавання музею або пам'ятки до існуючої пропозиції утворено через функцію touristlandmark_proposition_point_create_view з параметром touristlandmarkproposition_id, що задає ідентифікатор пропозиції відвідування музею або пам'ятки, та класом форми TouristlandmarkpropositionpointForm в основі.

Представлення адресування пропозиції перевізнику за колективною туристичною поїздкою утворено через функцію touristlandmark_proposition_transporter_create_view з параметром touristlandmarkproposition_id, що задає ідентифікатор пропозиції відвідування музею або пам'ятки, та класом форми TouristlandmarkpropositiontransporterForm в основі.

Представлення рейтингування перевізника за результатами колективної туристичної поїздки утворено через функцію transporter_rating_create_view з параметром transporter, що задає ідентифікатор перевізника, та класом форми TransporterratingForm в основі.

Представлення рейтингування музею або пам'ятки утворено через функцію touristlandmark_rating_create_view з параметром touristlandmark, що задає ідентифікатор музею або пам'ятки, та класом форми TouristlandmarkratingForm в основі.

Представлення скасування пропозиції для відвідування музею або пам'ятки утворено через функцію touristlandmark_proposition_delete_view з параметром touristlandmarkproposition_id, що задає ідентифікатор пропозиції для відвідування музею або пам'ятки.

Представлення скасування пропозиції перевізнику за колективною туристичною поїздкою утворено через функцію touristlandmark_proposition_transporter_cancel_view з параметром touristlandmarkpropositiontransporter_id, який задає ідентифікатор пропозиції

перевізнику за колективною туристичною поїздкою. Відбувається перевірка того, що скасування виконує той турист, який створив початкову пропозицію.

Представлення заповнення даних перевізника утворено через функцію transporter_add_view, утворену класом форми TransporterForm в основі.

Представлення оновлення даних перевізника утворено через функцію transporter_edit_view, утворену класом форми TransporterForm в основі. Функція не має параметру, оскільки дані перевізника зв’язані з акаунтом користувача, тому він визначається автоматично.

Представлення відтворення непідтверджених пропозицій перевізнику за колективними туристичними поїздками утворено через функцію unapproved_touristlandmark_transporter_propositions_view. Функція не має параметру, оскільки дані перевізника зв’язані з акаунтом користувача, тому він визначається автоматично.

Представлення відтворення скасованих пропозицій перевізнику за колективними туристичними поїздками утворено через функцію cancelled_touristlandmark_transporter_propositions_view. Функція не має параметру, оскільки дані перевізника зв’язані з акаунтом користувача, тому він визначається автоматично.

Представлення відтворення підтверджених пропозицій перевізнику за колективними туристичними поїздками утворено через функцію approved_touristlandmark_transporter_propositions_view. Функція не має параметру, оскільки дані перевізника зв’язані з акаунтом користувача, тому він визначається автоматично.

Представлення відтворення запланованих відвідувань музеїв або пам'яток утворено через функцію planned_touristlandmark_transporter_propositions_view. Функція не має параметру, оскільки дані перевізника зв’язані з акаунтом користувача, тому він визначається автоматично.

Представлення оновлення екземпляру музею або пам'ятки утворено через функцію touristlandmark_edit_view з параметром touristlandmark_id, який задає ідентифікатор музею або пам’ятки.

Представлення вилучення екземпляру музею або пам'ятки утворено через функцію touristlandmark_delete_view з параметром touristlandmark_id, який задає ідентифікатор музею або пам’ятки.

Представлення підтвердження пропозиції перевізнику за колективною туристичною поїздкою утворено через функцію touristlandmark_proposition_transporter_approve_view з параметром touristlandmarkpropositiontransporter_id, який задає ідентифікатор пропозиції перевізнику за колективною туристичною поїздкою, при цьому відбувається перевірка того, що ця пропозиція адресована саме тому перевізнику, з яким пов’язаний акаунт, який авторизувався.

### 3.3 Висновки за розділом 3

Реалізація програми організації колективних туристичних поїздок виконана в повній мірі з забезпеченням всіх функцій, які було заплановано розробити, та забезпеченням усіх програмних одиниць, які передбачено структурою Django-застосунку для підтримки їх роботи.

# 4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

## 4.1 Призначення програми

Програмне забезпечення організації колективних туристичних поїздок призначене для користувачів, які мають бажання відвідати музеї або пам’ятки в різних містах, але при цьому не мають власного транспорту або бажають скористатися перевагами відвідування у складі групи, та для перевізників, які готові забезпечити підтримку таких поїздок на основі маршрутів, сформованих самими туристами.

## 4.2 Умови виконання програми

Для виконання програми організації колективних туристичних поїздок необхідний сервер, що підтримує використання мови програмування Python, системою керування базами даних MySQL, забезпечує наявність процесора мінімально з 2 ядрами, 4 Гб оперативної пам’яті, 100 Гб сховища на жорсткому диску.

## 4.3 Експлуатація програми організації колективних туристичних поїздок

Програма організації колективних туристичних поїздок передбачає початково роботу користувача в режимі неавторизованого користувача (рис. 4.1). Тоді користувачу доступне меню, з якого можна вибрати пошук за музеями та пам’ятками, відтворення музеїв та пам’яток за заданою країною або містом або перейти до авторизації. Меню доступне для вибору у верхньому правому куті. Там можна перемкнутись між варіантами стрілками або обрати іконку для відтворення всього меню.

У випадку вибору відтворення музеїв та пам’яток за заданою країною або містом доступний випадаючий список, звідки можна задати або тільки

країну, тоді назва сторінки відповідно оновлюється (рис. 4.1), або і країну, і місто, після чого назва також оновлюється. У обох випадках відображається список музеїв та пам’яток за обраною країною або містом. За кожним з них відображається основне зображення, є можливість перейти до сторінки цього музею або пам’ятки.

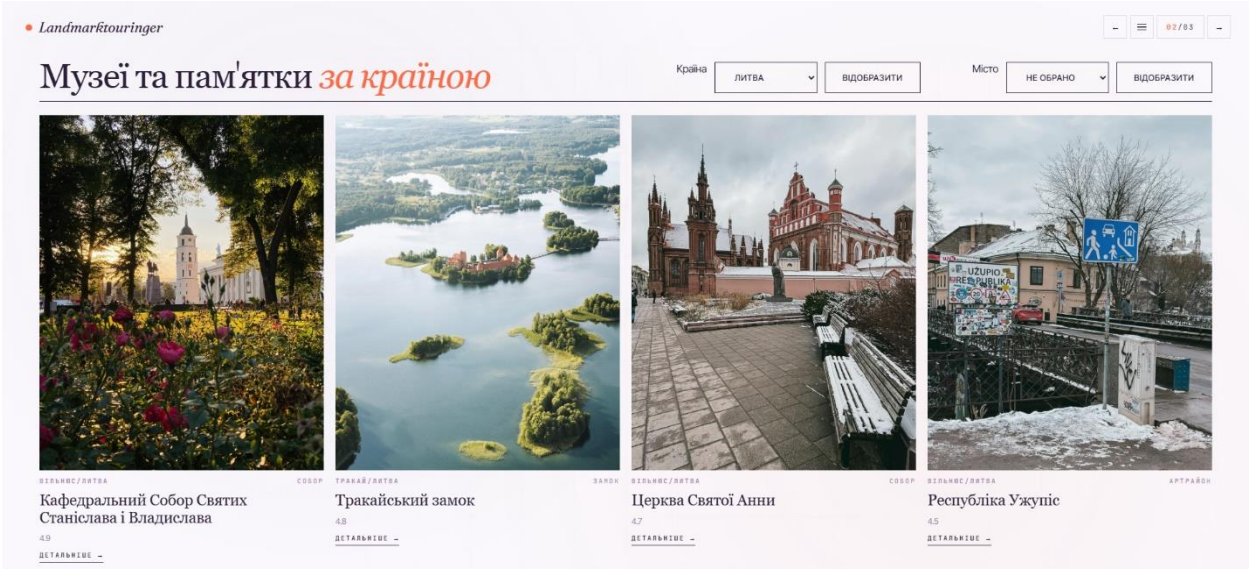


Рисунок 4.1 – Відтворення музеїв та пам’яток за країною для неавторизованого користувача

У випадку роботи туриста після авторизації сторінка дещо змінює свій вигляд (рис. 4.2). При наведенні на картинку відображаються іконки додавання до закладок, бажань.

Коли з програмою працює турист після авторизації, йому доступний набір пунктів меню, що включає перехід до закладок, до бажань з відтворенням відповідних списків музеїв та пам’яток, перехід до пропозицій з вибором відповідного варіанту, при чому заплановані вже для відвідування колективні туристичні поїздки виділені в окремий пункт меню. Свої власні рейтингування і музеїв та пам’яток, і перевізників доступні через відповідний пункт меню (рис. 4.3).

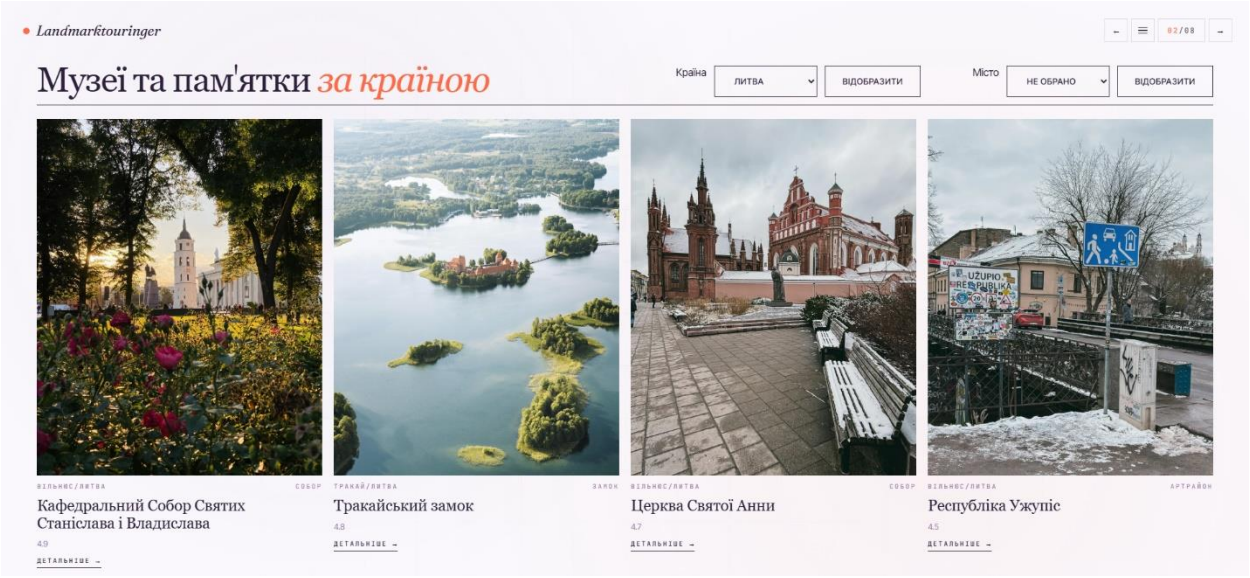


Рисунок 4.2 – Відтворення музеїв та пам’яток за країною для туриста

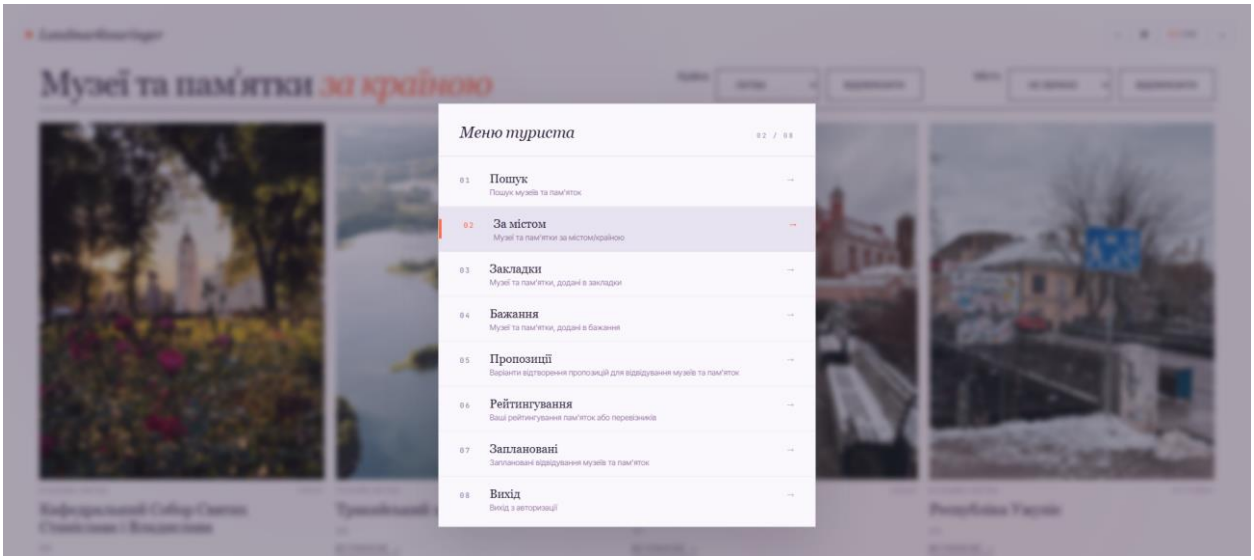


Рисунок 4.3 – Меню туриста

В роботі туриста з програмою після авторизації важливою особливістю є пропозиції для відвідування музеїв та пам’яток. Кожна така пропозиція деталізується окремою сторінкою, яка розпочинається з приведення даних основного музею або пам’ятки, навколо якої така пропозиція побудована (рис. 4.4). Верх сторінки містить основні параметри, дані автора цієї пропозиції. Нижче знаходиться перелік бажаючих відвідати музей або пам’ятку за цієї пропозицією (рис. 4.5). Звідти можна доєднатись до відвідування, якщо користувач ще це не зробив.


Рисунок 4.4 – Сторінка пропозиції відвідування музею або пам’ятки у верхній частині


Рисунок 4.5 – Список бажаючих відвідати музей або пам’ятку за пропозицією

Оскільки до кожної пропозиції відвідування основного музею або пам’ятки інші туристи, які визначили бажання скористатись цією пропозицією, можуть додавати свої власні музеї або пам’ятки, то нижче на сторінці знаходиться список таких запропонованих музеїв або пам’яток (рис. 4.6). За кожним таким варіантом вказується сам об’єкт, часові параметри відвідування, кількість голосів, стан. Якщо кількість голосів перевищує мінімальну потрібну, то стан замінюється на включений. Так само з основною пропозицією. Якщо кількість учасників подолала мінімальну, то пропозиція переходить до стану підготовленої. Як тільки вона направлена перевізнику, пропозиція стає запропоноваю, а далі підтвердженою або скасованою.

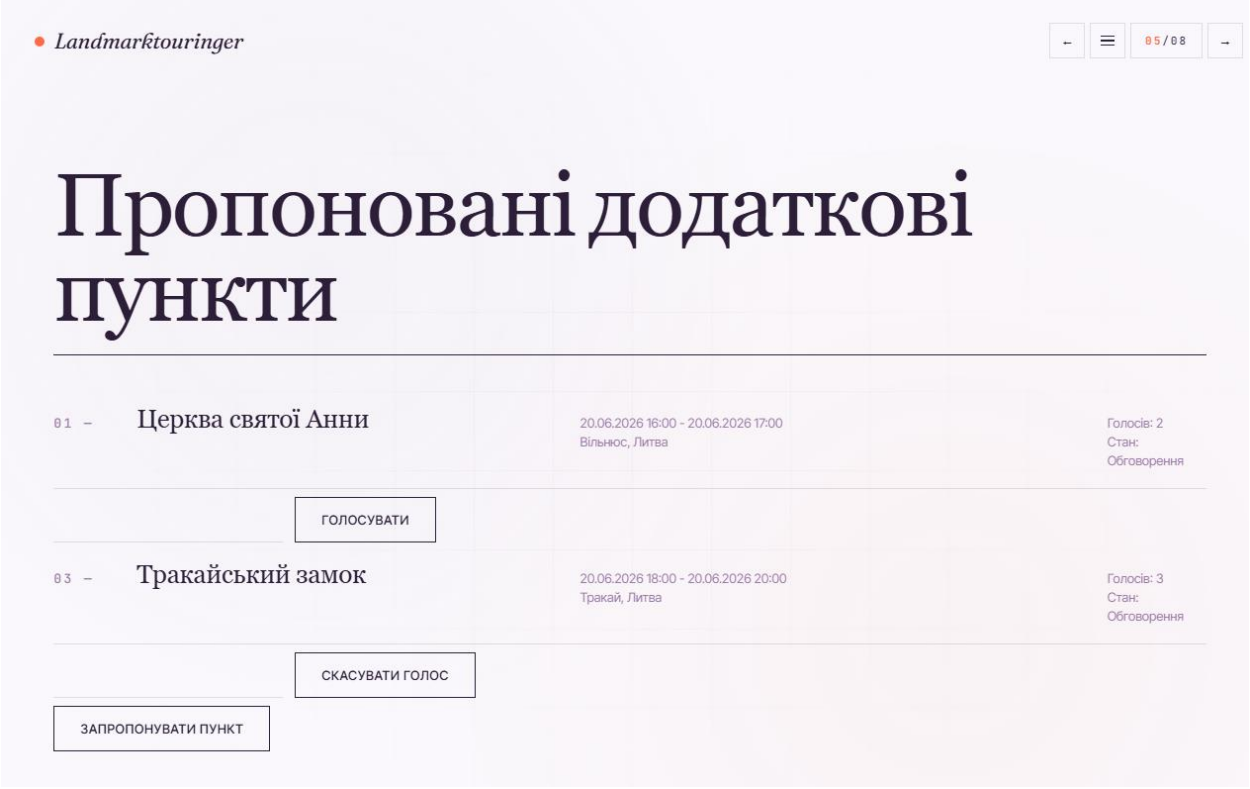


Рисунок 4.6 – Список музеїв або пам’яток, які долучені до пропозиції відвідування основного музею або пам’ятки

У свою чергу перевізник має меню, яке включає перехід до власних даних, до пропозицій та засоби пошуку і відображення музеїв та пам’яток за містами та країнами (рис. 4.7).

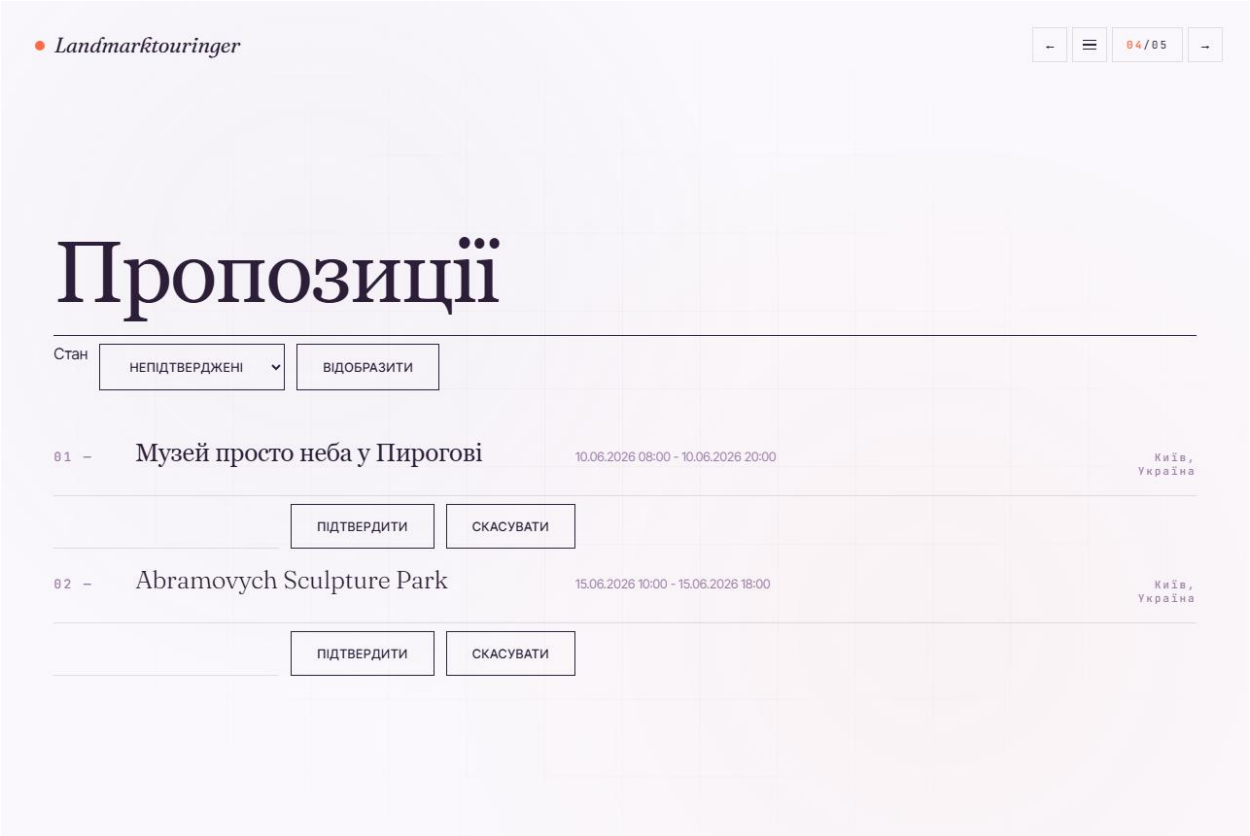


Рисунок 4.7 – Робота перевізника з програмою

Під час роботи перевізника з програмою він має доступ до сторінки з пропозиціями, звідки може з випадаючого списку обрати непідтверджені, підтверджені пропозиції та скасовані. За кожною пропозицією перевізник може вибрати або її підтвердити, або скасувати, якщо така дія не була ним виконана раніше.

## 4.4 Тестування програми організації колективних туристичних поїздок

Тестування програми організації колективних туристичних поїздок було забезпечено з формуванням переліку всіх сценаріїв роботи всіх користувачів з програмою та перевіркою доступності таких сценаріїв, виконання з очікуваним результатом (табл. 4.1).

Таблиця 4.1 – Результати тестування програми організації колективних туристичних поїздок

| Функція | Результат |
|---|---|
| 1 | 2 |
| Відтворення інформації про музей або пам'ятку | + |
| Відтворення музеїв та пам'яток за містом | + |
| Відтворення музеїв та пам'яток за країною | + |
| Пошук серед музеїв та пам'яток | + |
| Авторизація туристів | + |
| Авторизація перевізників | + |
| Відтворення рейтингувань перевізника | + |
| Відтворення рейтингувань музею або пам'ятки | + |
| Відтворення інформації про перевізника | + |
| Реєстрація туристів | + |
| Реєстрація перевізників | + |
| Закладання музею або пам'ятки | + |
| Відтворення закладених музеїв та пам'яток | + |
| Висловлення бажання відвідування музею або пам'ятки | + |
| Відтворення бажаних для відвідування користувачем музеїв або пам'яток | + |
| Формування пропозиції для відвідування музею або пам'ятки | + |
| Відтворення власних пропозицій для відвідування музею або пам'ятки | + |
| Відтворення задіяних пропозицій для відвідування музею або пам'ятки | + |
| Приєднання до існуючої пропозиції відвідування музею або пам'ятки | + |

Продовження таблиці 4.1

| 1 | 2 |
|---|---|
| Визначення параметрів додавання музеїв або пам'яток до пропозиції | + |
| Додавання музею або пам'ятки до існуючої пропозиції | + |
| Голосування за музей або пам'ятку в існуючій пропозиції | + |
| Адресування пропозиції перевізнику за колективною туристичною поїздкою | + |
| Відтворення інформації пропозиції для відвідування музею або пам'ятки | + |
| Рейтингування перевізника за результатами колективної туристичної поїздки | + |
| Рейтингування музею або пам'ятки | + |
| Відтворення своїх рейтингувань перевізників | + |
| Відтворення своїх рейтингувань музеїв або пам'яток | + |
| Вилучення закладеного музею або пам'ятки | + |
| Скасування бажання відвідування музею або пам'ятки | + |
| Скасування пропозиції для відвідування музею або пам'ятки | + |
| Скасування приєднання до існуючої пропозиції відвідування музею або пам'ятки | + |
| Оновлення параметрів додавання музеїв або пам'яток до пропозиції | + |
| Скасування голосування за музей або пам'ятку в існуючій пропозиції | + |
| Скасування пропозиції перевізнику за колективною туристичною поїздкою | + |
| Заповнення даних перевізника | + |

Кінець таблиці 4.1

| 1 | 2 |
|---|---|
| Відтворення непідтверджених пропозицій перевізнику за колективними туристичними поїздками | + |
| Відтворення скасованих пропозицій перевізнику за колективними туристичними поїздками | + |
| Відтворення підтверджених пропозицій перевізнику за колективними туристичними поїздками | + |
| Відтворення запланованих відвідувань музеїв або пам'яток | + |
| Підтвердження пропозиції перевізнику за колективною туристичною поїздкою | + |
| Оновлення даних перевізника | + |
| Оновлення екземпляру музею або пам'ятки | + |
| Вилучення екземпляру музею або пам'ятки | + |
| Оновлення даних туриста | + |
| Відтворення даних туриста | + |

### 4.5 Висновки за розділом 4

Програма організації колективних туристичних поїздок повністю реалізована, про що свідчить вивчення потоків роботи з програмою, опис яких виконано в даному розділі, та проведене тестування, яке підтвердило реалізацію всіх функцій організації колективних туристичних поїздок, що були заплановані.

# ВИСНОВКИ

У роботі розроблено програмне забезпечення організації колективних туристичних поїздок для зменшення витрат на організацію персональних подорожей. Таке зменшення витрат досягається за рахунок формування групи туристів. Відповідно програмне забезпечення дозволяє знайти туристів, за допомогою яких сумарні витрати на транспортування, а частково і на відвідування музеїв, пам’яток будуть розподілені та таким чином зменшені.

Для розробки програми використана мова Python як основна структуруюча одиниця розробки загалом та фреймворк Django як основна структуруюча одиниця веброзробки, також система MySQL для організації роботи з даними.

Програма створена з виділенням основних прийнятих рішень для реалізації програмного забезпечення організації колективних туристичних поїздок, включаючи етапи аналізу проблеми, проєктування і реалізації програми, та описом цих рішень у роботі.

Програмне забезпечення організації колективних туристичних поїздок призначене для користувачів, які мають бажання відвідати музеї або пам’ятки в різних містах, але при цьому не мають власного транспорту або бажають скористатися перевагами відвідування у складі групи, та для перевізників, які готові забезпечити підтримку таких поїздок на основі маршрутів, сформованих самими туристами.

Для врахування особливостей індивідуалістичного підходу до формування колективних туристичних поїздок користувачі формують початкову пропозицію на основі одного конкретно виділеного музею або пам’ятки, тільки після чого до неї шляхом голосування з визначенням мінімальної кількості відвідувачів додаються інші музеї або пам’ятки, а вже сформована таким чином група для подорожі пропонує перевізнику забезпечити підтримку такого маршруту.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Howbout – shared calendar. URL: https://howbout.app/ (date of access: 03.05.2026).

2. Group Tour – Apps on Google Play. URL: https://play.google.com/store/apps/details?id=com.grouptour.travelapp (date of access: 03.05.2026).

3. Split expenses with friends: Splitwise. URL: https://www.splitwise.com/ (date of access: 03.05.2026).

4. FastAPI. URL: https://fastapi.tiangolo.com/ (date of access: 06.05.2026).

5. Design philosophies. Django documentation. Django. URL: https://docs.djangoproject.com/en/6.0/misc/design-philosophies/ (date of access: 06.05.2026).

6. Erickson J. MySQL: Understanding What It Is and How It's Used. *Oracle United Kingdom*. URL: https://www.oracle.com/uk/mysql/what-is-mysql/ (date of access: 06.05.2026).

7. Writing your first Django app, part 1. Django documentation. Django. URL: https://docs.djangoproject.com/en/6.0/intro/tutorial01/ (date of access: 07.05.2026).

8. Models. Django documentation. Django. URL: https://docs.djangoproject.com/en/6.0/topics/db/models/ (date of access: 07.05.2026).

9. Free Template 625 Folio Slideshow. URL: https://templatemo.com/tm-625-folio-slideshow (date of access: 10.05.2026).

10. Working with forms. Django documentation. Django. URL: https://docs.djangoproject.com/en/6.0/topics/forms/#forms-in-django (date of access: 10.05.2026).

# ДОДАТОК А

## Технічне завдання

## Вступ

Колективні туристичні поїздки можуть бути організовані не тільки для існуючої групи, яка формується туристичною компанією, але і для колективу, який формується віддалено на основі використання програмного забезпечення. У такому випадку колективна туристична поїздка передбачає необхідність формування маршруту, доєднання до нового колективу, визначення перевізника. Саме це має бути забезпечено головним чином програмою організації колективних туристичних поїздок.

## А.1 Підстави для розробки

Робота виконана за темою «Програмне забезпечення організації колективних туристичних поїздок» у відповідності з даними Наказу № 139 від 7.04.2026 року в Національному університеті «Запорізька політехніка».

## А.2 Призначення розробки

Програма організації колективних туристичних поїздок призначена для користувачів, які мають бажання відвідати музеї або пам’ятки в різних містах, але при цьому не мають власного транспорту або бажають скористатися перевагами відвідування у складі групи, та для перевізників, які готові забезпечити підтримку таких поїздок. Першою групою користувачів програма може використовуватися для формування маршруту поїздки, доєднання до колективної туристичної поїздки, пошуку перевізників. Другою групою користувачів програма може використовуватися для знаходження варіантів підтримки поїздок.

## А.3 Основні вимоги до програми

### А.3.1 Вимоги до функціональних характеристик

Програма організації колективних туристичних поїздок має відповідати функціональним характеристикам, серед яких:

– визначення музеїв та пам'яток шляхом додавання екземплярів, оновлення екземплярів та вилучення;

– відтворення інформації про музей або пам'ятку;

– відтворення музеїв та пам'яток за містом;

– відтворення музеїв та пам'яток за країною;

– пошук серед музеїв та пам'яток;

– закладання музеїв та пам'яток і вилучення закладених;

– відтворення закладених музеїв та пам'яток;

– висловлення бажання відвідування музею або пам'ятки або скасування такого бажання;

– відтворення бажаних для відвідування користувачем музеїв або пам'яток;

– формування пропозиції для відвідування музею або пам'ятки або скасування її;

– відтворення власних пропозицій для відвідування музею або пам'ятки;

– відтворення задіяних пропозицій для відвідування музею або пам'ятки;

– приєднання до існуючої пропозиції відвідування музею або пам'ятки або скасування;

– визначення параметрів додавання музеїв або пам'яток до пропозиції;

– оновлення параметрів додавання музеїв або пам'яток до пропозиції;

– додавання музею або пам'ятки до існуючої пропозиції;

– голосування за музей або пам'ятку в існуючій пропозиції або його скасування;

– адресування пропозиції перевізнику за колективною туристичною поїздкою або її скасування;

– відтворення непідтверджених пропозицій перевізнику за колективними туристичними поїздками;

– відтворення скасованих пропозицій перевізнику за колективними туристичними поїздками;

– відтворення підтверджених пропозицій перевізнику за колективними туристичними поїздками;

– відтворення запланованих відвідувань музеїв або пам'яток;

– підтвердження або скасування пропозиції перевізнику за колективною туристичною поїздкою;

– визначення інформації про перевізника шляхом заповнення даних, оновлення;

– авторизація і реєстрація;

– рейтингування перевізника за результатами колективної туристичної поїздки;

– відтворення інформації про перевізника;

– рейтингування музею або пам'ятки;

– відтворення рейтингувань перевізника;

– відтворення рейтингувань музею або пам'ятки;

– відтворення своїх рейтингувань перевізників;

– відтворення своїх рейтингувань музеїв або пам'яток;

– відтворення інформації пропозиції для відвідування музею або пам'ятки.

### А.3.2 Умови експлуатації

Експлуатаційні вимоги до програми організації колективних туристичних поїздок включають виділення серверу для роботи з програмою шляхом використання сервера підприємства або замовлення хостингу. У обох

випадках сервер повинен забезпечувати роботу з системою керування базами даних, мовою програмування, що застосовуватимуться для роботи з програмою після прийняття цих рішень в даній роботі.

### А.3.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення хостингу шляхом оренди сервера або використання власного серверу необхідно враховувати, що мінімальними вимогами, які на сервері виділені виключно під роботу з застосунком організації колективних туристичних поїздок передбачають наявність процесора мінімально з 2 ядрами, 4 Гб оперативної пам’яті, 100 Гб сховища на жорсткому диску.

## А.4 Порядок контролю та приймання

Робота над дипломною кваліфікаційною роботою має виконуватися протягом 8-тижневого циклу, під час якого має бути виконано саму роботу, проведено перевірку роботи на нормоконтроль, проведено процедури рецензування та захисту роботи.

# ДОДАТОК Б

## Текст програми

```
from django.shortcuts import render
from .models import Touristcountry,
    Touristcity,
    Landmarkcategory,
    Touristlandmark,
    Touristlandmarkphoto,
    Tourist,
    Touristlandmarkproposition,
    Touristlandmarkpropositionpoint,
    Transporter,
    Transporterrating,
    Touristlandmarkrating,
    Touristlandmarkpropositiontransporter
from django.core.paginator import Paginator


def touristlandmark_view(request, touristlandmark_id,
proposition_period_start = None, proposition_period_end = None):
    tourist_landmark =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").filte
r(id = touristlandmark_id)
    if not tourist_landmark:
        return render(
            request,
            "landmarktouringer/error.html",
                {
                    "message": "Немає такої пам’ятки, ви задали
неправильні параметри"
                }
            )
    else:
        tourist_landmark = tourist_landmark[0]
        tourist_landmark_photos =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
            basephoto = False)
        tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
            basephoto = True)
        if not tourist_landmark_mainphoto:
```

```
                tourist_landmark_mainphoto =
tourist_landmark_photos[0]
            tourist_landmark_ratings =
Touristlandmarkrating.objects.select_related("tourist").select_related
("tourist__djappuser").filter(touristlandmark = tourist_landmark)
            tourist_landmark_ratings =
tourist_landmark_ratings.order_by("-ratingtime")
            tourist_landmark_ratings =
tourist_landmark_ratings[0:10]
            user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
            if not user_tourist:
                return render(
                    request,
                    "landmarktouringer/tourlandmark.html",
                        {
                            "touristlandmark": tourist_landmark,
                            "lmarkmainnphoto":
tourist_landmark_mainphoto,
                            "lmarkphotos": tourist_landmark_photos,
                            "ratings":  tourist_landmark_ratings
                        }
                    )
            else:
                user_tourist = user_tourist[0]
                if tourist_landmark in
user_tourist.touristlandmarkwishes:
                    tourist_landmark.wished = True
                else:
                    tourist_landmark.wished = False
                user_tourist.touristlandmarkwishes = None
                if tourist_landmark in
user_tourist.touristlandmarkbooks:
                    tourist_landmark.booked = True
                else:
                    tourist_landmark.booked = False
                user_tourist.touristlandmarkbooks = None
                ready_own_tourist_propositions =
Touristlandmarkproposition.objects.select_related("tourist").select_re
lated("touristlandmark").prefetch_related("touristlandmarkpropositiong
uests").filter(touristlandmark = tourist_landmark,
```

```
                    touristlandmarkpropositionguests__in =
[user_tourist])
                if ready_own_tourist_propositions:
                    tourist_landmark.ownpropositions =
ready_own_tourist_propositions
                else:
                    tourist_landmark.ownpropositions = None
                from datetime import datetime
                if proposition_period_start:
                    proposition_period_start =
datetime.strptime(proposition_period_start, '%d %m %Y %I:%M%p')
                else:
                    proposition_period_start = datetime.now()
                from dateutil.relativedelta import relativedelta
                if proposition_period_end:
                    proposition_period_end =
datetime.strptime(proposition_period_end, '%d %m %Y %I:%M%p')
                else:
                    proposition_period_end =
proposition_period_start + relativedelta(months=1)
                tourist_landmark_propositions =
Touristlandmarkproposition.objects.filter(touristlandmark =
tourist_landmark)
                tourist_landmark_propositions =
tourist_landmark.propositions.exclude(touristvisittimestart__gt =
proposition_period_end)
                tourist_landmark_propositions =
tourist_landmark.propositions.exclude(touristvisittimeend__lt =
proposition_period_start)
                tourist_landmark.propositions =
tourist_landmark_propositions
                return render(
                    request,
                    "landmarktouringer/tourlandmark.html",
                        {
                            "touristlandmark": tourist_landmark,
                            "lmarkmainnphoto":
tourist_landmark_mainphoto,
                            "lmarkphotos": tourist_landmark_photos,
                            "tourist": user_tourist,
                            "ratings":  tourist_landmark_ratings
                        }
                    )
```

```
      def touristlandmark_by_city_view(request, touristcity_id):
          touristcity =
Touristcity.objects.select_related("touristcountry").filter(id =
touristcity_id)
          touristcites =
Touristcity.objects.select_related("touristcountry").all()
          if touristcity:
              touristcity = touristcity[0]
              tourist_landmarks =
Touristlandmark.objects.select_related("landmarkcategory").filter(tour
istcity = touristcity).order_by("-landmarkrating")
              tourist_landmarks = Paginator(tourist_landmarks, 10)
              tourist_landmarks_on_page = request.GET.get('page')
              tourist_landmarks =
tourist_landmarks.get_page(tourist_landmarks_on_page)
              for tourist_landmark in tourist_landmarks:
                  tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                      basephoto = True)
                  if not tourist_landmark_mainphoto:
                      tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                          basephoto = False)
                      tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                  else:
                      tourist_landmark.mainphoto =
tourist_landmark_mainphoto
              user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
              if not user_tourist:
                  return render(
                      request,
                      "landmarktouringer/landmarksbycity.html",
                          {
                              "touristlandmarks": tourist_landmarks,
                              "touristcity": touristcity,
                              "touristcites": touristcites
                          }
                      )
```

```
                else:
                    user_tourist = user_tourist[0]
                    for tourist_landmark in tourist_landmarks:
                        if tourist_landmark in
user_tourist.touristlandmarkwishes:
                            tourist_landmark.wished = True
                        else:
                            tourist_landmark.wished = False
                        if tourist_landmark in
user_tourist.touristlandmarkbooks:
                            tourist_landmark.booked = True
                        else:
                            tourist_landmark.booked = False
                    user_tourist.touristlandmarkwishes = None
                    user_tourist.touristlandmarkbooks = None
                return render(
                    request,
                    "landmarktouringer/landmarksbycity.html",
                        {
                            "landmarks": tourist_landmarks,
                            "touristcity": touristcity,
                            "touristcites": touristcites
                        }
                    )
            else:
                 return render(
                    request,
                    "landmarktouringer/error.html",
                        {
                            "message": "Немає такого міста, ви задали
неправильні параметри"
                        }
                    )

        def touristlandmark_by_country_view(request, touristcountry_id):
            touristcountry = Touristcountry.objects.filter(id =
touristcountry_id)
            touristcountries = Touristcountry.objects.all()
            if touristcountry:
                touristcountry = touristcountry[0]
                tourist_landmarks =
Touristlandmark.objects.select_related("landmarkcategory").filter(tour
istcity__touristcountry = touristcountry).order_by("-landmarkrating")
```

```
            tourist_landmarks = Paginator(tourist_landmarks, 10)
            tourist_landmarks_on_page = request.GET.get('page')
            tourist_landmarks =
tourist_landmarks.get_page(tourist_landmarks_on_page)
            for tourist_landmark in tourist_landmarks:
                tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                    basephoto = True)
                if not tourist_landmark_mainphoto:
                    tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                        basephoto = False)
                    tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                else:
                    tourist_landmark.mainphoto =
tourist_landmark_mainphoto
            user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
            if not user_tourist:
                return render(
                    request,
                    "landmarktouringer/landmarksbycountry.html",
                        {
                            "touristlandmarks": tourist_landmarks,
                            "touristcountry": touristcountry,
                            "touristcountries": touristcountries
                        }
                    )
            else:
                user_tourist = user_tourist[0]
                for tourist_landmark in tourist_landmarks:
                    if tourist_landmark in
user_tourist.touristlandmarkwishes:
                        tourist_landmark.wished = True
                    else:
                        tourist_landmark.wished = False
                    if tourist_landmark in
user_tourist.touristlandmarkbooks:
                        tourist_landmark.booked = True
```

```
                    else:
                        tourist_landmark.booked = False
                user_tourist.touristlandmarkwishes = None
                user_tourist.touristlandmarkbooks = None
            return render(
                request,
                "landmarktouringer/landmarksbycountry.html",
                    {
                        "touristlandmarks": tourist_landmarks,
                        "touristcountry": touristcountry,
                        "touristcountries": touristcountries
                    }
                )
        else:
             return render(
                request,
                "landmarktouringer/error.html",
                    {
                        "message": "Немає такої країни, ви задали
неправильні параметри"
                    }
                )

    def touristlandmark_search_view(request):
        touristcountries = Touristcountry.objects.all()
        for touristcountry in touristcountries:
            tourist_cities =
Touristcity.objects.filter(touristcountry = touristcountry)
            touristcountry.cities = tourist_cities
        landmark_categories = Landmarkcategory.objects.all()
        touristcity_offname = request.GET.get('city')
        tourist_landmarks = []
        if touristcity_offname:
            touristcity = Touristcity.objects.filter(cityoffname =
touristcity_offname)
            if touristcity:
                touristcity = touristcity[0]
                tourist_landmarks =
Touristlandmark.objects.select_related("landmarkcategory").filter(tour
istcity = touristcity)
        touristcountry_offname = request.GET.get('country')
        if touristcountry_offname:
```

```
            touristcountry =
Touristcountry.objects.filter(countryoffname = touristcountry_offname)
            if touristcountry:
                touristcountry = touristcountry[0]
                if not tourist_landmarks:
                    tourist_landmarks =
Touristlandmark.objects.select_related("landmarkcategory").select_rela
ted("touristcity").select_related("touristcity__touristcountry").filte
r(touristcity__touristcountry = touristcountry)
                else:
                    tourist_landmarks =
tourist_landmarks.filter(touristcity__touristcountry = touristcountry)
        landmarkcategory_id = request.GET.get('category')
        if landmarkcategory_id:
            landmarkcategory_id = int(landmarkcategory_id)
            landmarkcategory = Landmarkcategory.objects.filter(id =
landmarkcategory_id)
            if landmarkcategory:
                landmarkcategory = landmarkcategory[0]
                if not tourist_landmarks:
                    tourist_landmarks =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").filte
r(landmarkcategory = landmarkcategory)
                else:
                    tourist_landmarks =
tourist_landmarks.filter(landmarkcategory = landmarkcategory)
        if not tourist_landmarks:
            tourist_landmarks =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").all()
        landmark_sort = request.GET.get('sort')
        if not landmark_sort:
            tourist_landmarks = tourist_landmarks.order_by("-
landmarkrating")
        else:
            landmark_sort = int(landmark_sort)
            if landmark_sort == 0:
                tourist_landmarks = tourist_landmarks.order_by("-
landmarkrating")
            elif landmark_sort == 1:
                tourist_landmarks =
tourist_landmarks.order_by("landmarkrating")
```

```
            elif landmark_sort == 2:
                tourist_landmarks =
tourist_landmarks.order_by("landmarktitle")
            elif landmark_sort == 3:
                tourist_landmarks = tourist_landmarks.order_by("-
landmarktitle")
        tourist_landmarks = Paginator(tourist_landmarks, 10)
        tourist_landmarks_on_page = request.GET.get('page')
        tourist_landmarks =
tourist_landmarks.get_page(tourist_landmarks_on_page)
        for tourist_landmark in tourist_landmarks:
            tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                basephoto = True)
            if not tourist_landmark_mainphoto:
                tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                    basephoto = False)
                tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
            else:
                tourist_landmark.mainphoto =
tourist_landmark_mainphoto
            user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
            if not user_tourist:
                return render(
                    request,
                    "landmarktouringer/landmarkssearch.html",
                        {
                            "touristlandmarks": tourist_landmarks
                        }
                    )
            else:
                user_tourist = user_tourist[0]
                for tourist_landmark in tourist_landmarks:
                    if tourist_landmark in
user_tourist.touristlandmarkwishes:
                        tourist_landmark.wished = True
                    else:
```

```
                        tourist_landmark.wished = False
                    if tourist_landmark in
user_tourist.touristlandmarkbooks:
                        tourist_landmark.booked = True
                    else:
                        tourist_landmark.booked = False
                user_tourist.touristlandmarkwishes = None
                user_tourist.touristlandmarkbooks = None
            return render(
                request,
                "landmarktouringer/landmarkssearch.html",
                    {
                        "touristlandmarks": tourist_landmarks
                    }
                )

      def touristlandmark_ratings_view(request, touristlandmark_id):
          tourist_landmark =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").filte
r(id = touristlandmark_id)
          if not tourist_landmark:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Немає такої пам’ятки, ви задали
неправильні параметри"
                      }
                  )
          else:
              tourist_landmark = tourist_landmark[0]
              tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                  basephoto = True)
              if not tourist_landmark_mainphoto:
                  tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                  basephoto = False)
                  tourist_landmark_mainphoto =
tourist_landmark_mainphoto[0]
```

```
            tourist_landmark_ratings =
Touristlandmarkrating.objects.select_related("tourist").select_related
("tourist__djappuser").filter(touristlandmark = tourist_landmark)
            tourist_landmark_ratings =
tourist_landmark_ratings.order_by("-ratingtime")
            tourist_landmark_ratings =
Paginator(tourist_landmark_ratings, 10)
            tourist_landmark_ratings_on_page =
request.GET.get('page')
            tourist_landmark_ratings =
tourist_landmark_ratings.get_page(tourist_landmark_ratings_on_page)
            user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
            if not user_tourist:
                return render(
                    request,
                    "landmarktouringer/tourlandmarkratings.html",
                        {
                            "touristlandmark": tourist_landmark,
                            "lmarkmainnphoto":
tourist_landmark_mainphoto,
                            "ratings":  tourist_landmark_ratings
                        }
                    )
            else:
                user_tourist = user_tourist[0]
                if tourist_landmark in
user_tourist.touristlandmarkwishes:
                    tourist_landmark.wished = True
                else:
                    tourist_landmark.wished = False
                user_tourist.touristlandmarkwishes = None
                if tourist_landmark in
user_tourist.touristlandmarkbooks:
                    tourist_landmark.booked = True
                else:
                    tourist_landmark.booked = False
                user_tourist.touristlandmarkbooks = None
                return render(
                    request,
                    "landmarktouringer/tourlandmarkratings.html",
```

```
                        {
                            "touristlandmark": tourist_landmark,
                            "lmarkmainnphoto":
tourist_landmark_mainphoto,
                            "tourist": user_tourist,
                            "ratings":  tourist_landmark_ratings
                        }
                    )

      def touristlandmark_wishes_view(request):
          user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related(Prefetch(
'touristlandmarkwishes', queryset =
Touristlandmark.objects.all().only('id')).prefetch_related("touristlan
dmarkbooks").filter(djappuser = request.user)
          if not user_tourist:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Для перегляду бажаних для
відвідування пам'яток потрібно авторизуватись"
                      }
                  )
          else:
                  tourist_landmark_ids =
user_tourist.touristlandmarkwishes
                  tourist_landmarks =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").filte
r(id__in = tourist_landmark_ids)
                  for tourist_landmark in tourist_landmarks:
                      tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                          basephoto = True)
                      if not tourist_landmark_mainphoto:
                          tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                              basephoto = False)
                          tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
```

```
                    else:
                        tourist_landmark.mainphoto =
tourist_landmark_mainphoto
                    if tourist_landmark in
user_tourist.touristlandmarkbooks:
                        tourist_landmark.booked = True
                    else:
                        tourist_landmark.booked = False
                user_tourist.touristlandmarkbooks = None
            return render(
                request,
                "landmarktouringer/landmarkwishes.html",
                    {
                        "touristlandmarks": tourist_landmarks,
                        "tourist": user_tourist
                    }
                )

      def touristlandmark_books_view(request):
          user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related(Prefetch(
'touristlandmarkbooks', queryset =
Touristlandmark.objects.all().only('id')).prefetch_related("touristlan
dmarkwishes").filter(djappuser = request.user)
          if not user_tourist:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Для перегляду закладених
пам'яток потрібно авторизуватись"
                      }
                  )
          else:
                  tourist_landmark_ids =
user_tourist.touristlandmarkbooks
                  tourist_landmarks =
Touristlandmark.objects.select_related("touristcity").select_related("
touristcity__touristcountry").select_related("landmarkcategory").filte
r(id__in = tourist_landmark_ids)
                  for tourist_landmark in tourist_landmarks:
```

```
                    tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                        basephoto = True)
                    if not tourist_landmark_mainphoto:
                        tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                            basephoto = False)
                        tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                    else:
                        tourist_landmark.mainphoto =
tourist_landmark_mainphoto
                    if tourist_landmark in
user_tourist.touristlandmarkwishes:
                        tourist_landmark.wished = True
                    else:
                        tourist_landmark.wished = False
                user_tourist.touristlandmarkwishes = None
            return render(
                request,
                "landmarktouringer/landmarkbooks.html",
                    {
                        "touristlandmarks": tourist_landmarks,
                        "tourist": user_tourist
                    }
                )

      def tourist_landmarkpropositions_view(request):
          user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related('touristl
andmarkbooks').prefetch_related("touristlandmarkwishes").filter(djappu
ser = request.user)
          if not user_tourist:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Для перегляду пропозицій для
відвідування пам'яток потрібно авторизуватись"
                      }
                  )
```

```
            else:
                    tourist_landmark_propositions =
Touristlandmarkproposition.objects.select_related("touristlandmark__to
uristcity").select_related("touristlandmark__touristcity__touristcount
ry").select_related("touristlandmark__landmarkcategory").filter(touris
t = user_tourist).order_by("-touristvisittimestart")
                    for tourist_landmark_propositionin
tourist_landmark_propositions:
                        tourist_landmark =
tourist_landmark_proposition.touristlandmark
                        tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                            basephoto = True)
                        if not tourist_landmark_mainphoto:
                            tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                                basephoto = False)
                            tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                        else:
                            tourist_landmark.mainphoto =
tourist_landmark_mainphoto
                        if tourist_landmark in
user_tourist.touristlandmarkwishes:
                            tourist_landmark.wished = True
                        else:
                            tourist_landmark.wished = False
                    tourist_landmark_proposition.touristlandmark =
tourist_landmark
                    user_tourist.touristlandmarkwishes = None
                    user_tourist.touristlandmarkbooks = None
                return render(
                    request,
                    "landmarktouringer/landmarkpropos.html",
                        {
                            "propositions":
tourist_landmark_propositions,
                            "tourist": user_tourist
                        }
                    )
```

```
      def tourist_participated_landmarkpropositions_view(request):
          user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related('touristl
andmarkbooks').prefetch_related("touristlandmarkwishes").filter(djappu
ser = request.user)
          if not user_tourist:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Для перегляду пропозицій для
відвідування пам'яток потрібно авторизуватись"
                      }
                  )
          else:
                  tourist_landmark_propositions =
Touristlandmarkproposition.objects.select_related("touristlandmark__to
uristcity").select_related("touristlandmark__touristcity__touristcount
ry").select_related("touristlandmark__landmarkcategory").prefetch_rela
ted("touristlandmarkpropositionguests").filter(touristlandmarkproposit
ionguests__in = [user_tourist]).order_by("-touristvisittimestart")
                  for tourist_landmark_proposition in
tourist_landmark_propositions:
                      tourist_landmark =
tourist_landmark_proposition.touristlandmark
                      tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                          basephoto = True)
                      if not tourist_landmark_mainphoto:
                          tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                              basephoto = False)
                          tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                      else:
                          tourist_landmark.mainphoto =
tourist_landmark_mainphoto
                      if tourist_landmark in
user_tourist.touristlandmarkwishes:
                          tourist_landmark.wished = True
                      else:
```

```
                        tourist_landmark.wished = False
                    tourist_landmark_proposition.touristlandmark =
tourist_landmark
                user_tourist.touristlandmarkwishes = None
                user_tourist.touristlandmarkbooks = None
            return render(
                request,
                "landmarktouringer/landmarkpartpropos.html",
                    {
                        "propositions":
tourist_landmark_propositions,
                        "tourist": user_tourist
                    }
                )

      def landmarkproposition_view(request, landmarkproposition_id):
          user_tourist = None
          user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related('touristl
andmarkbooks').prefetch_related("touristlandmarkwishes").filter(djappu
ser = request.user)
          tourist_landmark_proposition =
Touristlandmarkproposition.objects.select_related("touristlandmark__to
uristcity").select_related("touristlandmark__touristcity__touristcount
ry").select_related("touristlandmark__landmarkcategory").prefetch_rela
ted("touristlandmarkpropositionguests").filter(id =
landmarkproposition_id)
          if not tourist_landmark_proposition:
              return render(
                  request,
                  "landmarktouringer/error.html",
                      {
                          "message": "Такої пропозиції для
відвідування пам'яток немає"
                      }
                  )
          tourist_landmark =
tourist_landmark_proposition.touristlandmark
          tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
              basephoto = True)
          if not tourist_landmark_mainphoto:
```

```
                tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                    basephoto = False)
                tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
            else:
                tourist_landmark.mainphoto = tourist_landmark_mainphoto
            if user_tourist and tourist_landmark in
user_tourist.touristlandmarkwishes:
                tourist_landmark.wished = True
            else:
                tourist_landmark.wished = False
            tourist_landmark_proposition.touristlandmark =
tourist_landmark
            tourist_landmark_proposition_points =
Touristlandmarkpropositionpoint.objects.select_related("touristlandmar
k__touristcity").select_related("touristlandmark__touristcity__tourist
country").select_related("touristlandmark__landmarkcategory").prefetch
_related("touristlandmarkpropositionpointvotes").filter(touristlandmar
kproposition = tourist_landmark_proposition).order_by("-
touristvisittimestart")
            for tourist_landmark_proposition_point in
tourist_landmark_proposition_points:
                tourist_landmark =
tourist_landmark_proposition.touristlandmark
                tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                    basephoto = True)
                if not tourist_landmark_mainphoto:
                    tourist_landmark_mainphoto =
Touristlandmarkphoto.objects.filter(touristlandmark =
tourist_landmark,
                        basephoto = False)
                    tourist_landmark.mainphoto =
tourist_landmark_mainphoto[0]
                else:
                    tourist_landmark.mainphoto =
tourist_landmark_mainphoto
                if user_tourist and tourist_landmark in
user_tourist.touristlandmarkwishes:
                    tourist_landmark.wished = True
                else:
```

```
                tourist_landmark.wished = False
            tourist_landmark_proposition_point.touristlandmark =
tourist_landmark
        if user_tourist:
            user_tourist.touristlandmarkwishes = None
            user_tourist.touristlandmarkbooks = None
        touristlandmarkproposition_transporter =
Touristlandmarkpropositiontransporter.objects.select_related("transpor
ter").select_related("transporter__djappuser").filter(touristlandmarkp
roposition = tourist_landmark_proposition)
        return render(
            request,
                "landmarktouringer/landmarkprop.html",
                    {
                        "proposition": tourist_landmark_proposition,
                        "landmark": tourist_landmark,
                        "points":
tourist_landmark_proposition_points,
                        "tourist": user_tourist,
                        "transporter":
touristlandmarkproposition_transporter
                    }
                )

    def own_touristlandmark_ratings_view(request):
        user_tourist = None
        user_tourist =
Tourist.objects.select_related("djappuser").prefetch_related("touristl
andmarkwishes").prefetch_related("touristlandmarkbooks").filter(djappu
ser = request.user)
        if not user_tourist:
            return render(
                request,
                "landmarktouringer/error.html",
                    {
                        "message": "Для перегляду ваших рейтингувань
спочатку авторизуйтесь"
                    }
                )
        else:
            user_tourist = user_tourist[0]
```

```
            tourist_landmark_ratings =
Touristlandmarkrating.objects.select_related("tourist").select_related
("tourist__djappuser").filter(tourist = user_tourist)
            tourist_landmark_ratings =
tourist_landmark_ratings.order_by("-ratingtime")
            tourist_landmark_ratings =
Paginator(tourist_landmark_ratings, 10)
            tourist_landmark_ratings_on_page =
request.GET.get('page')
            tourist_landmark_ratings =
tourist_landmark_ratings.get_page(tourist_landmark_ratings_on_page)
            for tourist_landmark_rating in tourist_landmark_ratings:
                if tourist_landmark_rating.touristlandmark in
user_tourist.touristlandmarkwishes:
                    tourist_landmark_rating.touristlandmark.wished =
True
                else:
                    tourist_landmark_rating.touristlandmark.wished =
False
                user_tourist.touristlandmarkwishes = None
                if tourist_landmark_rating.touristlandmark in
user_tourist.touristlandmarkbooks:
                    tourist_landmark_rating.touristlandmark.booked =
True
                else:
                    tourist_landmark_rating.touristlandmark.booked =
False
                user_tourist.touristlandmarkbooks = None
                return render(
                    request,
                    "landmarktouringerown/ownlandmarkratings.html",
                        {
                            "tourist": user_tourist,
                            "ratings":  tourist_landmark_ratings
                        }
                    )
```

# ДОДАТОК В

## Слайди презентації


Рисунок В.1 – Початковий слайд

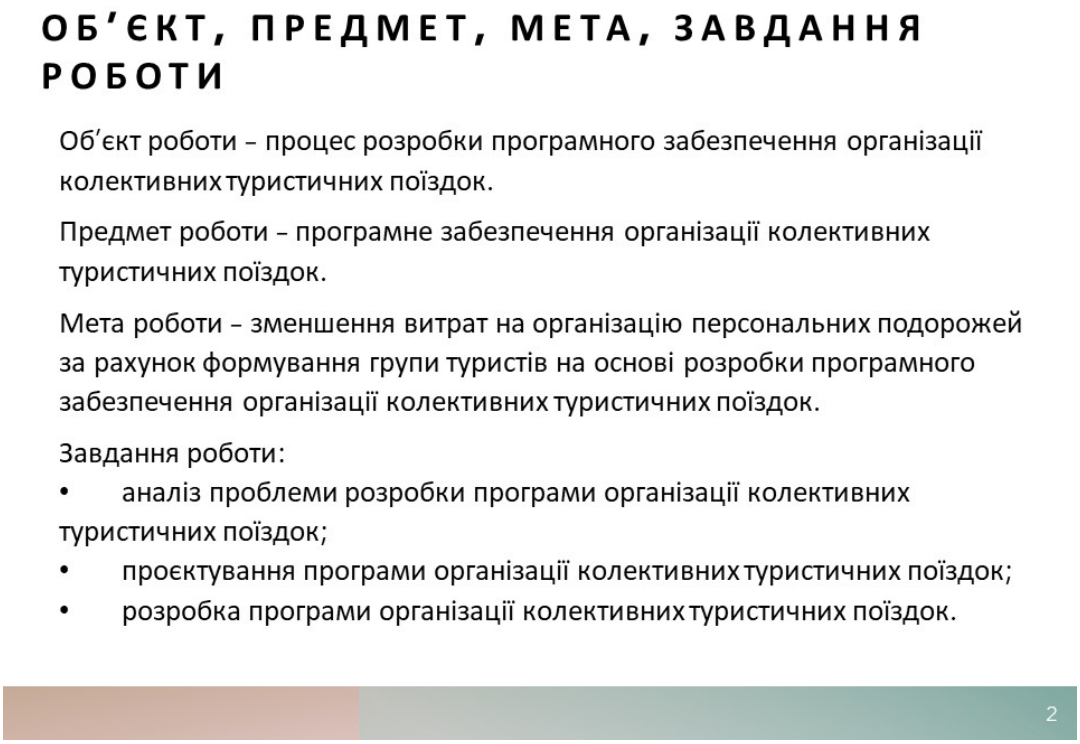


Рисунок В.2 – Об'єкт, предмет, мета, завдання роботи

## ПОРІВНЯННЯ ПРОГРАМНИХ ЗАСОБІВ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ТУРИСТИЧНИХ ПОЇЗДОК

| Критерій | Howbout | Group Tour | Landmark-touringer |
|---|---|---|---|
| Формування маршруту туристичної поїздки | + | - | + |
| Джерело організації туристичної поїздки | Знайомі, друзі | Агентство | Все коло бажаючих користувачів |
| Пошук перевізника для підтримки поїздки | - | - | + |
| Платформа | Мобільний застосунок | Мобільний застосунок | Вебрішення |
| Обов'язкова ідентифікація за номером телефону | + | + | - |

3

Рисунок В.3 – Порівняння програмних засобів організації колективних туристичних поїздок

## ПОРІВНЯННЯ ФРЕЙМВОРКІВ ДЛЯ РОЗРОБКИ ПРОГРАМИ ОРГАНІЗАЦІЇ КОЛЕКТИВНИХ ТУРИСТИЧНИХ ПОЇЗДОК

| Критерій | FastAPI | Django |
|---|---|---|
| Концепція фреймворку | Мікрофреймворк | Все головне вбудовано |
| Продуктивність | Дуже висока для мікросервісів | Висока для великих рішень |
| Розширюваність | Готових розширень значно менше | Багато підтримуваних розширень |
| Опис, документу-вання, підтримка | Значно нижчі | Високі |

4

Рисунок В.4 – Порівняння фреймворків для розробки програми організації колективних туристичних поїздок

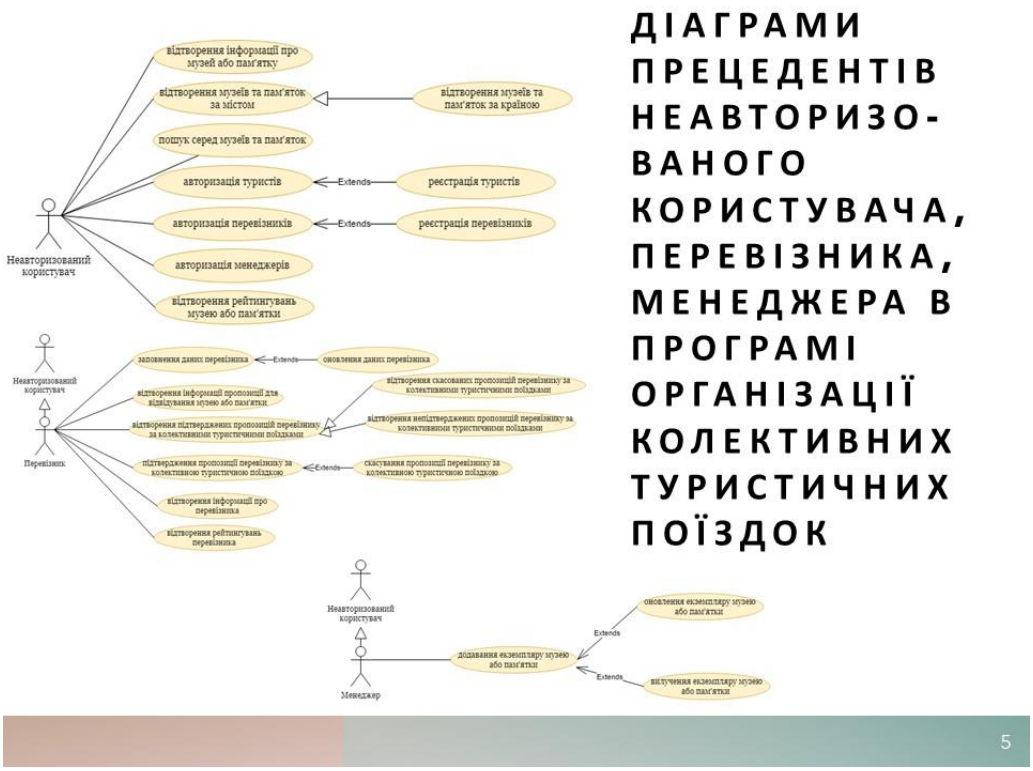


Рисунок В.5 – Діаграми прецедентів неавторизованого користувача, перевізника, менеджера в програмі організації колективних туристичних поїздок

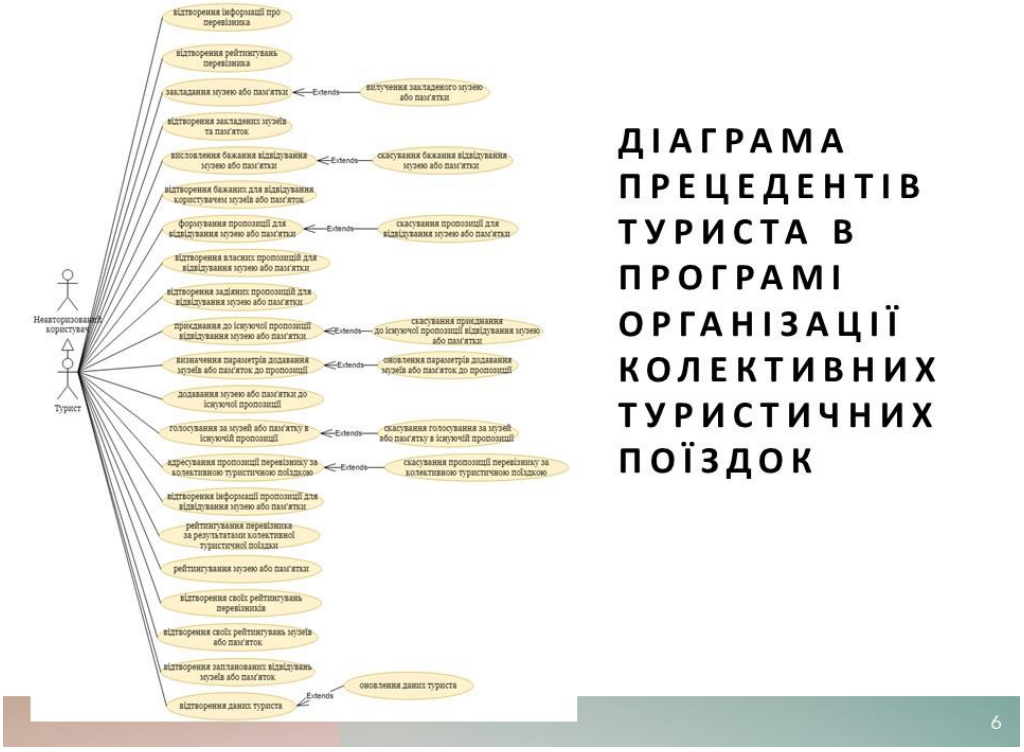


Рисунок В.6 – Діаграма прецедентів туриста в програмі організації колективних туристичних поїздок


Рисунок В.7 – Схема бази даних програми організації колективних туристичних поїздок


Рисунок В.8 – Схема функціонування програми організації колективних туристичних поїздок

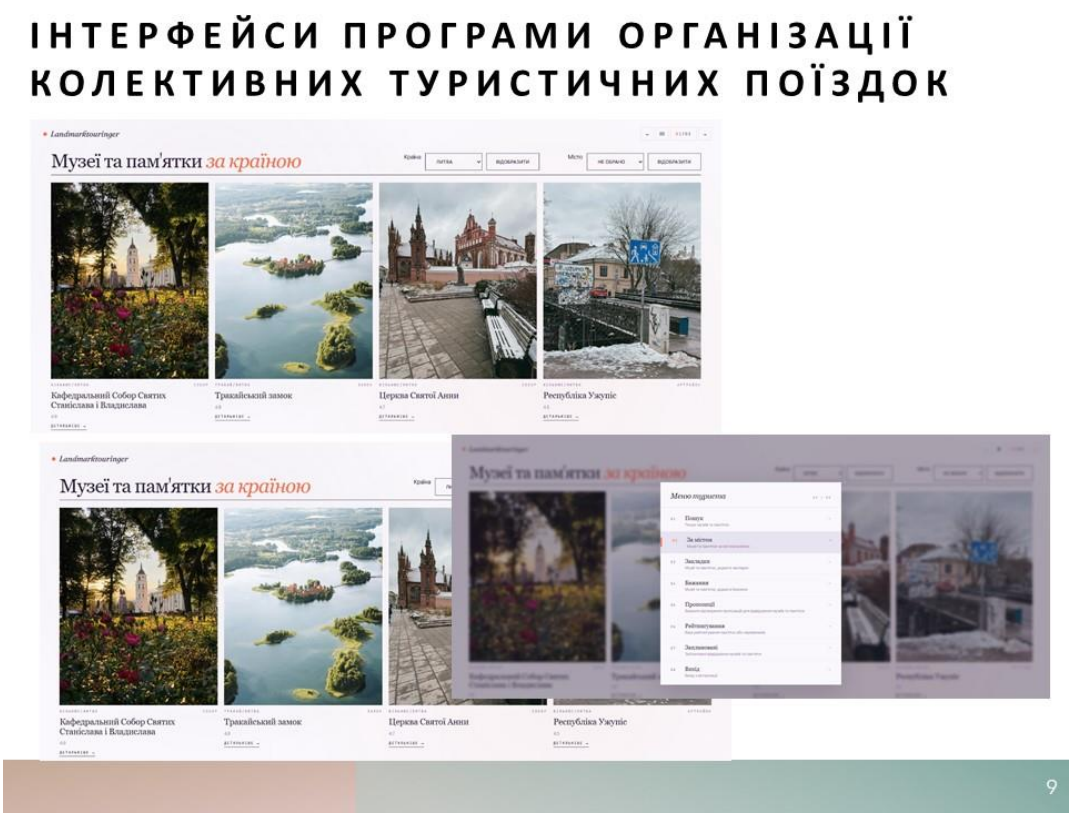


Рисунок В.9 – Інтерфейси програми організації колективних туристичних поїздок

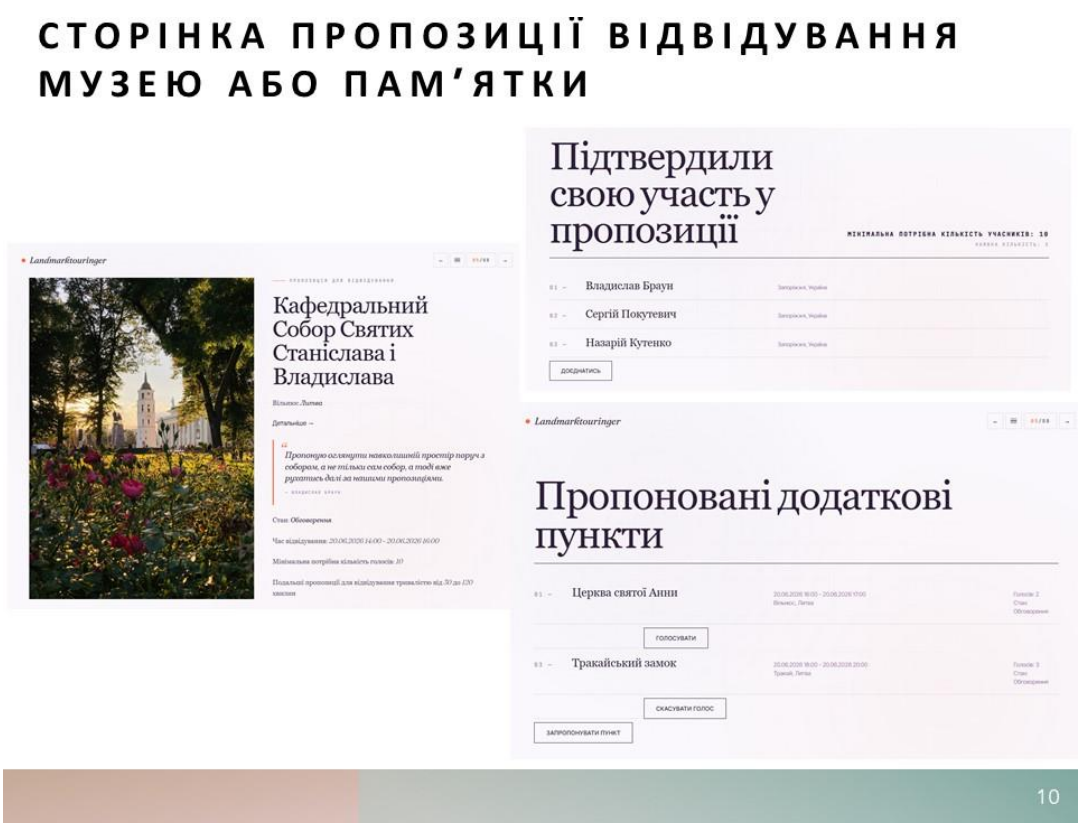


Рисунок В.10 – Сторінка пропозиції відвідування музею або пам’ятки

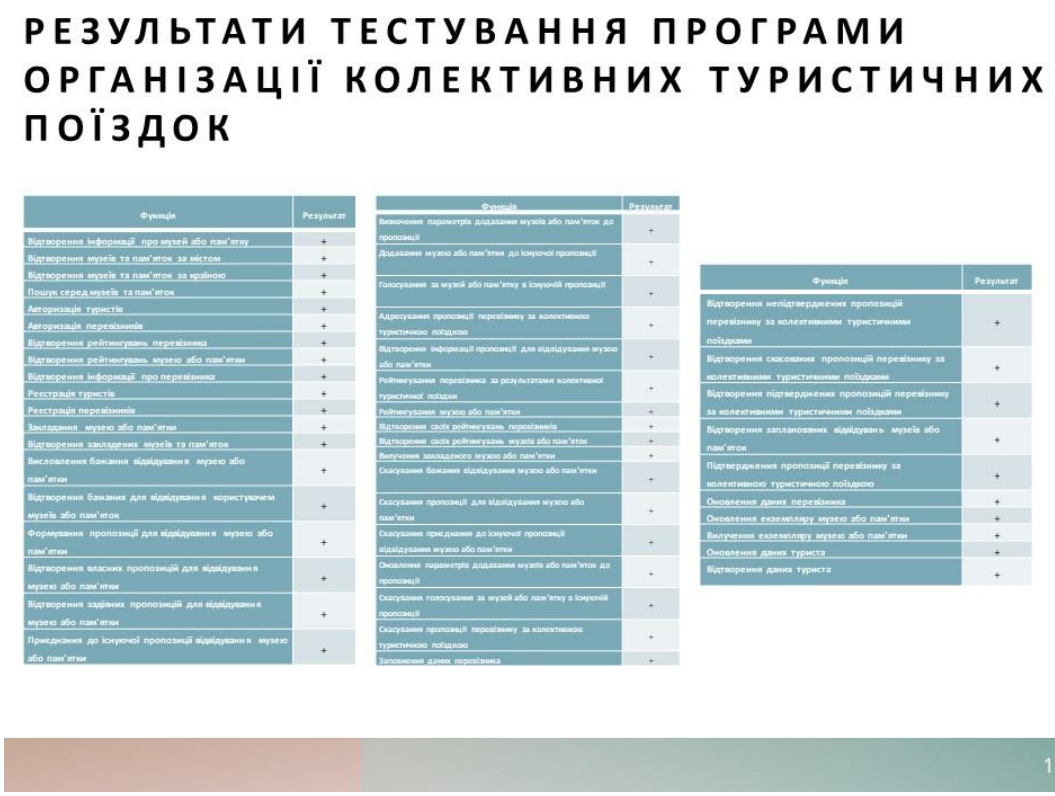


Рисунок В.11 – Результати тестування програми організації колективних туристичних поїздок

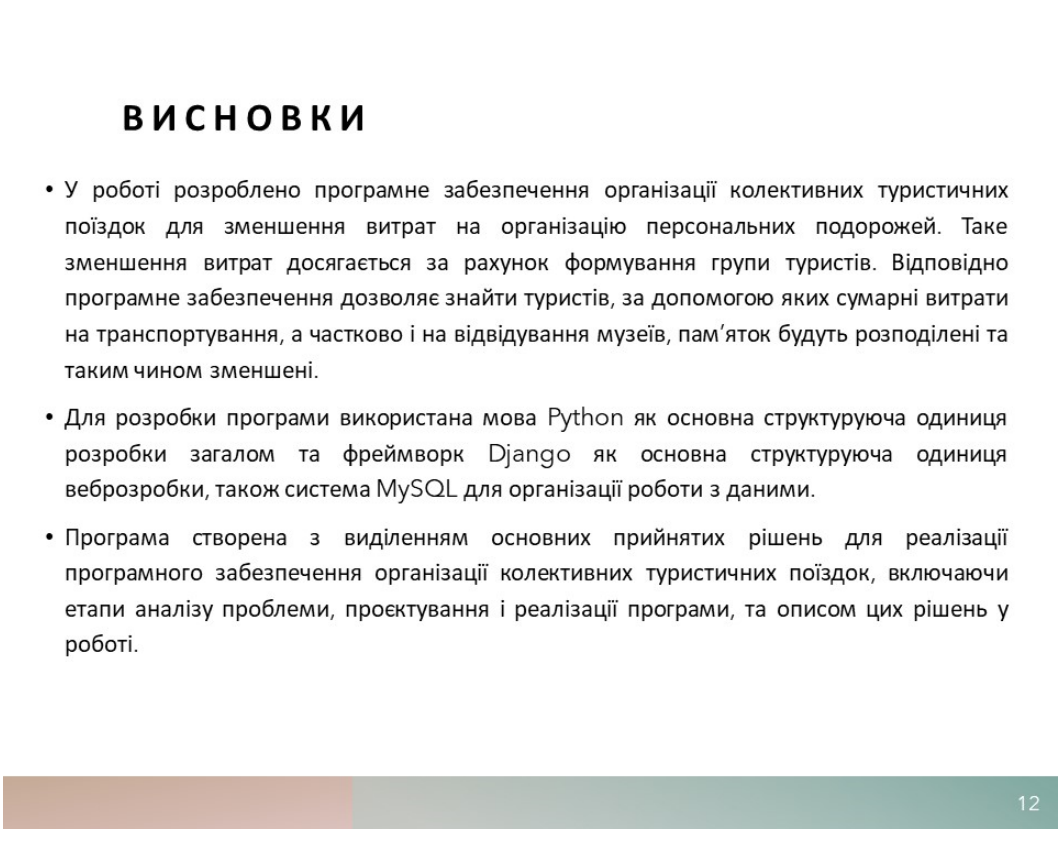


Рисунок В.12 – Висновки