

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ ПЛАНУВАННЯ ПОДОРОЖЕЙ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
TRAVEL PLANNING SUPPORT

Виконав: студент(ка) 2 курсу, групи КНТ-110м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Крупенко А.М.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

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

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)

Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Крупенка Андрія Миколайовича
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
підтримки планування подорожей
Research and Software Implementation of Methods for Travel Planning Support

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

затверджені наказом закладу вищої освіти від “5” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) грудень 2021 року

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

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

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2021 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 та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Крупенко А.М.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
94 с., 30 рис., 10 табл., 3 дод., 17 джерел.

ПЛАНУВАННЯ ПОДОРОЖЕЙ, СТАТТІ ВИТРАТ, МЕТОД K-НАЙБЛИЖЧИХ СУСІДІВ, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ.

Об'єкт дослідження – процес планування подорожей. Предмет дослідження – методи підтримки планування подорожей.

Мета роботи – розробити програмне забезпечення підтримки планування подорожей для забезпечення організації подорожей в міста, відвідувані вперше.

Матеріали, методи та технічні засоби: мова програмування PHP, система керування базами даних MySQL, вебсервер Apache, метод k-найближчих сусідів.

Результати. Розроблене програмне забезпечення і представлений метод, який лежить в основі його роботи, є основними результатами роботи. Наукова новизна роботи полягає в запропонованому методі підтримки планування подорожей, який виконує визначення міст в якості альтернативних варіантів для подорожі на основі методу k-найближчих сусідів з врахуванням уподобань інших користувачів і віддаленості такого уподобання в часі, виконує фінансове планування подорожі шляхом визначення плану на основі статей витрат, суми за якими обчислюються шляхом усереднення значень, отриманих іншими користувачами, що дозволяє самостійно організовувати подорожі в міста, відвідувані вперше.

Висновки. Проведене експериментальне дослідження підтвердило можливість використання представленого методу для визначення майбутнього міста для подорожі.

Галузь використання – туристична галузь, підтримка індивідуальних подорожей в окремі міста світу.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 94 pages, 30 figures, 10 tables, 3 appendixes, 17 sources.

TRAVEL PLANNING, COSTS, K-NEAREST NEIGHBOR METHOD, MACHINE LEARNING, CLASSIFICATION.

The object of research is the process of travel planning. The subject of research is methods to support travel planning.

The purpose of the work is to develop travel planning support software to ensure the organization of trips to the cities visited for the first time.

Materials, methods and technical means: PHP programming language, MySQL database management system, Apache web server, k-nearest neighbor method.

Results. The developed software and the presented method, which is the basis of its work, are the main results of the work. The scientific novelty of the work lies in the proposed method of travel planning support, which proposes cities as alternatives for traveling based on the method of k-nearest neighbors, taking into account preferences of other users and the remoteness of such preferences in time, performs financial travel planning calculating costs by averaging the values obtained by other users, which allows to organize independent trips to the cities visited for the first time.

Conclusions. The experimental study confirmed the possibility of using the presented method to propose cities for the next trips.

Field of use. The tourism industry. Support for individual travel to the cities around the world (one city at a time).

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Огляд програмних засобів підтримки планування подорожей	11
1.2 Визначення концепції розроблюваного програмного забезпечення	15
1.3 Висновки за розділом 1	16
2 Матеріали і методи	17
2.1 Колаборативна фільтрація	17
2.2 Метод k-найближчих сусідів	19
2.3 Метод підтримки планування подорожей	20
2.4 Висновки за розділом 2	23
3 Проєктування програмного забезпечення	24
3.1 Вибір мови програмування	24
3.2 Функціональні вимоги до програмного забезпечення	26
3.3 Проєктування бази даних	30
3.4 Структура програмного забезпечення	36
3.5 Висновки за розділом 3	40
4 Основні рішення щодо реалізації компонентів системи	41
4.1 Функціональна схема програмної системи	41
4.2 Опис модулів реалізованої програмної системи	43
4.3 Висновки за розділом 4	49
5 Експлуатація, тестування та експериментальне дослідження програми	50
5.1 Призначення програми	50
5.2 Умови виконання програми	50
5.3 Виконання програми	50
5.4 Експериментальне дослідження проблеми підтримки планування подорожей	56
5.5 Висновки за розділом 5	57

Висновки.....	58
Перелік джерел посилання	59
Додаток А Технічне завдання.....	61
Додаток Б Текст програми.....	65
Додаток В Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVC – Model-View-Controller;

БД – база даних.

ВСТУП

Інструменти планування подорожей беруть своє джерело в інструментах контролю власних фінансів. При цьому планування подорожей вимагає специфічних статей витрат. Однак варто враховувати, що при цьому планування подорожей має деяку специфіку. Вона пов'язана з тим, що такого програмного забезпечення вимагають досить різні групи людей: туристи, люди, які знаходяться у відрядженнях, і так звані цифрові кочівники, тобто люди, які працюють в онлайн-бізнесі. Зрозуміло, що відрядження відбуваються у наперед визначені міста, а ось туристи і цифрові кочівники мають визначити місто, в яке вони бажають поїхати, що є безпосередньою частиною планування подорожі. Тому якісна підтримка даного процесу є важливим етапом процесу планування.

Актуальність теми пов'язана з тим, як активно розвивається туристична галузь у світі: у період до пандемії туризм кожного року зростав, за даними Всесвітньої туристичної організації у 2018 році була досягнута рекордна кількість у 1,4 мільярда міжнародних туристів, що на 6% більше, ніж у 2017 році, загалом туризм відіграє все більш важливу роль у світовій економіці, при цьому в 1950 році було лише 25 мільйонів міжнародних туристичних візитів, 166 мільйонів у 1970 році та 435 мільйонів у 1990 році, що говорить про темпи розвитку [1]. Відповідно виникає потреба в розробці відповідного програмного забезпечення для підтримки даного процесу.

Оскільки таким програмним забезпеченням будуть користуватися туристи і цифрові кочівники, то частиною завдання з їх підтримки для збільшення залучення їх в роботу з програмою є завдання допомоги у визначенні міста подорожі. Тому важливе впровадження широкої системи оцінки міст, що загалом за рахунок цієї інформації, за рахунок пошуку подібних виборів міст іншими користувачами та наявності засобів перегляду середньої статистики витрат у кожному місті, що потрібно під час створення плану подорожі. Плануючи подорожі найчастіше (в міста, в яких людина ще не була), людина не має точної інформації про рівень витрат у місті, тому

наявність такого програмного інструмента є важливим практичним рішенням для підвищення зручності їх роботи. У такому разі не потрібно шукати інформацію на окремих спеціалізованих вебсайтах, писати повідомлення в соціальних мережах, на форумах з запитаннями про рівень витрат, а одразу можна отримати конкретні значення на основі досвіду інших користувачів. При цьому фактичні витрати кожного конкретного користувача нікому не надаються, а залишаються конфіденційними. Вони присутні для інших користувачів тільки у вигляді статистичних результатів шляхом усереднення досвіду всіх користувачів, які побували в цьому місті.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд програмних засобів підтримки планування подорожей

Розглянуті в даному огляді програмні засоби не завжди підтримують саме процес планування безпосередньо, вони можуть застосовуватися для отримання даних, необхідних під час планування. Одним з таких застосунків є вебсайт nomadlist.com (рис. 1.1) [2].

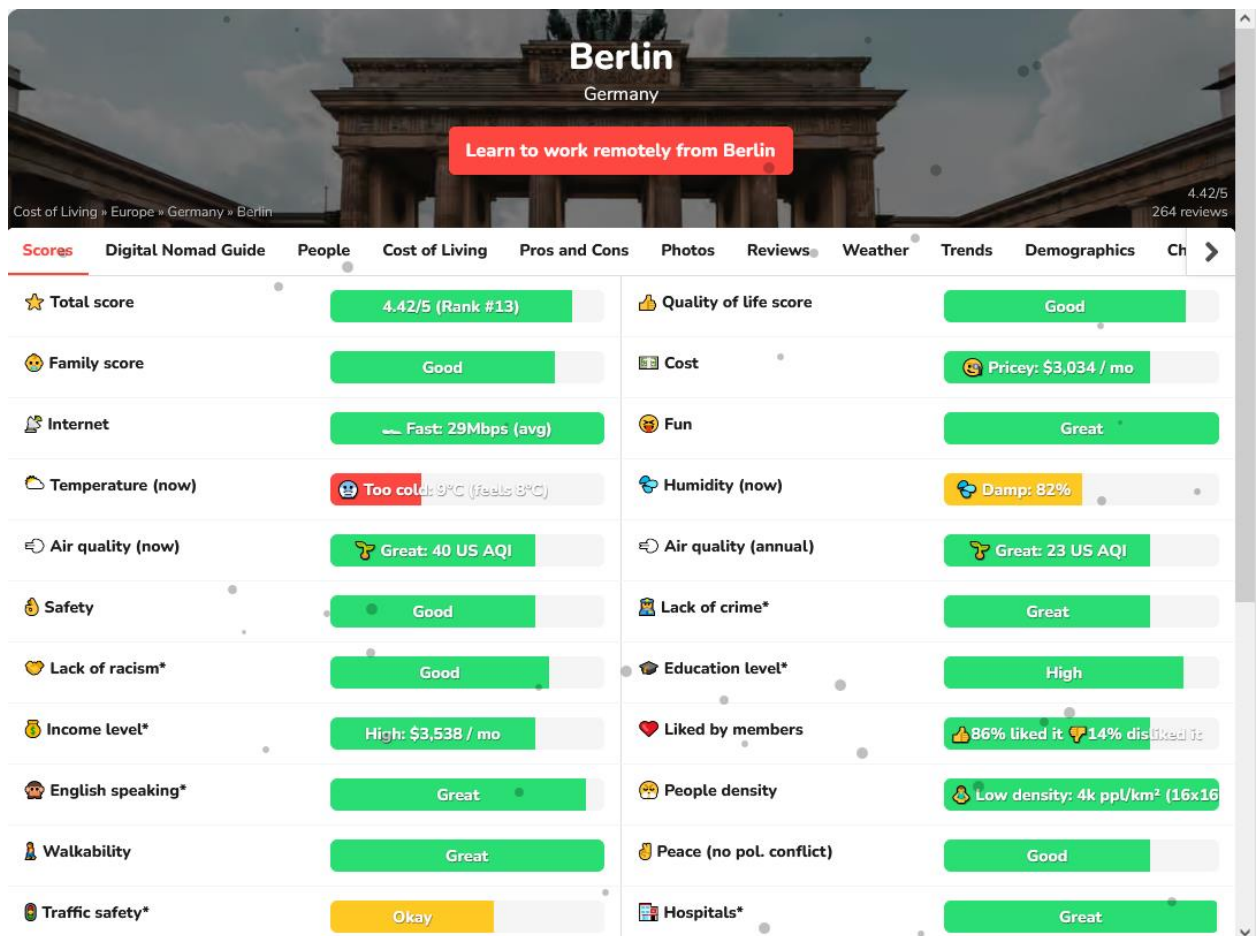


Рисунок 1.1 – Перегляд оцінок міста на вебсайті nomadlist.com

Даний вебсайт призначений для цифрових кочівників, а тому в інформації про міста надаються дані щодо тривалого проживання в даному місті і особливостей роботи в цьому місті. З різних сторін на даному вебсайті представляється інформація про міста. Ця інформація включає середні оцінки за різними параметрами:

- сумарна оцінка;
- оцінка для сімей;
- оцінка якості інтернету;
- температура, туманність і якість повітря в місті на поточний момент та річний рівень якості повітря;
- оцінка безпеки в місті;
- відсутність расизму;
- рівень доходів;
- рівень володіння англійською мовою;
- придатність до прогулянок пішки;
- безпечність трафіку;
- рівень щастя;
- вільний wi-fi в місті;
- якість роботи системи підігріву або охолодження;
- свобода слова;
- якість життя;
- рівень витрат;
- оцінка розваг;
- відсутність кримінальних випадків;
- рівень освіти;
- частка уподобань учасниками;
- щільність населення;
- відсутність військових конфліктів;
- якість лікарень;
- рівень нічного життя;
- наявність місць для роботи;
- дружність до іноземців;
- дружність до жінок;
- оцінка стартапу.

При чому велика кількість цих показників представлена на основі статистичних даних цієї країни, тобто не з обрахунків оцінок користувачів.

Користувачу надається можливість переглянути перелік кращих міст, країн (рис. 1.2), виконати пошук потрібної країни або міста.

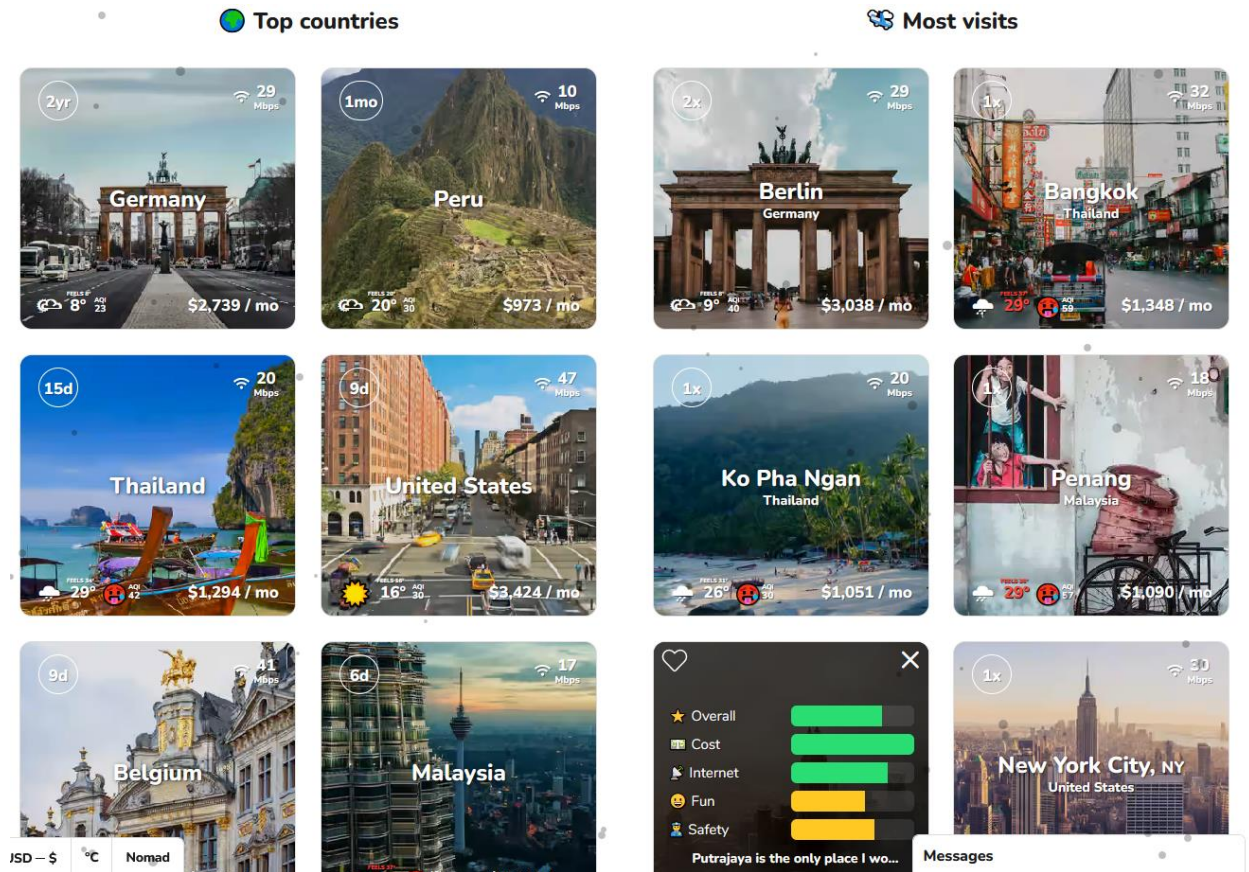


Рисунок 1.2 – Перегляд оцінок міста на вебсайті nomadlist.com

За кожним містом можна переглянути окрім перелічених вище оцінок, статистику витрат за невеликою кількістю показників, людей, які віддалено працювали з цього міста, загальну інформацію про місто, фотографії, відгуки. Окрім того кожен користувач може оцінити власну подорож, а потім за кожним користувачем можна переглянути його оцінки містам, подорожам. Кожна подорож виконується тільки в одне місто.

Оцінки визначаються користувачем загалом, за витратами, за інтернет, за розваги та за рівень безпеки.

Вебсайт numbeo.com (рис. 1.3) призначений для надання інформації про достатньо широку статистику витрат у місті. Усі витрати об'єднані в певні групи [3].

Користувач може виконати пошук міст, а потім переглянути статистику за містом, визначаючи при цьому іноземну валюту, в якій будуть виведені дані. Окрім того користувач може внести власні дані про вартість за кожною статтею.

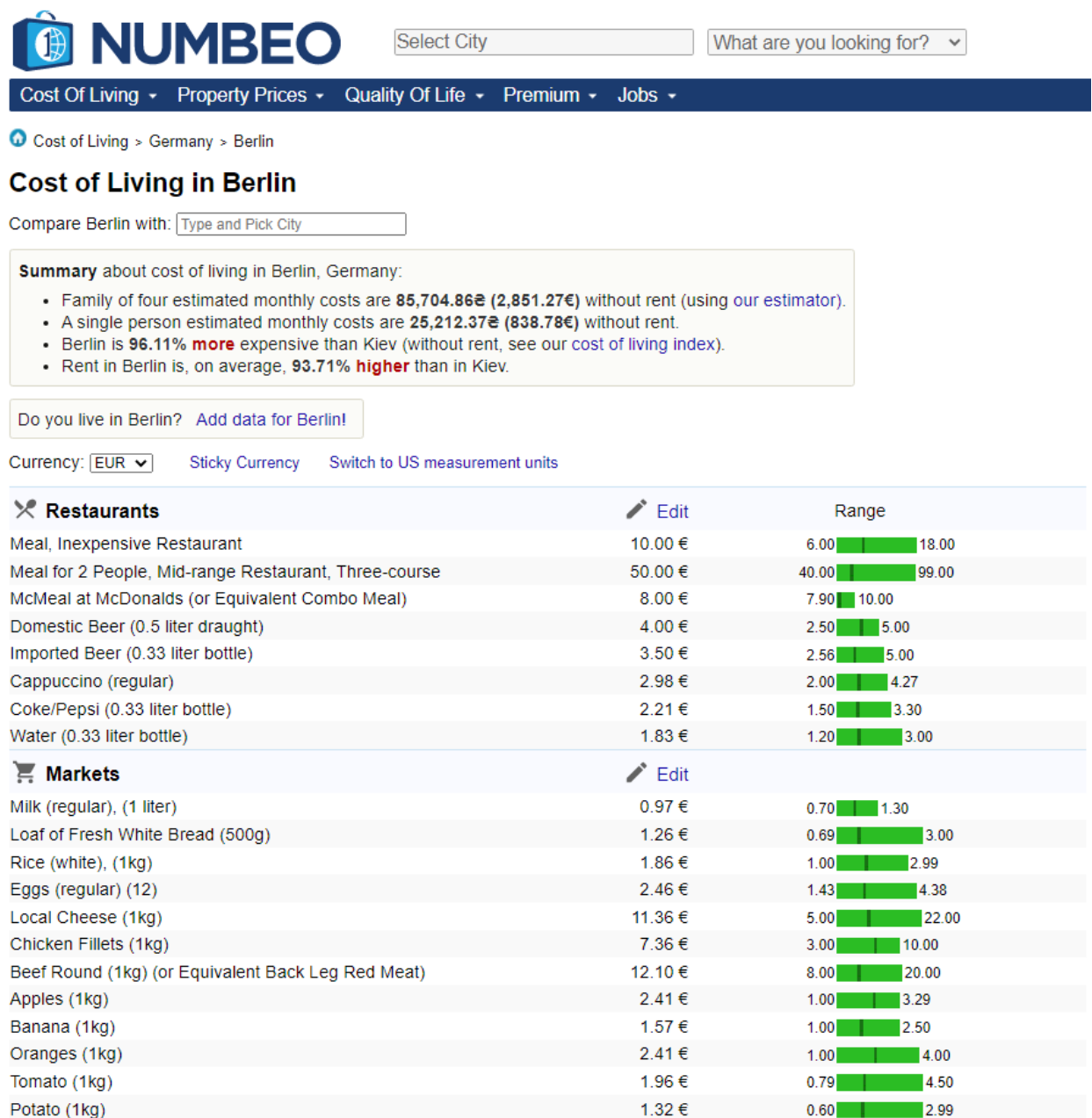


Рисунок 1.3 – Перегляд статистичних витрат у місті на вебсайті numbeo.com

1.2 Визначення концепції розроблюваного програмного забезпечення

Виконувана розробка має знаходитися на межі між розглянутими програмними аналогами, але і мати власні особливості. Основна її особливість – наявність засобів для планування подорожі. Адже жоден з програмних аналогів не надає можливості виконувати внесення даних за статтями та організовувати з цього план, обчислювати сумарні витрати. Проте вони дозволяють переглядати потрібну для цього інформацію (хоча в різній мірі) про статистику витрат за статтями. Це важливо для створення такого інструменту планування. Окрім того Nomadlist надає можливості оцінювання міст, в яких побувала людина, на основі цих даних можна створювати пропозиції щодо подальших подорожей, але це не реалізовано в Nomadlist.

У табл. 1.1 представлено результати порівняння програмних аналогів з розроблюваним рішенням.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	Nomadlist	Numbeo	Власна розробка
Планування подорожі	Немає	Немає	Так
Визначення пропозиції для майбутньої подорожі	Немає	Немає	Так
Визначення статистики витрат	Обмежене	Широке	Широке
Оцінювання міст	Так	Немає	Так

1.3 Висновки за розділом 1

Виконаний огляд програмних аналогів дозволив виявити, що на даний момент є програмні рішення, направлені на отримання інформації про статистику витрат в містах, а також оцінювання міст, але при цьому відчувається нестача програмних засобів планування і, як результат накопичених даних, визначення пропозиції для майбутньої подорожі. Саме це і винесено в якості основних особливостей розробки, яка виконується в роботі.

Завдання дипломної роботи:

- виконати аналіз програмних засобів підтримки планування подорожей;
- виконати проєктування програмного забезпечення;
- реалізувати програмне забезпечення;
- створити новий метод планування подорожей для розв’язання проблеми, що розглядається;
- виконати експериментальне дослідження проблеми.

2 МАТЕРІАЛИ І МЕТОДИ

Оскільки важливим етапом процесу планування є визначення міста, в яке дана подорож відбувається, то спочатку потрібно обрати інструмент, за допомогою якого таке рішення щодо пропозиції міст для подорожі буде прийматися.

Виходячи з розглянутих існуючих характеристик у даній сфері, виставлені оцінки містам можуть бути підґрунтям для такого рішення. Тоді важливо знайти подібні оцінки між різними користувачами і в якості майбутнього рішення виділити той досвід, яким характеризується інший користувач, який відноситься до того ж класу, що і той, для якого задача розглядається.

2.1 Колаборативна фільтрація

Задача, що розглядається, є задачею рекомендації об'єктів, що є підґрунтям для створення рекомендаційної системи.

Рекомендаційна система – це система, яка ґрунтується на підкласі фільтрації інформації, який прагне передбачити оцінку або перевагу, яку користувач надасть об'єкту, такому як продукт, фільм, пісня тощо на основі наступних принципів:

- системи рекомендацій надають персоналізовану інформацію, вивчаючи інтереси користувача через сліди взаємодії з цим користувачем: подібно до алгоритмів машинного навчання, рекомендаційна система робить прогноз на основі минулої поведінки користувача, зокрема вона розроблена, щоб передбачити оцінки або переваги користувачів для набору елементів на основі досвіду;

- математично завдання рекомендації має визначатися набором користувачів U і набором об'єктів I , які слід рекомендувати користувачу U ;

– за результатами навчання необхідно визначати функцію, засновану на даних минулої взаємодії користувача, яка передбачає ймовірність належності об'єкта I користувачу U ;

– усі системи рекомендацій поділяються на два типи на основі даних, які використовуються для створення висновків: фільтрація на основі вмісту, яка використовує атрибути об'єкта, та колаборативна фільтрація, яка використовує поведінку користувача на додаток до атрибутів об'єкта [4].

Колаборативна фільтрація виконує фільтрування інформації, використовуючи дані про взаємодію та дані, зібрані системою від інших користувачів, відзначається наступним:

– заснована на ідеї, що люди, які погодилися в оцінці певних об'єктів, ймовірно, знову погодяться в майбутньому;

– коли потрібно знайти новий об'єкт, часто людина звертається з проханням про рекомендацію до своїх друзів, при цьому звичайно людина більше довіряє рекомендаціям друзів, які поділяють смаки, схожі на наші;

– більшість систем колаборативної фільтрації використовують так звану техніку на основі індексу подібності: у підході на основі сусідства кількість користувачів вибирається на основі їх схожості з активним користувачем, висновок для активного користувача робиться шляхом обчислення середньозваженої оцінки вибраних користувачів;

– системи колаборативної фільтрації зосереджені на відношеннях між користувачами та об'єктами: схожість об'єктів визначається схожістю оцінок цих об'єктів користувачами, які оцінили обидва об'єкти;

– існує два класи колаборативної фільтрації: на основі користувачів, який вимірює схожість між цільовими користувачами та іншими користувачами, на основі об'єктів, який вимірює подібність між об'єктами, на які націлені оцінки користувачів [4]-[5].

2.2 Метод k-найближчих сусідів

Метод k-найближчих сусідів є одним із найбільш відомих методів класифікації і відповідно може використовуватися для розв'язання задачі, що розглядається в даному випадку.

Даний метод ґрунтується на наступних кроках і принципах:

- отримуючи точку, яку потрібно класифікувати, класифікатор знаходить k найближчих точок (найближчих сусідів) із навчальних даних, після цього він призначає мітку класу відповідно до міток класу найближчих сусідів: основна ідея полягає в тому, що якщо певний екземпляр характеризується точкою, що потрапляє в певний район, де переважає мітка деякого класу, то це відбувається тому, що запис, ймовірно, належить саме до цього класу;

- дано точку запити q , для якої потрібно дізнатися її клас l , і навчальну множину даних $X = \{\{x_1, l_1\}, \dots, \{x_n, l_n\}\}$, де x_j – j -й елемент, а l_j – його мітка класу, тоді метод k-найближчих сусідів знайде підмножину $Y = \{\{y_1, l_1\}, \dots, \{y_k, l_k\}\}$ таку, що $Y \in X$, а відстань $\sum_{i=1}^k d(q, y_i)$ мінімальна;

- підмножина Y містить k точок у X , які найближчі до точки запити q , тоді міткою класу $q \in l = f(\{l_1 \dots l_k\})$;

- міра відстані d зазвичай є евклідовою відстанню;

- існують різні кандидати на функцію f , за допомогою якої призначається нова мітка класу: найпоширенішим є правило більшості голосів, коли за більшістю голосів точці запити q присвоюється найпоширеніша мітка її найближчих сусідів, варіант голосування більшості полягає у зважуванні голосів відповідно до відстані між навчальними точками y_i та точкою запити q ;

- окрім того визначення результуючого класу може виконуватися на основі способу, коли голос найближчого сусіда важливіший більше, ніж голос найдалшого – це один із кращих підходів при використанні методу k-найближчих сусідів в умовах колаборативної фільтрації;

– інша стратегія визначення результуючого класу полягає у використанні правила консенсусу: на відміну від правила більшості голосів, консенсус призначає мітку лише тоді й тільки тоді, коли всі k сусідів мають однакову мітку класу [6]-[10].

Метод k -наближчих сусідів є непараметричним алгоритмом, що означає, що він не робить жодних припущень щодо базових даних, його також називають алгоритмом лінивого навчання, оскільки він не вчиться з навчального набору відразу, а зберігає набір даних, а під час класифікації він виконує дії з набором даних: алгоритм k -найближчих сусідів на етапі навчання просто зберігає набір даних, і коли він отримує нові дані, він класифікує ці дані в категорію, яка дуже схожа на нові дані [8]-[10].

2.3 Метод підтримки планування подорожей

Метод підтримки планування подорожей об'єднує в собі два основних концептуальних етапи, перший з яких є передумовою другого. Це базовий потік. У такому випадку спочатку метод має порадити користувачу міста, в які він міг би подорожувати, а далі на основі виконаного вибору користувача відбувається безпосередньо планування витрат подорожі. У випадку, якщо користувач сам чітко знає, яке місто він хоче відвідати, він може не скористатися запропонованим вибором, а обрати самостійно потрібне місто і фактично одразу перейти до другого етапу, не скориставшись результатами виконання першого етапу.

Визначення поради для відвідування має виконуватись на основі методу k -найближчих сусідів, адже він достатньо простий у використанні, надає засоби модифікації під конкретну ситуацію, а окрім того не висуває вимог щодо наявності величезної вибірки даних, що може бути проблемою під час початкового запуску програмного забезпечення в експлуатацію.

Метод підтримки планування подорожей складається з послідовності наступних етапів.

Етап 1. Визначення масиву даних T , що містить оцінки, виставлені кожним користувачем містам з наявного набору, і дати виставлення цих оцінок.

Даний масив на практиці має формуватися періодично. Тобто метод не може виконуватися кожен раз для кожної спроби спланувати подорож одним з користувачів або під час оцінювання кожним користувачем кожної своєї подорожі. Якщо на практиці не притримуватися даного рішення, то це призведе до того, що за великої кількості користувачів і доступної кількості міст потрібно буде для кожного такого випадку виконувати величезну кількість обчислень. У такому випадку фактично метод не зможе використовуватися, адже одна послідовність операцій ще не буде завершуватися на сервері, як вже буде починатися наступна.

Етап 2. Перетворення дати виставлення оцінок на вагову мітку віддаленості за часом від поточного моменту.

Для цього визначається півріччя, до якого відноситься перша виставлена в програмі оцінка, та півріччя, до якого належить поточна дата. Різниця між цими півріччями дорівнює $P - 1$, відповідно перша дата отримує номер півріччя $- 1$, а поточна дата $- P$. Відповідно номери півріч розподіляються послідовно від 1 до P . За таким принципом кожній даті встановлюється номер півріччя p_i , а далі за цим принципом визначається вага оцінки w_i за кожною датою для всіх n оцінок на основі формули (2.1):

$$w_i = \frac{p_i}{\sum_{i=1}^n p_i}. \quad (2.1)$$

Етап 3. Вибір в якості поточного користувача j послідовно кожного користувача з наявних і виконання для кожного з поточних користувачів етапів 4-6, після чого перехід до етапу 7.

Етап 4. Обчислення подібності між оцінками користувача j та оцінками всіх інших користувачів u (значення від 1 до U) окрім j на основі формул (2.2)-(2.3):

$$sim_a(T_j^S, T_u^S) = 1 - \frac{|T_j^S(a) - T_u^S(a)|}{\max(a) - \min(a)}, \quad (2.2)$$

$$SIM(T_j^S, T_u^S) = \sum_{a=1}^n w_a sim_a(T_j^S, T_u^S), \quad (2.3)$$

де $sim_a(T_j^S, T_u^S)$ – подібність між оцінкою T_j^S , виставленою j -м користувачем, і оцінкою T_u^S , виставленою u -м користувачем, місту з номером a ;

$SIM(T_j^S, T_u^S)$ – сумарна подібність між всіма оцінками T_j^S , виставленими j -м користувачем, і всіма оцінками T_u^S , виставленими u -м користувачем, містам;

$T_j^S(a)$ – оцінка, виставлена j -м користувачем місту з номером a ;

$T_u^S(a)$ – оцінка, виставлена u -м користувачем місту з номером a [11].

Етап 5. Визначення для j -го користувача k користувачів, які мають найбільший рівень подібності $SIM(T_j^S, T_u^S)$ до його оцінок.

Етап 6. Виділення з міст з найкращою сумарною оцінкою серед k користувачів.

Для цього мають враховуватися тільки оцінки з рівнем вище або дорівнює 4,5. Кожна така оцінка враховується в сумі для відповідного міста як значення $SIM(T_j^S, T_u^S)$. Після обчислення суми таких оцінок з міст з найбільшою такою сумарною оцінкою виділяються як результат роботи методу для кожного користувача.

Етап 7. Збереження отриманих результатів.

Даний етап необхідний для уникнення повторних обчислень. Метод періодично має виконуватися повторно, результати зберігатися в файлі, після чого під час наступного запуску дані будуть прочитані з файлу. Періодом може бути, наприклад, день.

Етап 8. Вибір міста з представленого набору з міст користувачем.

Етап 9. Обчислення середнього рівня витрат за кожною статтею витрат для заданого міста або витягання відповідних збережених даних з представленням в результаті масиву C .

Етап 10. Визначення плану витрат та обчислення його вартості за значеннями елементів з масиву C .

2.4 Висновки за розділом 2

Проведений аналіз проблеми планування подорожей дозволив звести проблему до проблеми рекомендування міст та питання обчислення витрат на основі статистики минулих подорожей. Для розв'язання проблеми було спочатку розглянуто поняття колаборативної фільтрації, методу k -найближчих сусідів, а далі запропоновано метод підтримки планування подорожей. Даний метод виконує визначення міст в якості альтернативних варіантів для подорожі на основі методу k -найближчих сусідів з врахуванням уподобань інших користувачів і віддаленості такого уподобання в часі, виконує фінансове планування подорожі шляхом визначення плану на основі статей витрат, суми за якими обчислюються шляхом усереднення значень, отриманих іншими користувачами, що дозволяє самостійно організовувати подорожі в міста, відвідувані вперше.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

Одними з найбільш відомих мов програмування для створення вебзастосунків на даний момент є мови програмування PHP та Ruby [12]-[14]. Відомо, що близько половини всіх існуючих на даний момент вебсайтів створено мовою PHP. Тому обґрунтованим є розгляд її як одного з основних варіантів для організації розробки. Однак даний проєкт має окрім веброзробки виконувати ще роботу з даними щодо планування подорожей. Тому ці мови програмування повинні бути розглянуті на основі широкої кількості критеріїв (табл. 3.1).

Таблиця 3.1 – Порівняння мов програмування

Критерій	PHP	Ruby
Швидкість роботи створених проєктів	Вища	Менша
Простота вивчення можливостей	Вища	Менша
Засоби веброзробки	Співмірні	Співмірні
Засоби машинного навчання	Співмірні	Співмірні
Простота роботи з базами даних	Співмірна	Співмірна
Простота запуску на різних платформах	Вища	Менша
Підтримка великої кількості розробників	Вища	Менша

Основними критеріями, які визначали результати порівняння, було обрано:

- швидкість роботи створених проєктів: цей критерій є важливими для майже будь-якого проєкту і для будь-якого проєкту, який передбачає реальне використання з потенційно великою кількістю користувачів;

- простота вивчення можливостей: деякі засоби кожної мови можуть бути невідомими, тому під час реалізації потрібно розширювати свої вміння, а тому важливим є швидке вивчення можливостей;

- засоби веброзробки: зрозуміло, що даний фактор є в основі розробки, адже саме на створення вебзастосунку загалом витрачається в даному проєкті найбільша кількість часу;

- засоби машинного навчання: окрім безпосередньо розробки вебзастосунку потрібно реалізувати представлений у роботі метод, без реалізації якого впровадження концепції рішення неможливо, а тому важлива наявність і зручність засобів машинного навчання;

- простота роботи з базами даних (БД): основою роботи з користувачами є збереження даних між сеансами, що має виконуватися через БД;

- простота запуску на різних платформах: це є важливим для розгортання в майбутньому вебзастосунку на сервері, а також можливості перенесення рішення під час створення;

- підтримка великої кількості розробників: велика кількість розробників визначає, що відповіді на запитання, які виникають під час розробки, можливо буде швидше знайти.

Виконавши порівняння мов програмування для створення розроблюваного програмного забезпечення, було обрано мову PHP, яка має вищу швидкість роботи створених проєктів, більшу простоту вивчення можливостей, простоту запуску на різних платформах та підтримку великої кількості розробників.

3.2 Функціональні вимоги до програмного забезпечення

Проведений аналіз концепції програмного забезпечення і виконаний огляд програмних аналогів дозволив визначити функціональні вимоги до програмного забезпечення, що включають наступні функції:

- створення і планування подорожі;
- редагування параметрів подорожі;
- оцінювання подорожі;
- перегляд власних подорожей користувачем;
- перегляд плану подорожі;
- внесення фактичних витрат подорожі;
- вилучення подорожі;
- перегляд найкращих міст загалом і за думкою конкретного користувача;
- перегляд найкращих подорожей користувача;
- внесення статей витрат;
- внесення груп статей витрат;
- пошук статті витрат;
- перегляд оцінок міста;
- перегляд статей витрат за заданою групою;
- перегляд витрат за статтями у місті;
- визначення статистики витрат за статтями у містах;
- внесення міст, у які можуть здійснюватися подорожі;
- обчислення середніх оцінок за подорожами у місто;
- авторизація і реєстрація користувачів.

Звичайний користувач програмного забезпечення може працювати як авторизований або неавторизований. Окремо для керування забезпечуючими даними та запуску потрібних процесів виділено засоби роботи для адміністратора.

Базовий потік роботи неавторизованого користувача (рис. 3.1):

- перегляд найкращих міст;
- перегляд витрат за статтями у місті;
- перегляд оцінок міста;
- реєстрація користувача.

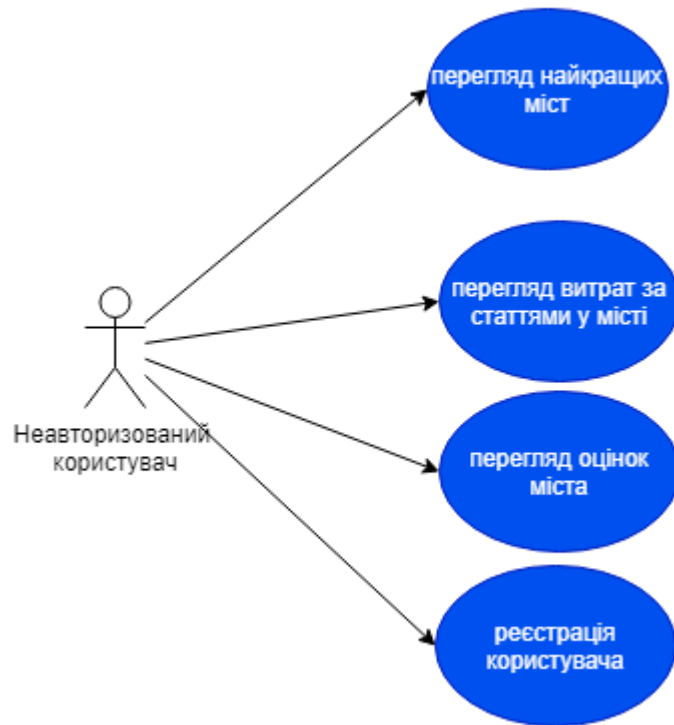


Рисунок 3.1 – Діаграма прецедентів неавторизованого користувача

Базовий потік роботи авторизованого користувача (рис. 3.2):

- планування подорожі;
- оцінювання подорожі;
- перегляд власних подорожей;
- перегляд плану подорожі;
- внесення фактичних витрат подорожі;
- перегляд найкращих міст;
- перегляд оцінок міста;
- перегляд витрат за статтями у місті;
- авторизація користувачів.

Альтернативні потоки:

- вилучення подорожі, який реалізується в тому випадку, коли користувач помилково створив подорож;
- редагування параметрів подорожі, який реалізується у тому випадку, коли користувач змінює заплановану подорож, переносячи її у інше місто або на інші дати.

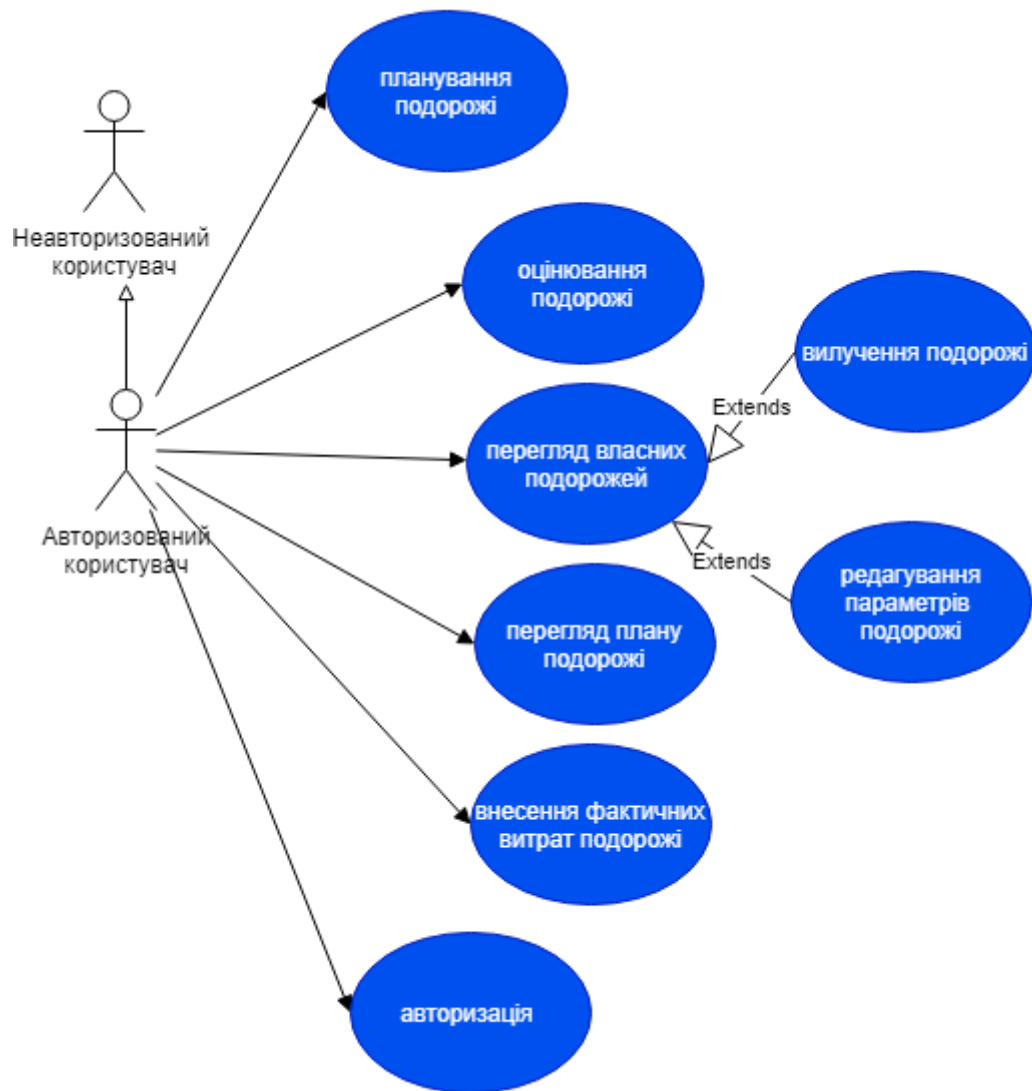


Рисунок 3.2 – Діаграма прецедентів авторизованого користувача

Базовий потік роботи адміністратора (рис. 3.3):

- перегляд статей витрат;
- додавання нових статей витрат;
- перегляд груп статей витрат;
- додавання нових груп статей витрат;

- перегляд міст, у які можуть здійснюватися подорожі;
- додавання нових міст, у які можуть здійснюватися подорожі;
- ініціалізація визначення статистики витрат за статтями у містах;
- ініціалізація обчислення середніх оцінок за подорожами у місто;
- авторизація.



Рисунок 3.3 – Діаграма прецедентів адміністратора

Альтернативні потоки:

- редагування даних міст, що відбувається у випадку перейменування міст, змінення приналежності до певної країни, і реалізується у процесі роботи над потоком перегляду міст;
- редагування статей витрат, що відбувається у випадку змінювання формулювань статей витрат, і реалізується у процесі роботи над потоком перегляду статей витрат;
- редагування груп витрат, що реалізується у випадку змінювання формулювань груп статей витрат, і реалізується у процесі роботи над потоком перегляду груп статей витрат.

3.3 Проєктування бази даних

Для структуризації даних програмного забезпечення було зпроєктовано БД на основі використання системи керування БД MySQL [15].

Таблиця User (табл. 3.2) містить дані про користувача, який здійснює подорожі і контактує з програмою.

Таблиця 3.2 – Структура таблиці User

Но- мер	Назва поля	Тип поля	Опис поля
1	id	int	номер користувача
2	login	varchar(25)	логін користувача
3	password	varchar(25)	пароль користувача
4	admin	bool	адміністративний запис

Таблиця Trip (табл. 3.3) містить дані про конкретну подорож у місто. Кожна подорож може відбуватися тільки в одному конкретному місті.

Таблиця Trip пов'язана через поле cityid з таблицею City.

Таблиця Trip пов'язана через поле userid з таблицею User.

Таблиця 3.3 – Структура таблиці Trip

Но- мер	Назва поля	Тип поля	Опис поля
1	2	3	4
1	id	int	номер подорожі
2	cityid	int	номер відвідуваного міста
3	userid	int	номер користувача
4	begdate	date	дата початку подорожі
5	enddate	date	дата завершення подорожі
6	description	varchar(255)	опис
7	score	tinyint	оцінка, виставлена відвіданому місту
8	safety	tinyint	оцінка за рівень безпеки в місті
9	internet	tinyint	оцінка за якість і швидкість інтернету в місті
10	costs	tinyint	оцінка за рівень витрат в місті
11	fun	tinyint	оцінка розвагам у місті
12	traffic	tinyint	оцінка заторів в місті
13	heating	tinyint	оцінка системи опалення (або охолодження) в місті
14	speechfreedom	tinyint	оцінка свободи слова
15	walkability	tinyint	оцінка зручності міста для прогулянок пішки
16	english	tinyint	оцінка рівня володіння англійською мовою в місті
17	foreigners	tinyint	оцінка сприйняття іноземців у місті
18	nightlife	tinyint	оцінка нічному життю в місті
19	medicine	tinyint	оцінка рівня медичних послуг

Продовження таблиці 3.3

1	2	3	4
20	costtotal	decimal(7, 2)	сумарна фактична вартість подорожі
21	plannedcosttotal	decimal(7, 2)	сумарна запланована вартість подорожі
22	scoredata	date	дата оцінювання подорожі

Таблиця City (табл. 3.4) містить дані про місто, в яке відбуваються подорожі користувачів програми. У таблиці міститься узагальнена інформацію про оцінки користувачів.

Таблиця 3.4 – Структура таблиці City

Но- мер	Назва поля	Тип поля	Опис поля
1	2	3	4
1	id	int	номер міста
2	name	varchar(50)	назва міста
3	country	varchar(3)	код країни
4	overallscore	decimal(4, 2)	сумарна оцінка, виставлена відвіданому місту
5	overallsafety	decimal(4, 2)	сумарна оцінка за рівень безпеки в місті
6	overallinternet	decimal(4, 2)	сумарна оцінка за якість і швидкість інтернету в місті
7	overallcosts	decimal(4, 2)	сумарна оцінка рівня витрат в місті
8	overallfun	decimal(4, 2)	сумарна оцінка розвагам у місті

Продовження таблиці 3.4

1	2	3	4
9	overalltraffic	decimal(4, 2)	сумарна оцінка важкості трафіку в місті
10	overallheating	decimal(4, 2)	сумарна оцінка системи опалення (або охолодження) в місті
11	overallspeech-freedom	decimal(4, 2)	сумарна оцінка свободи слова
12	overallwalkability	decimal(4, 2)	сумарна оцінка зручності міста до прогулянок пішки
13	overallenglish	decimal(4, 2)	сумарна оцінка рівня володіння англійською мовою в місті
14	overallforeigners	decimal(4, 2)	сумарна оцінка сприйняття іноземців у місті
15	overallnightlife	decimal(4, 2)	сумарна оцінка нічному життю в місті
16	overallmedicine	decimal(4, 2)	сумарна оцінка рівня медичних послуг

Таблиця Group (табл. 3.5) містить дані про групу витрат. Вона необхідна для об'єднання статей витрат в загальні категорії.

Таблиця 3.5 – Структура таблиці Group

Но-мер	Назва поля	Тип поля	Опис поля
1	id	int	номер групи витрат
2	name	varchar(50)	назва групи витрат

Таблиця Cost (табл. 3.6) містить дані про статті витрат.

Таблиця Cost пов'язана через поле groupid з таблицею Group.

Таблиця 3.6 – Структура таблиці Cost

Но- мер	Назва поля	Тип поля	Опис поля
1	id	int	номер статті витрат
2	name	varchar(100)	назва статті витрат
3	groupid	int	номер групи витрат
4	quantunit	tinyint	одиниця, за яку в статистиці вимірюється вартість статті витрат

Таблиця CostCity (табл. 3.7) містить дані про статистику витрат за містом, тобто вона дозволяє акумулювати дані за різними подорожами у дане місто.

Таблиця CostCity пов'язана через поле costid з таблицею Cost.

Таблиця CostCity пов'язана через поле cityid з таблицею City.

Таблиця 3.7 – Структура таблиці CostCity

1	id	int	номер запису
2	cityid	int	місто, для якого обчислена статистика за статтею витрат
3	costid	int	середня вартість даної статті витрат
4	price	decimal(7, 2)	статистична середня вартість статті витрат

Схема БД приведена на рис. 3.4.

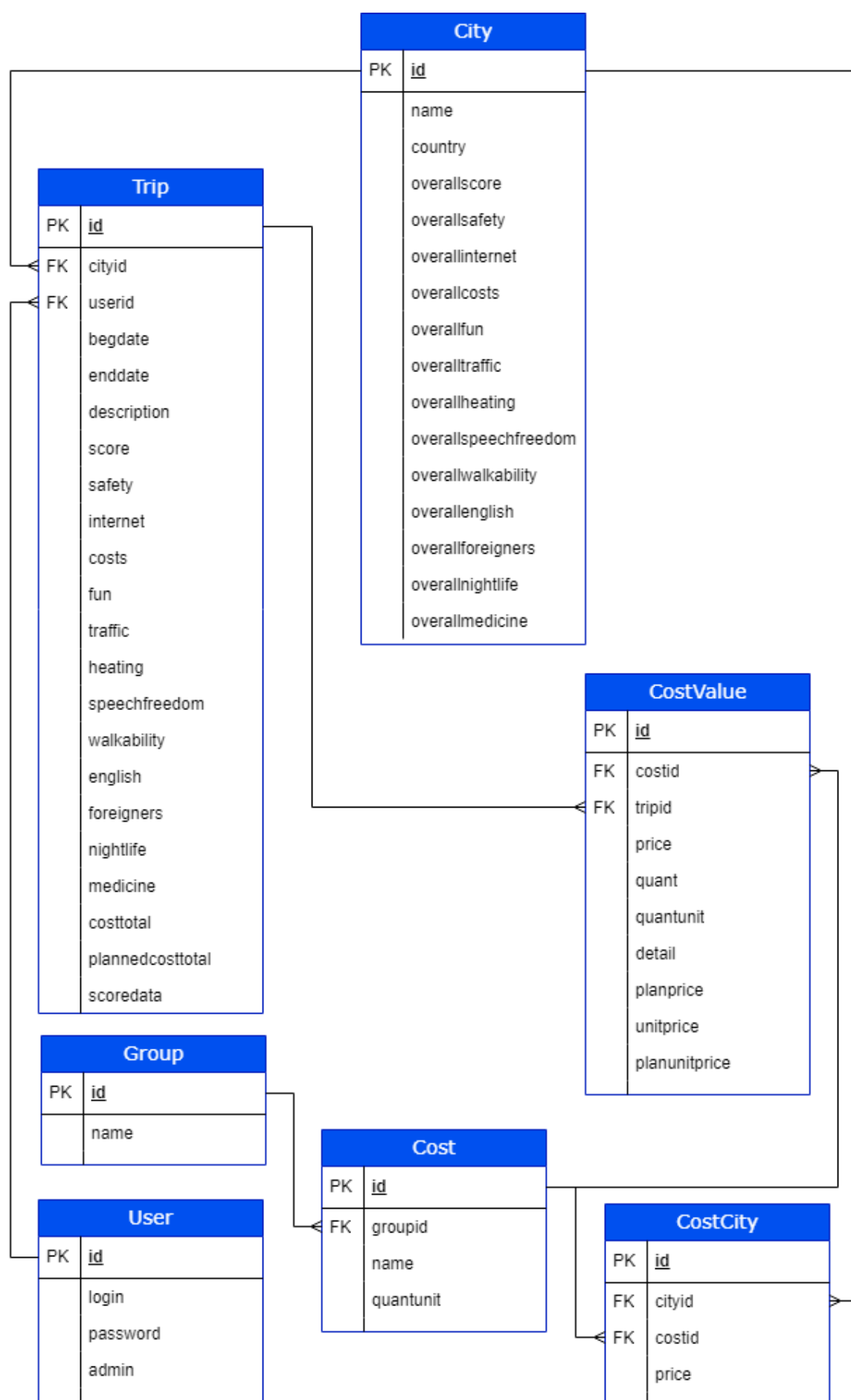


Рисунок 3.4 – Схема бази даних

Таблиця CostValue (табл. 3.8) містить дані про конкретні витрати подорожі.

Таблиця CostValue пов'язана через поле costid з таблицею Cost.

Таблиця CostValue пов'язана через поле tripid з таблицею Trip.

Таблиця 3.8 – Структура таблиці CostValue

Но- мер	Назва поля	Тип поля	Опис поля
1	id	int	номер запису про витрати в подорожі
2	costid	int	номер статті витрат
3	tripid	int	номер подорожі
4	price	decimal(7, 2)	сплачена сума
5	quant	int	кількість одиниць виміру
6	quantunit	tinyint	одиниця виміру витрат
7	detail	varchar(255)	деталізований опис статті витрат
8	planprice	decimal(7, 2)	запланована сума
9	unitprice	decimal(7, 2)	фактична вартість одиниці виміру витрат
10	planunitprice	decimal(7, 2)	запланована вартість одиниці виміру витрат

3.4 Структура програмного забезпечення

Структуру програмної системи було розроблено на основі використання шаблону проєктування Model-View-Controller (MVC) [16].

До складу моделей входять наступні файли:

- файл Trip.php призначений для створення моделі подорожі, що надає засоби роботи над планом подорожі, його витратами;
- файл City.php призначений для створення моделі міста;
- файл Cost.php призначений для створення моделі для роботи з витратами;
- файл Group.php призначений для створення моделі для роботи над групами витрат;
- файл Authorization.php призначений для створення моделі авторизації;
- файл Registration.php призначений для створення моделі реєстрації;
- файл UnAuthorization.php призначений для створення моделі виходу з сесії авторизованого користувача;
- файл Searcher.php призначений для реалізації методу, який був запропонований у розділі 2, в частині визначення пропозиції для подорожі;
- файл DBSlot.php призначений для реалізації з'єднання з БД.

Контролери у відповідності до моделі MVC включають файли:

- файл UserController.php – контролер авторизованого користувача;
- файл GuestController.php – контролер неавторизованого користувача;
- файл AdminController.php – контролер адміністратора;
- файл AuthController.php – контролер авторизації користувача;
- файл UnAuthController.php – контролер виходу з облікового запису користувача;
- файл RegisterController.php – контролер реєстрації облікового запису.

Представлення мають бути реалізовані наступними файлами:

- cityscores.php – представлення сторінки перегляду оцінок, виставлених місту;
- bestcities.php – представлення сторінки перегляду рейтингу міст;
- auth.php – представлення сторінки авторизації облікового запису;

- edittripcosts.php – представлення сторінки редагування плану подорожі;
- tripplan.php – представлення сторінки перегляду статей витрат за планом подорожі;
- usertrips.php – представлення сторінки перегляду власних подорожей користувача;
- plantrip.php – представлення сторінки планування подорожі;
- scoretrip.php – представлення сторінки оцінювання подорожі;
- newtrip.php – представлення сторінки створення нової подорожі;
- unauth.php – представлення сторінки виходу з облікового запису;
- register.php – представлення сторінки реєстрації облікового запису;
- edittrip.php – представлення сторінки редагування подорожі;
- user.php – представлення основної сторінки для роботи авторизованого користувача;
- addcost.php – представлення сторінки додавання нової статті витрат;
- editcost.php – представлення сторінки редагування статті витрат;
- costs.php – представлення сторінки перегляду доступних для планування статей витрат;
- groups.php – представлення сторінки перегляду доступних груп витрат для планування;
- addgroup.php – представлення сторінки додавання групи витрат;
- editgroup.php – представлення сторінки редагування групи витрат;
- admin.php – представлення основної сторінки для роботи адміністратора;
- cities.php – представлення сторінки перегляду доступних міст для планування подорожей;
- addcity.php – представлення сторінки додавання нового міста для подорожей;
- editcity.php – представлення сторінки редагування інформації про місто (рис. 3.5).

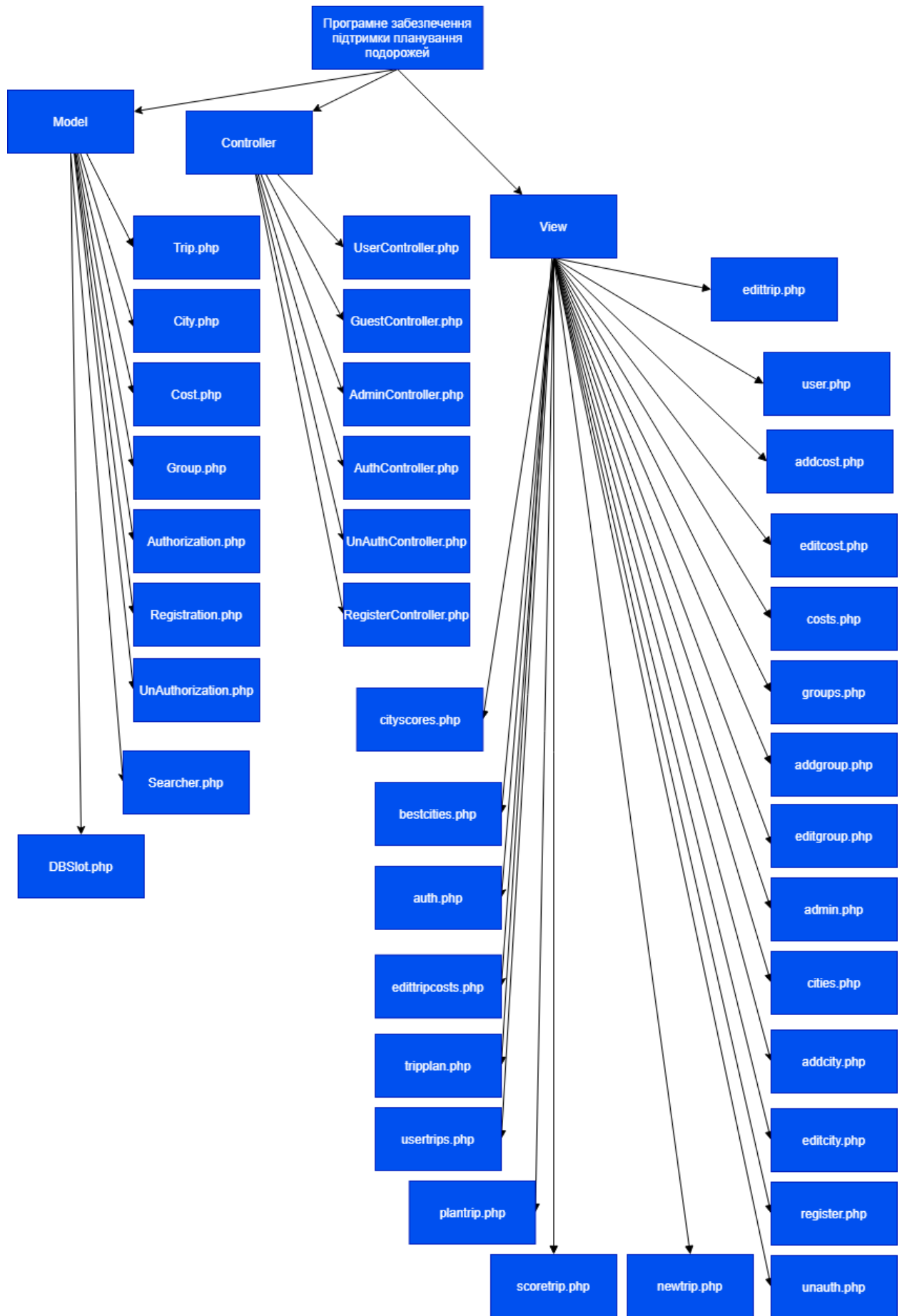


Рисунок 3.5 – Структурна схема програмної системи

3.5 Висновки за розділом 3

Мова програмування PHP обрана для реалізації програмного забезпечення через свої переваги для виконання веброзробки і зручність для машинного навчання в частині реалізації представленого в розділі 2 рішення. Промодельовано взаємодію користувачів з програмним забезпеченням, ґрунтуючись на діаграмах прецедентів. Структуру програмного забезпечення створено у відповідності з шаблоном проєктування MVC.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Функціональна схема програмної системи

Функціонування програмної системи розпочинається з відображення переліку найкращих міст, звідки або за допомогою пошуку користувач може обрати потрібне йому місто і переглянути оцінки даного міста (рис. 4.1).

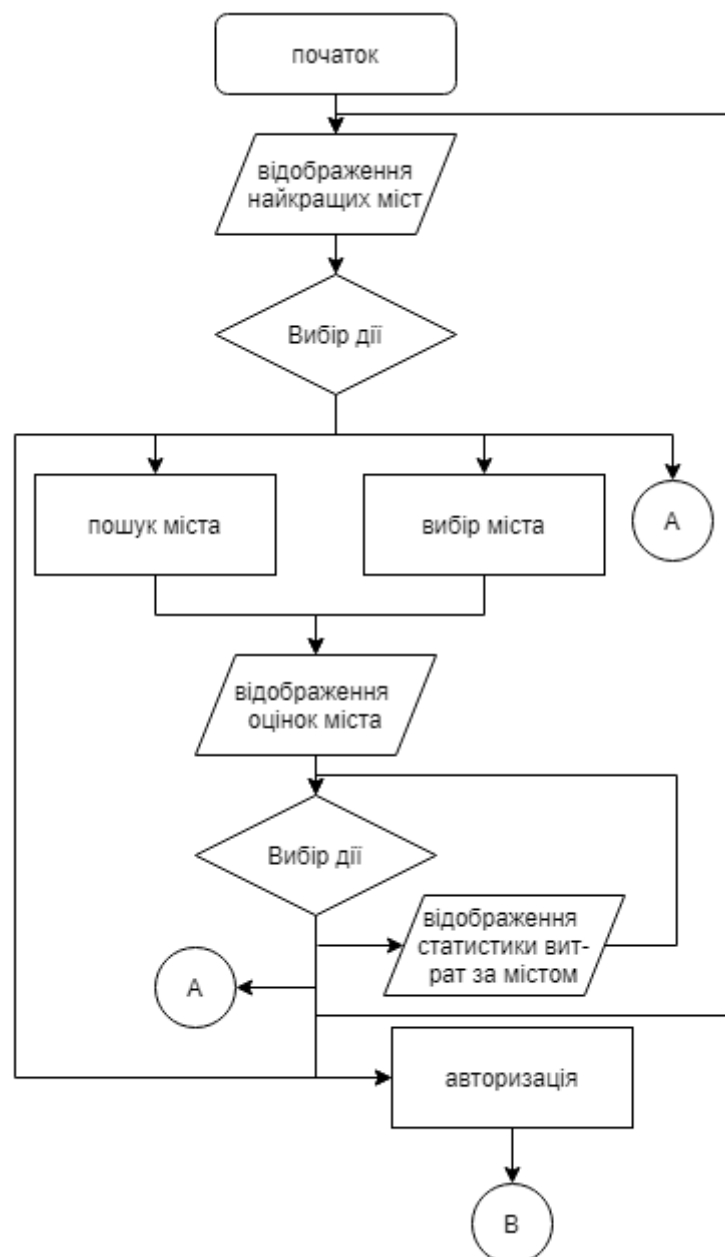


Рисунок 4.1 – Функціональна схема програмної системи: робота без авторизації

Після цього користувач може переглянути статистику за містом за конкретними статтями витрат, повернутися до переліку найкращих міст або виконати авторизацію (рис. 4.2).

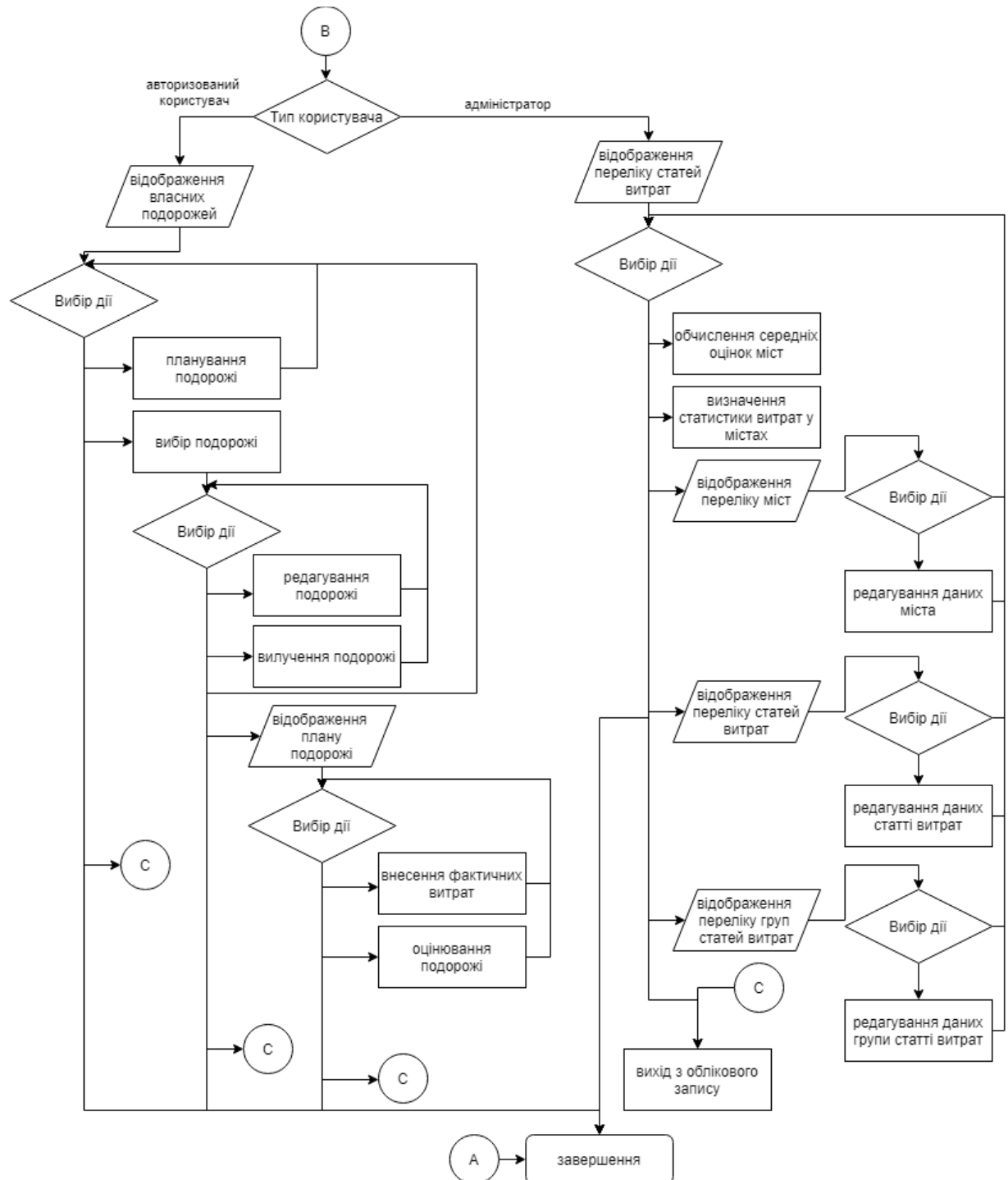


Рисунок 4.2 – Функціональна схема програмної системи: робота після авторизації

Після авторизації користувача виконується вибір, чи було авторизовано звичайного користувача, чи адміністратора. Це визначає подальший потік роботи.

Якщо авторизований звичайний користувач, то йому відображається перелік власних подорожей. Користувач може обрати потрібну подорож і працювати над нею або створити нову подорож і планувати її.

Якщо авторизований адміністратор, то йому відображається перелік статей витрат, після чого він може перейти до переліку груп статей витрат, переліку міст або ініціалізувати обчислення середніх оцінок міст, визначення статистики витрат у містах. За результатами відображення переліку міст, статей витрат або груп статей витрат можна обрати потрібне місто, статтю або групу для редагування.

4.2 Опис модулів реалізованої програмної системи

В силу специфіки шаблону MVC основна функціональність програмної системи реалізована в класах моделей.

Клас `Trip` призначений для надання функціональних засобів, які пов'язані з роботою над подорожами.

Для безпосередньо початкового створення подорожі призначений метод `newTrip ($city, $user, $begdate, $enddate, $description)`, який створює нову подорож, заносючи в БД дані про неї, що включають місто подорожі – `city`, користувача, який цю подорож створив, – `user`, дату початку подорожі – `begdate`, дату завершення подорожі – `enddate`, а також опис подорожі – `description`.

Для початкового редагування подорожі, тобто зміни внесених значень подорожі в пару до методу створення нової подорожі призначений метод редагування параметрів подорожі `editTrip ($id, $city, $begdate, $enddate, $description)`, який змінює в БД дані про подорож з номером `id`, вносячи наступну інформацію: місто подорожі – `city`, дата початку подорожі –

begdate, дата завершення подорожі – enddate, опис подорожі – description. Змінити автора подорожі неможливо. У випадку некоректно створеної подорожі (внесення даних іншим користувачем) подорож потрібно вилучити.

Після завершення подорожі користувач, який її створив, може оцінити цю подорож. Для оцінювання подорожі призначений метод scoreTrip (\$id, \$description, \$score, \$safety, \$internet, \$costs, \$fun, \$traffic, \$heating, \$speechfreedom, \$walkability, \$english, \$foreigners, \$nightlife, \$medicine), який встановлює подорожі з номером id опис description (так можна описати загальне враження про подорож), загальну оцінку – score, оцінку рівня безпеки – safety, оцінку інтернету – internet, оцінку рівня витрат – costs, оцінку розваг – fun, оцінку важкості трафіку в місті – traffic, оцінку системи обігріву або охолодження в місті – heating, оцінку рівня свободи слова – speechfreedom, оцінку зручності міста для піших прогулянок – walkability, оцінку знання жителями англійської мови – english, оцінку ставлення до іноземців – foreigners, оцінку нічного життя – nightlife та оцінку надання медичних послуг – medicine.

Для отримання повного переліку подорожей, в яких брав участь користувач з номером облікового опису id, призначений метод getUserTrips (\$id). Він повертає масив усіх подорожей з повною інформацією по кожній подорожі.

У випадку перегляду повного плану подорожі потрібно отримати дані про весь перелік витрат подорожі, включаючи заплановані та фактичні (після здійснення подорожі) значення. Для виконання цієї функції призначений метод getTripCosts (\$id), який повертає перелік усіх витрат за подорожжю з номером id.

Для формування самого переліку статей витрат, використаних під час планування подорожі, використовується метод addTripCost (\$id, \$cost, \$quant, \$quantunit, \$detail, \$uprice, \$purprice), який додає до переліку витрат подорожі з номером id статті витрат cost, за якою витрачено quant одиниць часу quantunit, при чому кожна одиниця фактично вартує uprice, а заплановано

мала б вартувати – `price`. Для деталізації витрати використовується параметр `detail`, де може міститись інформація про те, на що саме за цією статтею витрат було витрачено кошти. Сумарні витрати за даною статтею заплановані та фактичні розраховуються в методі автоматично на основі множення кількості одиниць на вартість одиниці.

У випадку, якщо необхідно уточнити дані за статтею витрат у подорожі, використовується метод `editTripCost ($id, $cost, $quant, $quantunit, $detail, $price, $price)`, який дозволяє змінити дані статті витрат подорожі `id`, встановлюючи для неї відповідну статтю витрат у `cost`, одиницю виміру витрат – `quantunit`, кількість одиниць – `quant`, деталізований опис витрат – `detail`, фактичну вартість одиниці – `price`, а заплановану вартість одиниці – `price`. Розрахунки сумарної вартості фактичної і запланованої виконуються в методі.

Для видалення статті витрат з номером `id` з подорожі призначений метод `deleteTripCost ($id)`.

Для обчислення сумарних витрат подорожі призначений метод `calculateTripCosts ($id, $mode)`, який виконує підсумовування фактичних витрат за подорожжю з номером `id`, при цьому повертаючи в результаті отримане значення. У випадку, якщо в якості параметра `mode` передане істинне значення, то окрім обчислень ще й виконується оновлення відповідних значень стосовно подорожі в БД.

Для обчислення сумарних запланованих витрат використовується метод `calculatePlannedTripCosts ($id, $mode)`, який обчислює сумарні заплановані витрати за подорожжю з номером `id`, при цьому повертаючи в результаті отримане значення. У випадку, якщо в якості параметра `mode` передане істинне значення, то окрім обчислень ще й виконується оновлення відповідних значень стосовно подорожі в БД.

Для отримання повних даних подорожі з номером `id` призначений метод `getTripById ($id)`.

Для вилучення подорожі з номером `id` призначений метод `deleteTrip ($id)`.

Для отримання даних про відхилення фактичних сум витрат від запланованих призначений метод `getTripCostsDeviation ($id)`. Для цього у повертаємому масиві додається стовпчик з назвою `deviation`, який містить відповідні значення у вигляді відсотків.

Якщо потрібно отримати узагальнені дані про відхилення, тобто співвідношення між сумарними фактичними та запланованими витратами подорожі з номером `id`, можна використати метод `getTripDeviation ($id)`.

Для отримання пропозицій для подорожі на основі представленого методу для користувача з номером облікового запису `user` використовується метод `getPropositions ($user)`.

Формування плану витрат може виконуватися на основі статистичних даних. Для цього потрібний метод `getPlan ($costs, $city)`, який повертає середні значення витрат за переліком статей `costs` у місті `city`.

Для отримання інформації про 10 найкращих міст за середніми оцінками користувача `user` використовується метод `getUserBestCities ($user)`.

Для отримання інформації про 10 найкращих подорожей за середніми оцінками користувача з номером `user` використовується метод `getUserBestTrips ($user)`. Оскільки подорожей у одне місто може бути декілька, то результати даного і попереднього методів можуть відрізнятися.

Клас `Cost` визначає модель статей витрат і пов'язаних з ними засобів.

Безпосередньо додавання нових статей витрат, що належить до функцій адміністратора, належить метод `newCost ($groupid, $name, $quantunit)`, який створює нову статтю витрат з назвою `name`, що буде належати до групи витрат `groupid`, використовуючи для обчислень статистики витрат за цією статтею одиницю виміру `quantunit` (година або день).

Редагування відповідної статті витрат з номером `id` реалізується через метод `editCost ($id, $groupid, $name, $quantunit)`, який співвідносить її з

групою витрат `groupid`, встановлює назву статті в `name`, а в якості одиниці обчислення витрат обирає `quantunit` (бінарне значення для годин та днів).

Визначення переліку всіх статей витрат, які належать до групи з номером `id`, реалізується на основі методу `getCostsByGroup ($id)`.

Для повернення повного переліку доступних статей витрат у програмі призначений метод `getCosts ()`.

Отримати повну інформацію про статтю витрат з номером `id` можна за допомогою методу `getCostById ($id)`.

У випадку необхідності пошуку статті витрат за частиною її назви `name`, підтримку даного процесу може бути реалізовано через метод `searchCost ($name)`.

Метод `getCityCosts ($id)` повертає перелік статистичних, тобто усереднених за всіма подорожами у місто з номером `id`, даних про витрати в цьому місті за всіма статтями витрат.

Переобчислення статистичних даних про витрати у містах реалізується не кожен раз, коли вносяться дані по одній зі статей. Таке рішення призвело би до надто великої кількості потрібних переобчислень у випадку великої кількості користувачів системи. Тому було створено окремий метод для виконання цих дій, який має бути активовано на виконання. І тільки в такому випадку за результатами переобчислень отримані показники будуть внесені в БД. Для цього призначений метод `analyzeCityCosts ()`. Він виконує аналіз за всіма містами і всіма статтями витрат, тобто виконує повне переобчислення. При переобчисленні використовуються два варіанти одиниць обчислення витрат: 0 визначає годину, а 1 – добу. Відповідно для перетворення перший варіант ділиться на 24, а другий – множиться на 24.

Клас `City` реалізує модель міста, в яке виконується подорож.

Внесення адміністратором нового міста з назвою `name` з країни `country` виконується через метод `newCity ($name, $country)`. Назва країни передається у повному вигляді (не скорочено). Відповідно редагування

раніше доданого міста з номером `id` виконується через метод `editCity ($id, $name, $country)` зі зміною назви міста на `name`, а назви країни на `country`.

Метод `getCityIdByName ($name)` повертає номер міста за його назвою `name`, а метод `getCityById ($id)` відповідно навпаки повертає назву і повну інформацію про місто з номером `id`.

За необхідності отримання повного переліку міст використовується метод `getCities ()`. Для кожного міста зазначається його країна з вказівкою повної назви країни.

Для повернення 10 найкращих міст за оцінками всіх користувачів використовується метод `getBestCities ()`.

Переобчислення середніх оцінок за всіма містами за всіма критеріями виконується не кожного разу за оцінювання міста користувачем, а тільки періодично. Для цього застосовується метод `calculateCities ()`.

Назви країн використовуються в інтерфейсі у повній формі, а зберігаються в БД у вигляді коду з 3 символів, для підтримки цих дій реалізовано метод `getCountryCode ($name, mode)`. Він за повною назвою країни `name` повертає її код, якщо значення другого параметра істинне, а для хибного значення навпаки за кодом повертає повну назву країни.

Клас `Group` визначає модель групи витрат.

Безпосередньо додавання нових груп статей витрат виконується через метод `newGroup ($name)`, який створює нову групу статей витрат з назвою `name`.

Редагування відповідної групи статей витрат з номером `id` реалізується через метод `editGroup ($id, $name)`, який змінює назву цієї групи на `name`.

Для забезпечення розмітки і відображення даних, переданих з моделей, було використано шаблон [17], у результаті розроблено відповідну зі структурною схемою кількість представлень застосунку.

4.3 Висновки за розділом 4

Описано порядок функціонування програмної системи на основі відповідних функціональних схем. Представлено основні рішення, які визначають реалізацію класів вебзастосунку.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для надання інформації про витрати в подорожах різними містами світу та планування власних майбутніх і поточних подорожей, а також фіксації фактичних витрат у цих подорожах порівняно з планами.

5.2 Умови виконання програми

Умовами виконання програми є відповідність сервера наступним вимогам:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору;
- наявність доступу до мережі інтернет зі швидкістю не менше 1 Гб/с;
- встановлене програмне забезпечення: PHP версії не нижче 7.3, система керування БД MySQL, вебсервер Apache.

5.3 Виконання програми

Для звернення до програми з метою подальшого використання функціональних можливостей програми потрібно звернутися до вебсервера за адресою, за якою розміщено вебзастосунок. Для цього спочатку адміністратор має розгорнути вебзастосунок і запустити вебсервер.

Після початкового звернення користувачу відображається основна сторінка вебсайту, яка надає доступ до переліку найкращих міст (рис. 5.1). Найкращі міста відображаються в каруселі, якою можна керувати за допомогою кнопок вліво та вправо нижче. Усі міста в каруселі розташовані,

починаючи від міста з найвищим середнім загальним балом. Якщо навести курсор на відповідний елемент з фотографією міста, то з'явиться меню, яке відображає назву міста, країну, в якій місто розташоване, та кнопку “Детальніше” для відкриття сторінки з оцінками міста. У кожному елементі згори в лівому куті виводиться середній бал, виставлений місту всіма користувачами.

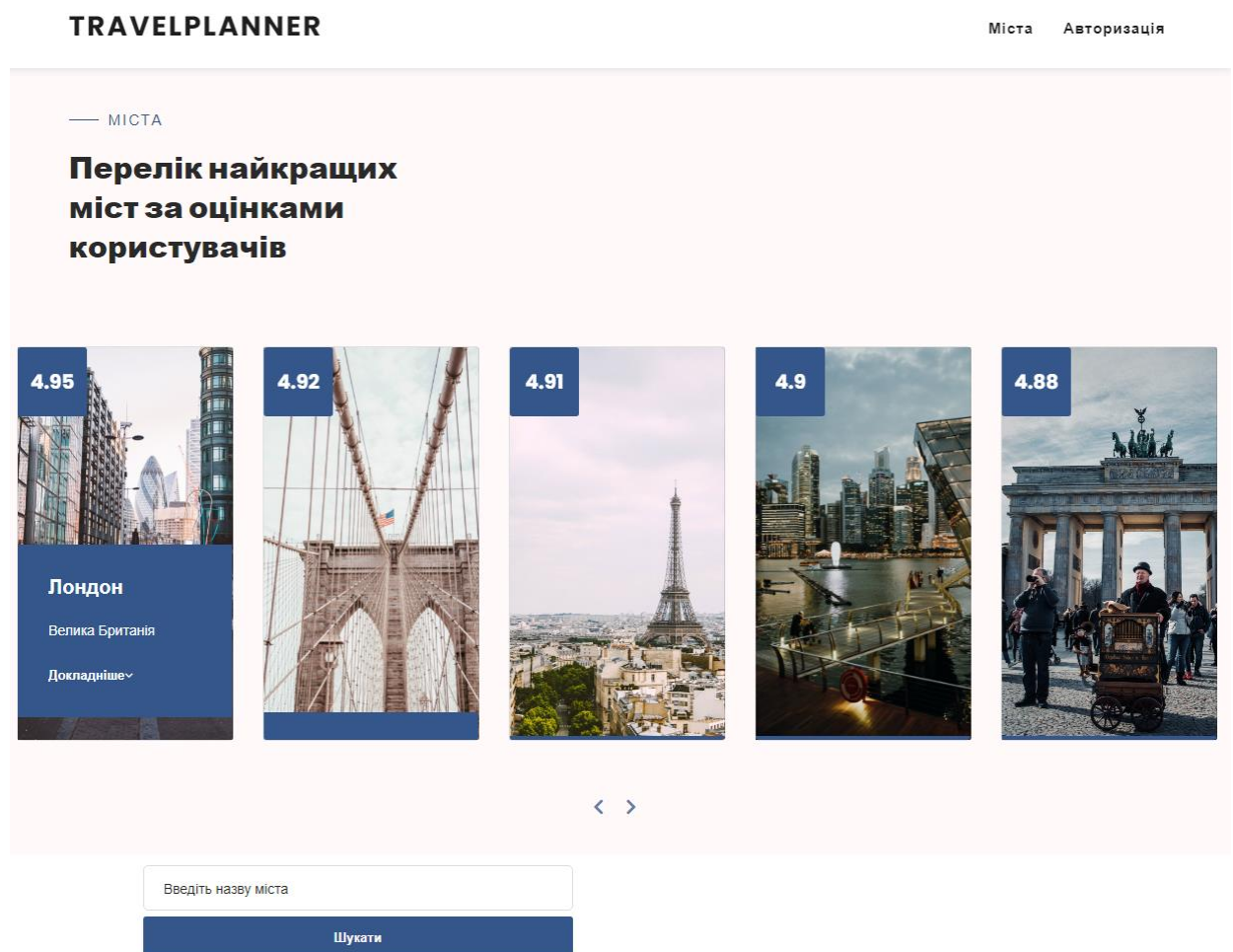


Рисунок 5.1 – Перегляд переліку найкращих міст

Окрім вибору одного з міст з переліку найкращих, можна також ввести в поле нижче назву міста, яке цікавить користувача, та натиснути на кнопку “Шукати”. Це дозволить знайти потрібне місто швидше.

Одноіменний пункт меню дозволяє виконати за потребу авторизацію облікового запису. Це необхідне для планування подорожей зокрема. Якщо

при цьому користувач не має облікового запису, то зі сторінки авторизації він зможе перейти до сторінки реєстрації.

Авторизація дозволяє користувачу отримати розширений вигляд вебзастосунку (рис. 5.2).

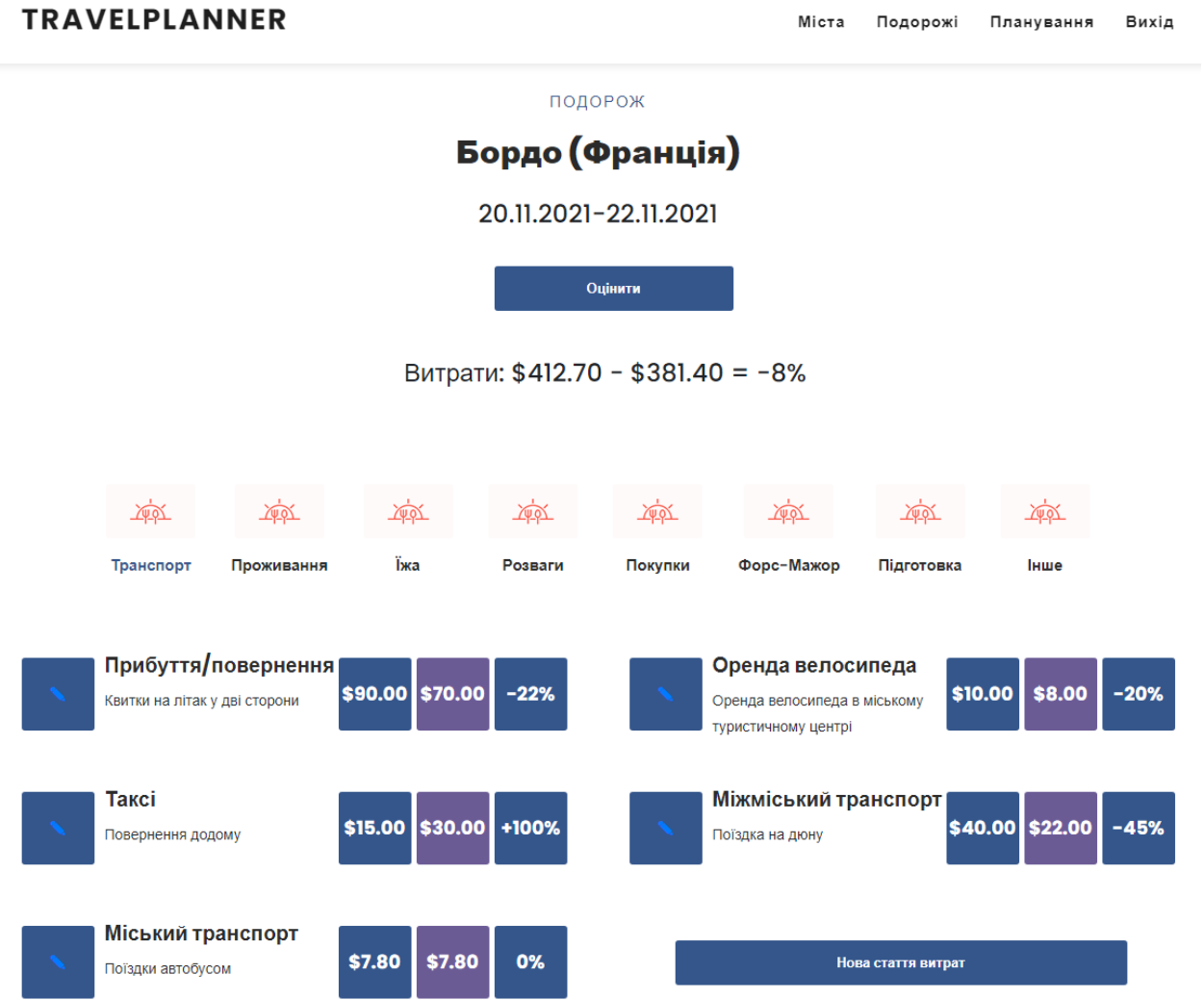


Рисунок 5.2 – Перегляд плану подорожі

У розширеному вигляді користувач може переглянути за пунктом меню “Міста” перелік найкращих, обраних ним міст, а також найкращих загалом міст. Звідси також можна виконати пошук.

Пункт меню “Подорожі” дозволяє отримати перелік найкращих за власною оцінкою його подорожей, а також повний перелік подорожей. Натискаючи на місто і дату потрібної подорожі, можна перейти до її

відповідної сторінки (рис. 5.2). Звідси можна побачити план подорожі. Зверху можна перейти до виставлення оцінки за подорож за допомогою однойменної кнопки. Також там відображається запланована сумарна вартість подорожі, фактична сумарна вартість та різниця у відсотках між даними показниками.

Нижче розташовано перелік іконок, які визначають групи витрат. Якщо обрати одну з доступних, то відобразиться перелік статей витрат за цією групою. За кожною статтею доступна її назва, деталізація, запланована вартість (синій колір), фактична вартість (фіолетовий колір) та різниця у відсотках між цими показниками.

На цій сторінці не відображається інформація про кількість та одиницю витрат. Цю інформацію можна переглянути, натиснувши на кнопку з зображенням олівця. Звідси також доступна можливість редагування статті витрат.

Пункт меню “Планування” (рис. 5.3) дозволяє створити нову подорож, при цьому одразу надається перелік пропозицій міст для подорожі, створених на основі використання представленого в роботі методу. Користувач може або обрати одне з цих міст, або виконати пошук потрібного йому (результати будуть виведені нижче). Для додавання потрібного міста необхідно натиснути на посилання “Відвідати” при наведенні на картинку міста.

Після додавання міста відкривається нова сторінка (рис. 5.4), на якій відображається перелік витрат. Для вибору нової витрати потрібно обрати групу витрат за допомогою іконок, а потім обрати з переліку назву статті витрат, тоді буде відображено вартість статті витрат за статистикою, а далі можна або повторити ці дані, або внести власні дані про вартість одиниці витрат, одиницю витрат, а сумарна вартість буде розрахована автоматично. Після завершення планування потрібно натиснути на кнопку “Зберегти”.

TRAVELPLANNER

Міста Подорожі Планування Вихід

— НОВА ПОДОРОЖ

Пропонуємо

відвідати

4.85

Стокгольм

4.82

4.80

Введіть назву міста

Шукати

Рисунок 5.3 – Вибір напрямку для подорожі

TRAVELPLANNER

Міста Подорожі Планування Вихід

ПОДОРОЖ

Стокгольм (Швеція)

Дата початку

27.11.2021

Дата завершення

30.11.2021

Зберегти

Сумарні витрати: \$90.00

Транспорт

Проживання

Їжа

Розваги

Покупки

Форс-Мажор

Підготовка

Інше

Апартаменти

Зйомна квартира

2

Доба

\$90.00

Стаття витрат:

Готель

Вартість одиниці (рекомендовано - \$150.00)

150.00

Кількість одиниць:

1

Доба

Сумарна вартість за статтею: \$150.00

Внести

Рисунок 5.4 – Створення плану подорожі

Адміністратору надається можливість роботи з даними (рис. 5.5).

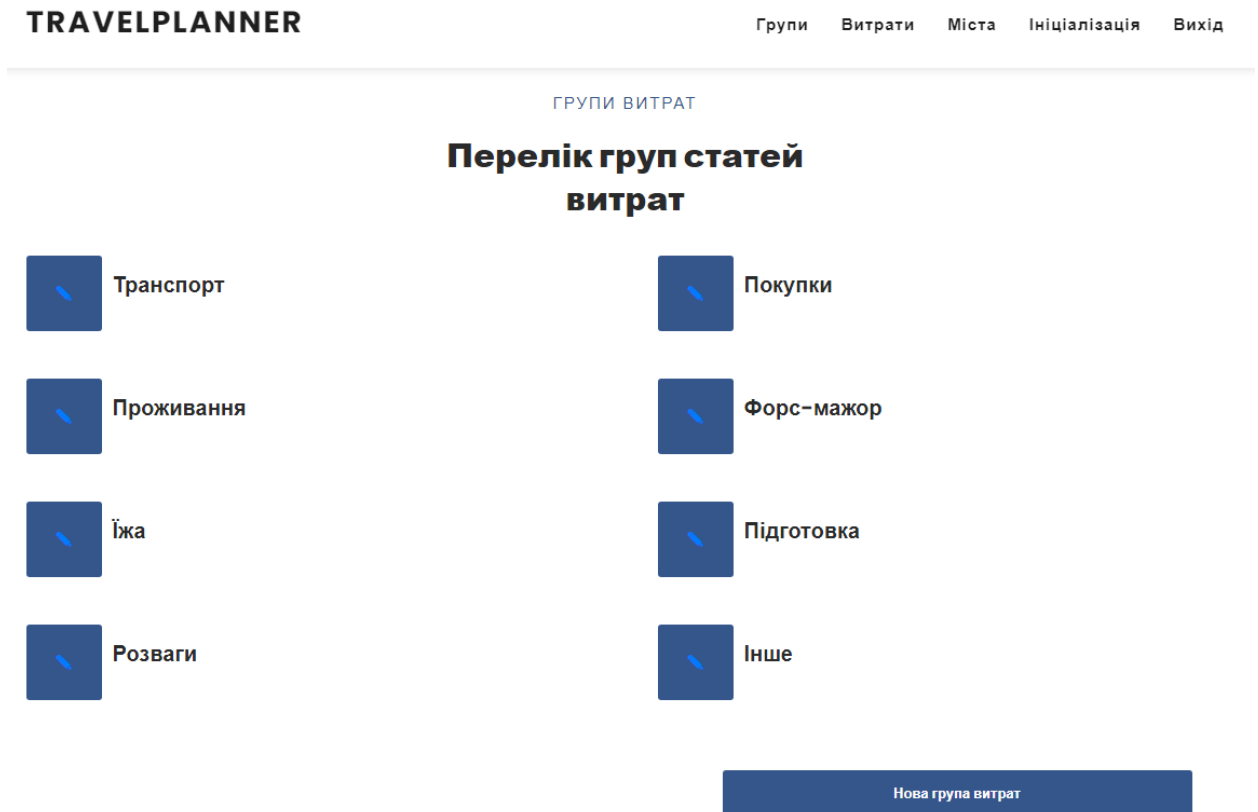


Рисунок 5.5 – Перелік груп статей витрат

З цієї сторінки адміністратор отримує доступ за допомогою пунктів меню до переліку груп, витрат, міст за допомогою однойменних пунктів меню, а також до можливості ініціалізувати перерахунки статистики витрат, оцінок міст і пропозицій для вибору міст для майбутньої подорожі.

За кожним варіантом меню окрім останнього відображається перелік відповідних внесених значень. Можна внести додатковий новий пункт за допомогою відповідної кнопки внизу справа. Можна відредагувати наявний пункт за допомогою олівця. Для міста окрім назви можна задати також його фотографію. Даний файл зберігається на сервері і ідентифікується номером міста. Так само як і результати роботи представленого методу, фотографії не зберігаються в БД. Результати роботи методу містяться в файлі, а потім звідти дістаються за номером користувача.

5.4 Експериментальне дослідження проблеми підтримки планування подорожей

Експериментальне дослідження представленого методу планування подорожей було проведено в частині визначення рекомендованого для наступної подорожі міста. Для цього було виконано парсинг даних з вебсайту [2], який дозволив отримати масив оцінок міст користувачами, який і сформував вибірку даних, над якою було проведено роботу.

Експериментальне дослідження (табл. 5.1) було проведено на основі використання класичної постановки методу k-найближчих сусідів та представленого методу. В класичній постановці використовувалась для обчислення подібності євклідова відстань, при цьому не враховувались часові проміжки, коли була виставлена оцінка, загальне рішення приймалось на основі звичайного складання голосів за кожне місто.

Оцінювання отриманих результатів виконувалось шляхом відділення частини вибірки в 20 %, за кожним екземпляром якої виконувалось співставлення з навчальною вибіркою і виділення рішення про з рекомендованих міст. На вхід було подано не повністю вся послідовність усіх оцінених міст користувачем, а з відкиданням з його останніх оцінок. Для кожного такого екземпляра за кожним з рекомендованих міст було виконано перевірку, чи було це місто фактично оцінено даним користувачем на бали не менші за 4,5. Якщо так, то даний випадок розглядався як випадок правильної класифікації і враховувався як 1, якщо ні, то як 0. Далі всі ці випадки сумувались, отримане значення було розділено на кількість всіх таких випадків загалом, що сформувало показник точності.

Таблиця 5.1 – Результати експериментального дослідження

Метод	Точність
Метод k-найближчих сусідів	0,71
Метод підтримки планування подорожей	0,84

В результаті представлений метод підтримки планування подорожей дозволив отримати точність, обчислену описаним вище способом, у 0,84, що дозволяє застосовувати його для практичного використання. Представлене формулювання методу дозволило покращити точність класифікації на 18,3 %.

5.5 Висновки за розділом 5

Представлено основні можливості програмного забезпечення на основі розгляду відповідних графічних інтерфейсів користувача. Проведено експериментальне дослідження проблеми рекомендування міст для подорожі, в результаті чого представлений метод дозволив отримати точність у 0,84.

ВИСНОВКИ

У роботі розглянуто проблему підтримки планування подорожей. Дану проблему представлено в ракурсі подорожей туристів та цифрових кочівників. Для цього було проаналізовано існуюче програмне забезпечення, в результаті чого було вирішено розробляти рішення, яке ґрунтується на плануванні витрат подорожі і визначення майбутнього міста для подорожі. Розглянуто подорожі, які виконуються в одне конкретне місто.

Наукова новизна роботи полягає в запропонованому методі підтримки планування подорожей, який виконує визначення міст в якості альтернативних варіантів для подорожі на основі методу k-найближчих сусідів з врахуванням уподобань інших користувачів і віддаленості такого уподобання в часі, виконує фінансове планування подорожі шляхом визначення плану на основі статей витрат, суми за якими обчислюються шляхом усереднення значень, отриманих іншими користувачами, що дозволяє самостійно організовувати подорожі в міста, відвідувані вперше.

Розроблене програмне забезпечення дозволяє створювати і планувати подорожі, редагувати параметри подорожі, оцінювати подорож, переглядати власні подорожі, план подорожі, вносити фактичні витрати подорожі, вилучати власні подорожі, переглядати найкращі міста загалом і за думкою конкретного користувача, переглядати власні найкращі подорожі, шукати статті витрат, переглядати оцінки міста, переглядати статті витрат за заданою групою, переглядати витрати за статтями у місті, визначати статистику витрат за статтями у містах, обчислювати середні оцінки за подорожами у місто, а також адміністратору вносити статті витрат, групувати їх, вносити міста, у які можуть здійснюватися подорожі. Програмне забезпечення розроблене мовою програмування PHP на основі взаємодії з системою керування БД MySQL, для якої визначено структуру БД.

Усі завдання, які було поставлено на роботу, виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Blackall, M. Global tourism hits record highs – but who goes where on holiday? [Electronic resource] / M. Blackall. – Access mode : <https://www.theguardian.com/news/2019/jul/01/global-tourism-hits-record-highs-but-who-goes-where-on-holiday>.
2. Nomad List – Best Places to Live for Digital Nomads [Electronic resource]. – Access mode : <https://nomadlist.com/>.
3. Cost of Living [Electronic resource]. – Access mode : <https://www.numbeo.com/cost-of-living/>.
4. Kurama, V. A Simple Introduction to Collaborative Filtering [Electronic resource] / V. Kurama. – Access mode : <https://builtin.com/data-science/collaborative-filtering-recommender-system>.
5. Протоковський, А. Аналіз стратегій рекомендаційних систем знайомств людей та груп за інтересами у віртуальних спільнотах [Текст] / А. Протоковський, Ю. Форкун // Information, Communication, Society (ICS-2020), 21-23 травня 2020 року, Чинадійово. – Львів : НУЛП, 2020. – С. 154-155.
6. Recommender Systems Handbook [Text] / Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor. – Boston, MA : Springer, 2011. – 842 p.
7. Классификатор kNN [Электрон. ресурс]. – Режим доступа : <https://habr.com/ru/post/149693/>.
8. K-Nearest Neighbor (KNN) Algorithm for Machine Learning [Electronic resource]. – Access mode : <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>.
9. Jayaswal, V. K-Nearest Neighbors (KNN) algorithm [Electronic resource] / V. Jayaswal. – Access mode : <https://towardsdatascience.com/k-nearest-neighbors-knn-algorithm-23832490e3f4>.
10. Learning k for kNN Classification [Text] / S. Zhang, X. Li, M. Zong, X. Zhu // ACM Transactions on Intelligent Systems and Technology. – 2017. – Volume 8 (3). – Pp. 1-19.

11. Karabulut, B. A Weighted Similarity Measure for k-Nearest Neighbors Algorithm [Text] / Bergen Karabulut, Guvenc Arslan, Halil Murat Unver // Celal Bayar University Journal of Science. – 2019. – Volume 15, Issue 4. – Pp. 393-400.
12. Tatroe, K. Programming PHP [Text] : Creating Dynamic Web Pages / K. Tatroe, P. MacIntyre. – Boston : O'Reilly Media, 2020. – 544 p.
13. Bierer, D. PHP 8 Programming Tips, Tricks and Best Practices [Text] : A practical guide to PHP 8 features, usage changes, and advanced programming techniques / D. Bierer, C. Evans. – Birmingham : Packt Publishing, 2021. – 528 p.
14. Cooper, P. Beginning Ruby [Text] / P. Cooper. – New York : Apress, 2016. – 476 p.
15. Nixon, R. Learning PHP, MySQL & JavaScript [Text] : With jQuery, CSS & HTML5 / R. Nixon. – Sebastopol : O'Reilly, 2018. – 832 p.
16. PHP Master | The MVC Pattern and PHP, Part 1 [Electronic resource]. – Access mode : <https://www.sitepoint.com/the-mvc-pattern-and-php-1/>.
17. Klassy Cafe Template [Electronic resource]. – Access mode : <https://templatemo.com/tm-558-klassy-cafe>

ДОДАТОК А
Технічне завдання

Вступ

Розроблюване програмне забезпечення направлене на те, щоб створити актуальний інструмент для підтримки:

- користувачів, які мають інтереси, пов'язані з туристичними поїздками;
- користувачів, які здійснюють робочі поїздки;
- користувачів, які фактично живуть в подорожах, тобто є мобільним прошарком, який змінює місце своєї дислокації, у цьому місці проживаючи, працюючи.

Будь-яка подорож розглядається як подорож у одне окреме місто. Тобто у випадку подорожей у декілька міст має створюватися декілька окремих планів подорожей.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів підтримки планування подорожей», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для надання інформації про витрати в подорожах різними містами світу та планування власних майбутніх і поточних подорожей, а також фіксації фактичних витрат у цих подорожах порівняно з планами.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики програмного забезпечення повинні включати:

- створення і планування подорожі;
- редагування параметрів подорожі;
- оцінювання подорожі;
- перегляд власних подорожей користувачем;
- перегляд плану подорожі;
- внесення фактичних витрат подорожі;
- вилучення подорожі;
- перегляд найкращих міст загалом і за думкою конкретного користувача;
- перегляд найкращих подорожей користувача;
- внесення статей витрат;
- внесення груп статей витрат;
- пошук статті витрат;
- перегляд оцінок міста;
- перегляд статей витрат за заданою групою;
- перегляд витрат за статтями у місті;
- визначення статистики витрат за статтями у містах;
- внесення міст, у які можуть здійснюватися подорожі;
- обчислення середніх оцінок за подорожами у місто;
- авторизація і реєстрація користувачів.

A.3.2 Вимоги до складу та параметрів технічних засобів

Апаратне забезпечення роботи вебсервера повинно відповідати вимогам роботи з щонайменше 200 користувачами одночасно. Для підтримки

цих вимог апаратне забезпечення повинно мінімально включати:

- процесор: чотири ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять: 10 Гб;
- жорсткий диск: 300 Мб доступного простору.

A.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, у свою чергу приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Trip.php

```

<?php

class Trip {

    public static function newTrip ($city, $user, $begdate,
$enddate, $description) {
        $connect = DBSlot::Connect();
        $cityid = City::getCityIdByName($city);
        $request = "INSERT INTO Trip(cityid, userid, begdate,
enddate, description)
            VALUES (:cityparam, :userparam, :begdateparam,
:enddateparam, :descriptionparam)";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':cityparam', $cityid,
PDO::PARAM_INT);
        $prep_req->bindParam(':userparam', $user,
PDO::PARAM_INT);
        $prep_req->bindParam(':begdateparam', $begdate,
PDO::PARAM_STR);
        $prep_req->bindParam(':enddateparam', $enddate,
PDO::PARAM_STR);
        $prep_req->bindParam(':descriptionparam', $description,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function editTrip ($id, $city, $begdate,
$enddate, $description) {
        $connect = DBSlot::Connect();
        $cityid = City::getCityIdByName($city);
        $request = "UPDATE Trip
SET city = :cityidparam,
begdate = :begdateparam,
enddate = :enddateparam,
description = :descriptionparam
WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        $prep_req->bindParam(':begdateparam', $begdate,
PDO::PARAM_STR);
        $prep_req->bindParam(':enddateparam', $enddate,
PDO::PARAM_STR);
    }
}

```

```

        $prep_req->bindParam(':cityidparam', $cityid,
PDO::PARAM_INT);
        $prep_req->bindParam(':descriptionparam', $description,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function scoreTrip ($id, $description, $score,
$safety, $internet, $costs, $fun, $traffic, $heating, $speechfreedom,
$walkability, $english, $foreigners, $nightlife, $medicine) {
        $today = date("m.d.y");
        $connect = DBSlot::Connect();
        $request = "UPDATE Trip
SET description = :descriptionparam,
score = :scoreparam,
safety = :safetyparam,
internet = :internetparam,
costs = :costsparam,
fun = :funparam,
traffic = :trafficparam,
heating = :heatingparam,
speechfreedom = :speechfreedomparam,
walkability = :walkabilityparam,
english = :englishparam,
foreigners = :foreignersparam,
nightlife = :nightlifeparam,
medicine = :medicineparam,
scoredata = :scoredataparam
WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        $prep_req->bindParam(':description', $description,
PDO::PARAM_STR);
        $prep_req->bindParam(':scoreparam', $score,
PDO::PARAM_INT);
        $prep_req->bindParam(':safetyparam', $safety,
PDO::PARAM_INT);
        $prep_req->bindParam(':internetparam', $internet,
PDO::PARAM_INT);
        $prep_req->bindParam(':costsparam', $costs,
PDO::PARAM_INT);
        $prep_req->bindParam(':funparam', $fun,
PDO::PARAM_INT);
        $prep_req->bindParam(':trafficparam', $traffic,
PDO::PARAM_INT);

```

```

        $prep_req->bindParam(':heatingparam', $heating,
PDO::PARAM_INT);
        $prep_req->bindParam(':speechfreedomparam',
$speechfreedom, PDO::PARAM_STR);
        $prep_req->bindParam(':walkabilityparam', $walkability,
PDO::PARAM_INT);
        $prep_req->bindParam(':englishparam', $english,
PDO::PARAM_INT);
        $prep_req->bindParam(':foreignersparam', $foreigners,
PDO::PARAM_INT);
        $prep_req->bindParam(':nightlifeparam', $nightlife,
PDO::PARAM_INT);
        $prep_req->bindParam(':medicineparam', $medicine,
PDO::PARAM_INT);
        $prep_req->bindParam(':scoredataparam', $today,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

```

```

    public static function getUserTrips ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT id, cityid, userid, begdate,
enddate, description, score, safety, internet, costs, fun, traffic,
heating, speechfreedom, walkability, english, foreigners, nightlife,
medicine, costtotal, plannedcosttotal, scoredata
        FROM Trip WHERE userid = :userparam
        ORDER BY id DESC";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':userparam', $id);
        $prep_req->execute();
        $trips = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $trips;
    }

```

```

    public static function getTripCosts ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT id, costid, tripid, price, quant,
quantunit, detail, planprice, unitprice, planunitprice
        FROM CostValue WHERE tripid = :tripparam
        ORDER BY id";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':tripparam', $id);
        $prep_req->execute();
        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $costs;
    }

```

```

    }

    public static function addTripCost ($id, $cost, $quant,
$quantunit, $detail, $uprice, $puprice) {
        $price = $uprice * $quantunit;
        $pprice = $puprice * $quantunit;
        $connect = DBSlot::Connect();
        $request = "INSERT INTO CostValue(costid, tripid,
quant, quantunit, detail, price, planprice, unitprice, planunitprice)
VALUES (:costidparam, :tripidparam, :quantparam,
:quantunitparam, :detailparam, :priceparam, :planpriceparam,
:unitpriceparam, :planunitpriceparam)";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':costidparam', $cost,
PDO::PARAM_INT);
        $prep_req->bindParam(':tripidparam', $id,
PDO::PARAM_INT);
        $prep_req->bindParam(':quantparam', $quant,
PDO::PARAM_INT);
        $prep_req->bindParam(':quantunitparam', $quantunit,
PDO::PARAM_INT);
        $prep_req->bindParam(':detailparam', $detail,
PDO::PARAM_INT);
        $prep_req->bindParam(':priceparam', $price,
PDO::PARAM_INT);
        $prep_req->bindParam(':planpriceparam', $pprice,
PDO::PARAM_INT);
        $prep_req->bindParam(':unitpriceparam', $uprice,
PDO::PARAM_INT);
        $prep_req->bindParam(':planunitpriceparam', $puprice,
PDO::PARAM_INT);
        return $prep_req->execute();
    }

    public static function editTripCost ($id, $cost, $quant,
$quantunit, $detail, $uprice, $puprice) {
        $price = $uprice * $quantunit;
        $pprice = $puprice * $quantunit;
        $connect = DBSlot::Connect();
        $request = "UPDATE CostValue
SET costid = :costidparam,
quant = :quantparam,
quantunit = :quantunitparam,
detail = :detailparam,
price = :priceparam,

```

```

        planprice = :planpriceparam,
        unitprice = :unitpriceparam,
        planunitprice = :planunitpriceparam
        WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        $prep_req->bindParam(':costidparam', $cost,
PDO::PARAM_INT);
        $prep_req->bindParam(':quantparam', $quant,
PDO::PARAM_INT);
        $prep_req->bindParam(':quantunitparam', $quantunit,
PDO::PARAM_INT);
        $prep_req->bindParam(':detailparam', $detail,
PDO::PARAM_STR);
        $prep_req->bindParam(':priceparam', $price,
PDO::PARAM_STR);
        $prep_req->bindParam(':planpriceparam', $pprice,
PDO::PARAM_STR);
        $prep_req->bindParam(':unitpriceparam', $uprice,
PDO::PARAM_STR);
        $prep_req->bindParam(':planunitpriceparam', $puprice,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function deleteTripCost ($id) {
        $connect = DBSlot::Connect();
        $request = 'DELETE FROM CostValue
        WHERE id = :idparam';
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        return $prep_req->execute();
    }

    public static function calculateTripCosts ($id, $mode) {
        $connect = DBSlot::Connect();
        $request = "SELECT id, costid, tripid, price
        FROM CostValue WHERE tripid = :tripparam
        ORDER BY id";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':tripparam', $id);
        $prep_req->execute();
        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        $sum = 0;
        foreach ($costs as $cost) {

```

```

        $sum = $sum + $cost['price'];
    }
    $cityid = City::getCityIdByName($city);
    if ($mode) {
        $request = "UPDATE Trip
        SET costtotal = :costtotal
        WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id,
PDO::PARAM_INT);
        $prep_req->bindParam(':costtotal', $sum,
PDO::PARAM_STR);
    }
    return $sum;
}

public static function calculatePlannedTripCosts ($id, $mode) {
    $connect = DBSlot::Connect();
    $request = "SELECT id, costid, tripid, planprice
    FROM CostValue WHERE tripid = :tripparam
    ORDER BY id";
    $prep_req = $connect->prepare($request);
    $prep_req->bindParam(':tripparam', $id);
    $prep_req->execute();
    $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
    $sum = 0;
    foreach ($costs as $cost) {
        $sum = $sum + $cost['planprice'];
    }
    $cityid = City::getCityIdByName($city);
    if ($mode) {
        $request = "UPDATE Trip
        SET plancosttotal = :plancosttotalparam
        WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id,
PDO::PARAM_INT);
        $prep_req->bindParam(':plancosttotalparam', $sum,
PDO::PARAM_STR);
    }
    return $sum;
}

public static function getTripById ($id) {
    $connect = DBSlot::Connect();

```

```

        $request = "SELECT id, cityid, userid, begdate,
enddate, description, score, safety, internet, costs, fun, traffic,
heating, speechfreedom, walkability, english, foreigners, nightlife,
medicine, costtotal, plannedcosttotal, scoredata

```

```

        FROM Trip WHERE id = :idparam";

```

```

        $prep_req = $connect->prepare($request);

```

```

        $prep_req->bindParam(':idparam', $id);

```

```

        $prep_req->execute();

```

```

        $trip = $prep_req->fetch(PDO::FETCH_ASSOC);

```

```

        return $trip;
    }

```

```

    public static function deleteTrip ($id) {

```

```

        $connect = DBSlot::Connect();

```

```

        $request = 'DELETE FROM Trip

```

```

        WHERE id = :idparam';

```

```

        $prep_req = $connect->prepare($request);

```

```

        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);

```

```

        return $prep_req->execute();
    }

```

```

    public static function getTripCostsDeviation ($id) {

```

```

        $connect = DBSlot::Connect();

```

```

        $request = "SELECT id, costid, tripid, price, quant,
quantunit, detail, planprice

```

```

        FROM CostValue WHERE tripid = :tripparam

```

```

        ORDER BY id";

```

```

        $prep_req = $connect->prepare($request);

```

```

        $prep_req->bindParam(':tripparam', $id);

```

```

        $prep_req->execute();

```

```

        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);

```

```

        $sum = 0;

```

```

        foreach ($costs as $cost) {

```

```

            $p1 = $cost['planprice'];

```

```

            $p2 = $cost['price'];

```

```

            $cost['deviation'] = $p2 / $p1 * 100;

```

```

        }

```

```

        return $costs;
    }

```

```

    public static function getTripDeviation ($id) {

```

```

        $connect = DBSlot::Connect();

```

```

        $request = "SELECT id, cityid, userid, begdate,
enddate, description, score, safety, internet, costs, fun, traffic,

```

heating, speechfreedom, walkability, english, foreigners, nightlife,
 medicine, costtotal, plannedcosttotal, scoredata

```

    FROM Trip WHERE id = :idparam";
    $prep_req = $connect->prepare($request);
    $prep_req->bindParam(':idparam', $id);
    $prep_req->execute();
    $trip = $prep_req->fetch(PDO::FETCH_ASSOC);
    $p1 = $trip['plannedcosttotal'];
    $p2 = $trip['costtotal'];
    $trip['deviation'] = $p2 / $p1 * 100;
    return $trip;
}

public static function getPropositions ($user) {
    $propositions = file('propositions.dat');
    return $propositions[$user];
}

public static function getPlan ($costs, $city) {
    $connect = DBSlot::Connect();
    $request = "SELECT price
    FROM CostCity
    WHERE costid = :costparam AND cityid = :cityparam";
    $sum = 0;
    foreach ($costs as $cost) {
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':costparam', $cost['id']);
        $prep_req->bindParam(':cityparam', $city);
        $prep_req->execute();
        $price = $prep_req->fetch(PDO::FETCH_ASSOC);
        $cost['price'] = $price;
        $sum = $sum + $price;
    }
    $sumv['id'] = -1;
    $sumv['price'] = $sum;
    array_push($costs, $sumv);
    return $costs;
}

public static function getUserBestCities ($user) {
    $connect = DBSlot::Connect();
    $request = "SELECT DISTINCT City.name,
    FROM City
    INNER JOIN Trip ON City.id = Trip.cityid
    WHERE Trip.userid = :userparam

```

```

ORDER BY Trip.score
LIMIT 10");
$prep_req = $connect->prepare($request);
$prep_req->bindParam(':userparam', $user);
$prep_req->execute();
$cities = $prep_req->fetchAll(PDO::FETCH_ASSOC);
return $cities;
}

public static function getUserBestTrips ($user) {
    $connect = DBSlot::Connect();
    $request = "SELECT id, cityid, userid, begdate,
enddate, description, score, safety, internet, costs, fun, traffic,
heating, speechfreedom, walkability, english, foreigners, nightlife,
medicine, costtotal, plannedcosttotal
FROM Trip
WHERE userid = :userparam
ORDER BY score
LIMIT 10");
$prep_req = $connect->prepare($request);
$prep_req->bindParam(':userparam', $user);
$prep_req->execute();
$trips = $prep_req->fetchAll(PDO::FETCH_ASSOC);
return $trips;
}
}

```

Б.2 Текст файла Cost.php

```

<?php

class Cost {

    public static function newCost ($groupid, $name, $quantunit) {
        $connect = DBSlot::Connect();
        $cityid = City::getCityIdByName($city);
        $request = "INSERT INTO Trip(groupid, name, quantunit)
VALUES (:groupparam, :nameparam, :quantunitparam)";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':groupparam', $groupid,
PDO::PARAM_INT);
        $prep_req->bindParam(':nameparam', $name,
PDO::PARAM_STR);
    }
}

```

```

        $prep_req->bindParam(':quantunitparam', $quantunit,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function editCost ($id, $groupid, $name,
$quantunit) {
        $connect = DBSlot::Connect();
        $request = "UPDATE Cost
SET groupid = :groupparam,
name = :nameparam,
quantunit = :quantunitparam
WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        $prep_req->bindParam(':groupparam', $groupid,
PDO::PARAM_INT);
        $prep_req->bindParam(':nameparam', $name,
PDO::PARAM_STR);
        $prep_req->bindParam(':quantunitparam', $quantunit,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function getCostsByGroup ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT Cost.id, Cost.groupid,
Cost.quantunit, Group.name, Cost.name
FROM Cost WHERE groupid = :idparam
INNER JOIN Group ON Group.id = City.groupid
ORDER BY Cost.name";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id);
        $prep_req->execute();
        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $costs;
    }

    public static function getCosts () {
        $connect = DBSlot::Connect();
        $request = "SELECT Cost.id, Cost.groupid,
Cost.quantunit, Group.name, Cost.name
FROM Cost
INNER JOIN Group ON Group.id = City.groupid
ORDER BY Cost.name";
        $prep_req = $connect->prepare($request);
        $prep_req->execute();
    }

```

```

        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $costs;
    }

    public static function getCostById ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT Cost.id, Cost.groupid,
Cost.quantunit, Group.name, Cost.name
        FROM Cost WHERE id = :idparam
        INNER JOIN Group ON Group.id = City.groupid
        ORDER BY Cost.name";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id);
        $prep_req->execute();
        $cost = $prep_req->fetch(PDO::FETCH_ASSOC);
        return $cost;
    }

    public static function searchCost ($name) {
        $connect = DBSlot::Connect();
        $request = "SELECT Cost.id, Cost.groupid,
Cost.quantunit, Group.name, Cost.name
        FROM Cost WHERE Cost.name LIKE :pattern
        INNER JOIN Group ON Group.id = City.groupid
        ORDER BY Cost.name";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':pattern', $name);
        $prep_req->execute();
        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $costs;
    }

    public static function getCityCosts ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT CostValue.price, Cost.name
        FROM CostValue WHERE cityid = :idparam
        INNER JOIN Cost ON Cost.id = CostValue.costid";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id);
        $prep_req->execute();
        $costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        return $costs;
    }

    public static function analyzeCityCosts () {
        $connect = DBSlot::Connect();
        $request = "SELECT id, name FROM City ORDER BY id";
        $prep_req = $connect->prepare($request);

```

```

$prep_req->execute();
$cities = $prep_req->fetchAll(PDO::FETCH_ASSOC);
$request = "SELECT id, name, quantunit FROM Cost ORDER
BY id";

$prep_req = $connect->prepare($request);
$prep_req->execute();
$costs = $prep_req->fetchAll(PDO::FETCH_ASSOC);
$sum = 0;
foreach ($cities as $city) {
    foreach ($costs as $cost) {
        $request = "SELECT unitprice, quantunit
FROM CostValue
WHERE costid = :costparam AND Trip.cityid =
:cityparam

INNER JOIN Trip ON CostValue.tripid = Trip.id";
$prep_req = $connect->prepare($request);
$prep_req->bindParam(':costparam',
$cost['id']);
$prep_req->bindParam(':cityparam',
$city['id']);

$prep_req->execute();
$trcosts = $prep_req-
>fetchAll(PDO::FETCH_ASSOC);
$sum = 0;
$q = 0;
foreach ($trcosts as $trc) {
    if ($trc['quantunit'] ==
$cost['quantunit']) {

        $sum = $sum + trc['unitprice'];
    }
    if ($trc['quantunit'] == 0) {
        $sum = $sum + trc['unitprice'] / 24;
    }
    if ($trc['quantunit'] == 0) {
        $sum = $sum + trc['unitprice'] * 24;
    }
    $q = $q + 1;
}
if ($q == 0) {
    $request2 = "INSERT INTO CityCost(cityid,
costid, price)

VALUES (:cityparam, :costparam, 0)";
$prep_req2->bindParam(':cityparam', $city,
PDO::PARAM_INT);

```

```

                                $prep_req2->bindParam(':costparam', $cost,
PDO::PARAM_INT);
                                $prep_req2->execute();
                                }
                                else {
                                    $sum = $sum / q;
                                    $sum = round ($sum, 2);
                                    $request2 = "UPDATE CityCost
                                    SET price = :priceparam
                                    WHERE cityid = :cityparam AND costid =
:costparam";
                                    $prep_req2 = $connect->prepare($request2);
                                    $prep_req->bindParam(':priceparam', $sum,
PDO::PARAM_STR);
                                    $prep_req2->bindParam(':cityparam', $city,
PDO::PARAM_INT);
                                    $prep_req2->bindParam(':costparam', $cost,
PDO::PARAM_INT);
                                    $prep_req2->execute();
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

Б.3 Текст файла City.php

```

<?php

class City {

    public static function newCity ($name, $country) {
        $connect = DBSlot::Connect();
        $countrycode = getCountryCode($country, True);
        $request = "INSERT INTO City(name, country,
overallscore, overallsafety, overallinternet, overallcosts, overallfun,
overalltraffic, overallheating, overallspeechfreedom, overallwalkability,
overallenglish, overallforeigners, overallnightlife, overallmedicine)
VALUES (:nameparam, :countryparam, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0)";
        $prep_req = $connect->prepare($request);
    }
}

```

```

        $prep_req->bindParam(':nameparam', $name,
PDO::PARAM_STR);
        $prep_req->bindParam(':countryparam', $countrycode,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function editCity ($id, $name, $country) {
        $countrycode = getCountryCode($country, True);
        $connect = DBSlot::Connect();
        $request = "UPDATE City
SET name = :nameparam,
country = :countryparam
WHERE id = :idparam";
        $prep_req->bindParam(':nameparam', $name,
PDO::PARAM_STR);
        $prep_req->bindParam(':countryparam', $countrycode,
PDO::PARAM_STR);
        return $prep_req->execute();
    }

    public static function getCityIdByName ($name) {
        $connect = DBSlot::Connect();
        $request = "SELECT id
FROM City WHERE name = :nameparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':nameparam', $name,
PDO::PARAM_STR);
        $prep_req->execute();
        $city = $prep_req->fetch(PDO::FETCH_ASSOC);
        return $city;
    }

    public static function getCityById ($id) {
        $connect = DBSlot::Connect();
        $request = "SELECT name, country, overallscore,
overallsafety, overallinternet, overallcosts, overallfun, overalltraffic,
overallheating, overallspeechfreedom, overallwalkability, overallenglish,
overallforeigners, overallnightlife, overallmedicine
FROM City WHERE id = :idparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':idparam', $id, PDO::PARAM_INT);
        $prep_req->execute();
        $city = $prep_req->fetch(PDO::FETCH_ASSOC);
    }

```

```

        $city['country'] = getCountryCode($city['country'],
False);
        return $city;
    }

    public static function getCities () {
        $connect = DBSlot::Connect();
        $request = "SELECT name, country, overallscore,
overallsafety, overallinternet, overallcosts, overallfun, overalltraffic,
overallheating, overallspeechfreedom, overallwalkability, overallenglish,
overallforeigners, overallnightlife, overallmedicine
FROM City ORDER BY name";
        $prep_req = $connect->prepare($request);
        $prep_req->execute();
        $city = $prep_req->fetch(PDO::FETCH_ASSOC);
        $city['country'] = getCountryCode($city['country'],
False);
        return $city;
    }

    public static function getBestCities () {
        $connect = DBSlot::Connect();
        $request = "SELECT name, country, overallscore,
overallsafety, overallinternet, overallcosts, overallfun, overalltraffic,
overallheating, overallspeechfreedom, overallwalkability, overallenglish,
overallforeigners, overallnightlife, overallmedicine
FROM City ORDER BY overallscore LIMIT 10";
        $prep_req = $connect->prepare($request);
        $prep_req->execute();
        $cities = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        foreach ($cities as $city) {
            $city['country'] = getCountryCode($city['country'],
False);
        }
        return $cities;
    }

    public static function calculateCities () {
        $connect = DBSlot::Connect();
        $request = "SELECT id, name FROM City ORDER BY id";
        $prep_req = $connect->prepare($request);
        $prep_req->execute();
        $cities = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        $sum = 0;
        foreach ($cities as $city) {

```

```

        $request = "SELECT score, safety, internet, costs,
fun, traffic, heating, speechfreedom, walkability, english, foreigners,
nightlife, medicine, scoredata
        FROM Trip
        WHERE cityid = :cityparam";
        $prep_req = $connect->prepare($request);
        $prep_req->bindParam(':cityparam', $city['id']);
        $prep_req->execute();
        $trips = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        $score = 0;
        $safety = 0;
        $internet = 0;
        $costs = 0;
        $fun = 0;
        $traffic = 0;
        $heating = 0;
        $speechfreedom = 0;
        $walkability = 0;
        $english = 0;
        $foreigners = 0;
        $nightlife = 0;
        $medicine = 0;
        $tot = count($trips);
        foreach ($trips as $trip) {
            $score = $score + $trip['score'] / $tot;
            $safety = $safety + $trip['safety'] / $tot;
            $internet = $internet + $trip['internet'] /
$tot;

            $costs = $costs + $trip['costs'] / $tot;
            $fun = $fun + $trip['fun'] / $tot;
            $traffic = $traffic + $trip['traffic'] / $tot;
            $heating = $heating + $trip['heating'] / $tot;
            $speechfreedom = $speechfreedom +
$trip['speechfreedom'] / $tot;
            $walkability = $walkability +
$trip['walkability'] / $tot;
            $english = $english + $trip['english'] / $tot;
            $foreigners = $foreigners + $trip['foreigners']
/ $tot;

            $nightlife = $nightlife + $trip['nightlife'] /
$tot;

            $medicine = $medicine + $trip['medicine'] /
$tot;

        }
        $score = round($score, 2);

```

```

$safety = round($safety, 2);
$internet = round($internet, 2);
$costs = round($costs, 2);
$fun = round($fun, 2);
$traffic = round($traffic, 2);
$heating = round($heating, 2);
$speechfreedom = round($speechfreedom, 2);
$walkability = round($walkability, 2);
$english = round($english, 2);
$foreigners = round($foreigners, 2);
$nightlife = round($nightlife, 2);
$medicine = round($medicine, 2);
$request2 = "UPDATE City
SET overallscore = :scoreparam,
overallsafety = :safetyparam,
overallinternet = :internetparam,
overallcosts = :costsparam,
overallfun = :funparam,
overalltraffic = :trafficparam,
overallheating = :heatingparam,
overallspeechfreedom = :speechfreedomparam,
overallwalkability = :walkabilityparam,
overallenglish = :englishparam,
overallforeigners = :foreignersparam,
overallnightlife = :nightlifeparam,
overallmedicine = :medicineparam
WHERE cityid = :cityparam";
$prep_req2 = $connect->prepare($request2);
$prep_req2->bindParam(':scoreparam', $score,
PDO::PARAM_STR);
$prep_req2->bindParam(':safetyparam', $safety,
PDO::PARAM_STR);
$prep_req2->bindParam(':internetparam', $internet,
PDO::PARAM_STR);
$prep_req2->bindParam(':costsparam', $costs,
PDO::PARAM_STR);
$prep_req2->bindParam(':funparam', $fun,
PDO::PARAM_STR);
$prep_req2->bindParam(':trafficparam', $traffic,
PDO::PARAM_STR);
$prep_req2->bindParam(':heatingparam', $heating,
PDO::PARAM_STR);
$prep_req2->bindParam(':speechfreedomparam',
$speechfreedom, PDO::PARAM_STR);

```

```

        $prep_req2->bindParam(':walkabilityparam',
$walkability, PDO::PARAM_STR);
        $prep_req2->bindParam(':englishparam', $english,
PDO::PARAM_STR);
        $prep_req2->bindParam(':foreignersparam',
$foreigners, PDO::PARAM_STR);
        $prep_req2->bindParam(':nightlifeparam',
$nightlife, PDO::PARAM_STR);
        $prep_req2->bindParam(':medicineparam', $medicine,
PDO::PARAM_STR);
        $prep_req2->bindParam(':cityparam', $city['id'],
PDO::PARAM_STR);
        $prep_req2->execute();
    }
}

```

```

public static function getCountryCode ($name, mode) {
    $countries = array ("Австралія"=>"AUS", " Австрія"=>"AUT",
"Азербайджан"=>"AZE", "Албанія"=>"ALB", "Алжир"=>"DZA",
"Андорра"=>"AND", "Аргентина"=>"ARG", "Афганістан"=>"AFG",
"Бельгія"=>"BEL ", "Білорусь"=>"BLR", "Болгарія"=>"BGR ",
"Болівія"=>"BOL ", "Боснія і Герцеговина"=>"BIH ", "Бразилія"=>"BRA ",
"Велика Британія"=>"GBR ", "Венесуела"=>"VEN ", "В'єтнам"=>"VNM ",
"Вірменія"=>"ARM ", "Гаїті"=>"HTI", "Гана"=>"GHA", "Гватемала"=>"GTM",
"Гвінея"=>"GIN", "Гондурас"=>"HND", "Гонконг"=>"HKG",
"Гренада"=>"GRD", "Греція"=>"GRC", "Грузія"=>"GEO",
"Гібралтар"=>"GIB", "Гренландія"=>"GRL", "Данія"=>"DNK", "ДР
Конго"=>"COD", "Домініканська Республіка"=>"DOM", "Еквадор"=>"ECU",
"Екваторіальна Гвінея"=>"GNQ", "Еритрея"=>"ERI", "Естонія"=>"EST",
"Ефіопія"=>"ETH", "Єгипет"=>"EGY", "Ємен"=>"YEM", "Замбія"=>"ZMB",
"Зімбабве"=>"ZWE", "Ізраїль"=>"ISR", "Індія"=>"IND",
"Індонезія"=>"IDN", "Ірак"=>"IRQ", "Іран"=>"IRN", "Ірландія"=>"IRL",
"Ісландія"=>"ISL", "Іспанія"=>"ESP", "Італія"=>"ITA",
"Йорданія"=>"JOR", "Кабо-Верде"=>"CPV", "Казахстан"=>"KAZ",
"Камбоджа"=>"KHM", "Камерун"=>"CMR", "Канада"=>"CAN", "Катар"=>"QAT",
"Кенія"=>"KEN", "Киргизстан"=>"KGZ", "Китай"=>"CHN", "Кіпр"=>"CYP",
"Колумбія"=>"COL", "Республіка Конго"=>"COG", "Коста-Рика"=>"CRI",
"Кот-д'Івуар"=>"CIV", "Куба"=>"CUB", "Кувейт"=>"KWT", "Латвія"=>"LVA",
"Литва"=>"LTU", "Ліберія"=>"LBR", "Ліван"=>"LBN", "Лівія"=>"LBY",
"Ліхтенштейн"=>"LIE", "Люксембург"=>"LUX", "Маврикій"=>"MUS",
"Мавританія"=>"MRT", "Мадагаскар"=>"MDG", "Малайзія"=>"MYS",
"Малі"=>"MLI", "Мальдіви"=>"MDV", "Мальта"=>"MLT", "Марокко"=>"MAR",
"Мексика"=>"MEX", "Мозамбік"=>"MOZ", "Молдова"=>"MDA",
"Монако"=>"MCO", "Монголія"=>"MNG", "Намібія"=>"NAM", "Непал"=>"NPL",
"Нігерія"=>"NGA", "Нідерланди"=>"NLD", "Нікарагуа"=>"NIC",

```

```

"Німеччина"=>"DEU", "Нова Зеландія"=>"NZL", "Норвегія"=>"NOR",
"ОАЕ"=>"ARE", "Оман"=>"OMN", "Пакистан"=>"PAK", "Палестина"=>"PSE",
"Панама"=>"PAN", "Папуа Нова Гвінея"=>"PNG", "Парагвай"=>"PRY",
"Перу"=>"PER", "ПАР"=>"ZAF", "Південна Корея"=>"KOR", "Південний
Судан"=>"SSD", "Північна Корея"=>"PRK", "Північна Македонія"=>"MKD",
"Польща"=>"POL", "Португалія"=>"PRT", "Пуерто-Рико"=>"PRI",
"Росія"=>"RUS", "Руанда"=>"RWA", "Румунія"=>"ROU", "Сальвадор"=>"SLV",
"Самоа"=>"WSM", "Сан-Марино"=>"SMR", "Саудівська Аравія"=>"SAU",
"Сейшельські Острови"=>"SYC", "Сенегал"=>"SEN", "Сербія"=>"SRB",
"Сирія"=>"SYR", "Сінгапур"=>"SGP", "Словаччина"=>"SVK",
"Словенія"=>"SVN", "США"=>"USA", "Судан"=>"SDN", "Сьєрра-
Леоне"=>"SLE", "Таджикистан"=>"TJK", "Таїланд"=>"THA",
"Тайвань"=>"TWN", "Танзанія"=>"TZA", "Того"=>"TGO", "Тринідад і
Тобаго"=>"TTO", "Тувалу"=>"TUV", "Туніс"=>"TUN", "Туреччина"=>"TUR",
"Туркменістан"=>"TKM", "Уганда"=>"UGA", "Угорщина"=>"HUN",
"Узбекистан"=>"UZB", "Україна"=>"UKR", "Уругвай"=>"URY", "Фарерські
острови"=>"FRO", "Фіджі"=>"FJI", "Філіппіни"=>"PHL",
"Фінляндія"=>"FIN", "Фолклендські Острови"=>"FLK", "Франція"=>"FRA",
"Хорватія"=>"HRV", "ЦАР"=>"CAF", "Чад"=>"TCD", "Чехія"=>"CZE",
"Чилі"=>"CHL", "Чорногорія"=>"MNE", "Швейцарія"=>"CHE",
"Швеція"=>"SWE", "Шрі-Ланка"=>"LKA", "Ямайка"=>"JAM",
"Японія"=>"JPN",);

```

```

        if ($mode) {
            return $countries[$name];
        }
        else {
            $res = "";
            foreach ($countries as $k => $v) {
                if ($v == $name) {
                    $res = $k;
                }
            }
            return $res;
        }
    }
}

```

```

public static function getAllBestCities () {
    $connect = DBSlot::Connect();
    $request = "SELECT name, country, overallscore,
overallsafety, overallinternet, overallcosts, overallfun, overalltraffic,
overallheating, overallspeechfreedom, overallwalkability, overallenglish,
overallforeigners, overallnightlife, overallmedicine
FROM City ORDER BY overallscore";
    $prep_req = $connect->prepare($request);
    $prep_req->execute();
}

```

```

        $cities = $prep_req->fetchAll(PDO::FETCH_ASSOC);
        foreach ($cities as $city) {
            $city['country'] = getCountryCode($city['country'],
False);
        }
    }
}

```

ДОДАТОК В
Слайди презентації

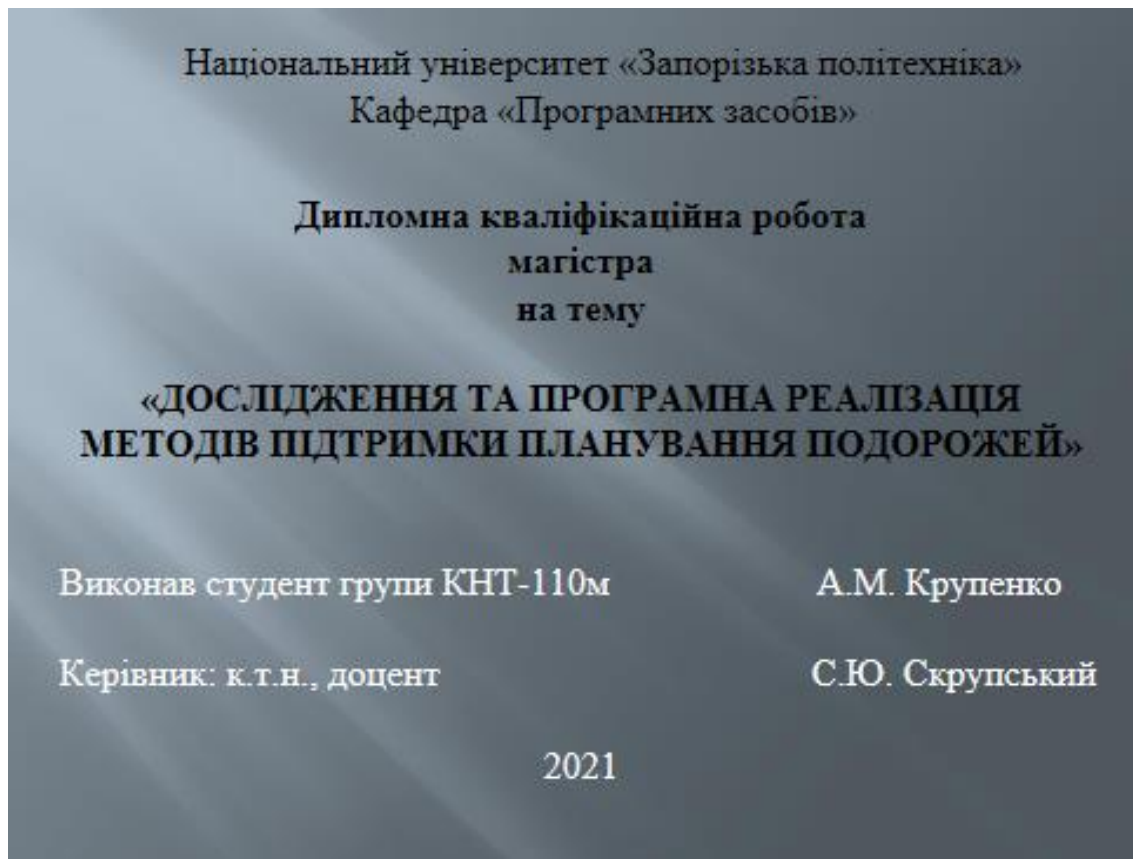


Рисунок В.1 – Слайд 1

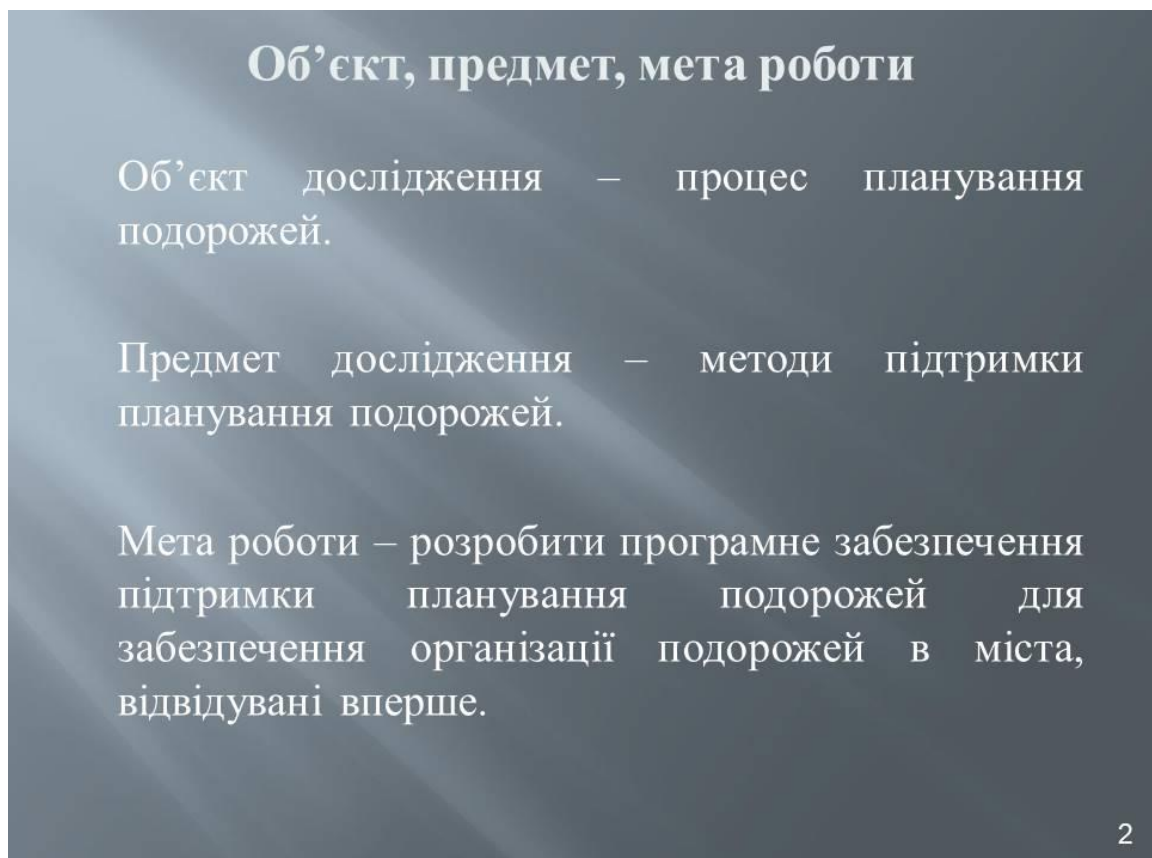


Рисунок В.2 – Слайд 2

Завдання роботи

- ✓ виконати аналіз програмних засобів підтримки планування подорожей;
- ✓ виконати проєктування програмного забезпечення;
- ✓ реалізувати програмне забезпечення;
- ✓ створити новий метод планування подорожей для розв'язання проблеми, що розглядається;
- ✓ виконати експериментальне дослідження проблеми.

3

Рисунок В.3 – Слайд 3

Порівняння програмних аналогів

Критерій	Nomadlist	Numbeo	Власна розробка
Планування подорожі	Немає	Немає	Так
Визначення пропозицій для майбутньої подорожі	Немає	Немає	Так
Визначення статистики витрат	Обмежене	Широке	Широке
Оцінювання міст	Так	Немає	Так

4

Рисунок В.4 – Слайд 4

Метод підтримки планування подорожей

Етап 1. Визначення масиву даних T , що містить оцінки, виставлені кожним користувачем містам з наявного набору, і дати виставлення цих оцінок.

Етап 2. Перетворення дати виставлення оцінок на вагову мітку віддаленості за часом від поточного моменту:

$$w_i = \frac{p_i}{\sum_{i=1}^n p_i}.$$

Етап 3. Вибір в якості поточного користувача j послідовно кожного користувача з наявних і виконання для кожного з поточних користувачів етапів 4-6, після чого перехід до етапу 7.

Етап 4. Обчислення подібності між оцінками користувача j та оцінками всіх інших користувачів u (значення від 1 до U) окрім j на основі формул:

$$\begin{aligned} \text{sim}_a(T_j^s, T_u^s) &= 1 - \frac{|T_j^s(a) - T_u^s(a)|}{\max(a) - \min(a)}, \\ M(T_j^s, T_u^s) &= \sum_{a=1}^n w_a \text{sim}_a(T_j^s, T_u^s). \end{aligned}$$

Етап 5. Визначення для j -го користувача k користувачів, які мають найбільший рівень подібності $\text{SIM}(T_j^s, T_u^s)$ до його оцінок.

Етап 6. Виділення z міст з найкращою сумарною оцінкою серед k користувачів.

Етап 7. Збереження отриманих результатів.

Етап 8. Вибір міста з представленого набору z міст користувачем.

Етап 9. Обчислення середнього рівня витрат за кожною статтею витрат для заданого міста або витягання відповідних збережених даних з представленням в результаті масиву C .

Етап 10. Визначення плану витрат та обчислення його вартості за значеннями елементів з масиву C .

5

Рисунок В.5 – Слайд 5

Порівняння мов програмування

Критерій	PHP	Ruby
Швидкість роботи створених проєктів	Вища	Менша
Простота вивчення можливостей	Вища	Менша
Засоби веброзробки	Співмірні	Співмірні
Засоби машинного навчання	Співмірні	Співмірні
Простота роботи з базами даних	Співмірна	Співмірна
Простота запуску на різних платформах	Вища	Менша
Підтримка великої кількості розробників	Вища	Менша

6

Рисунок В.6 – Слайд 6



Рисунок В.7 – Слайд 7

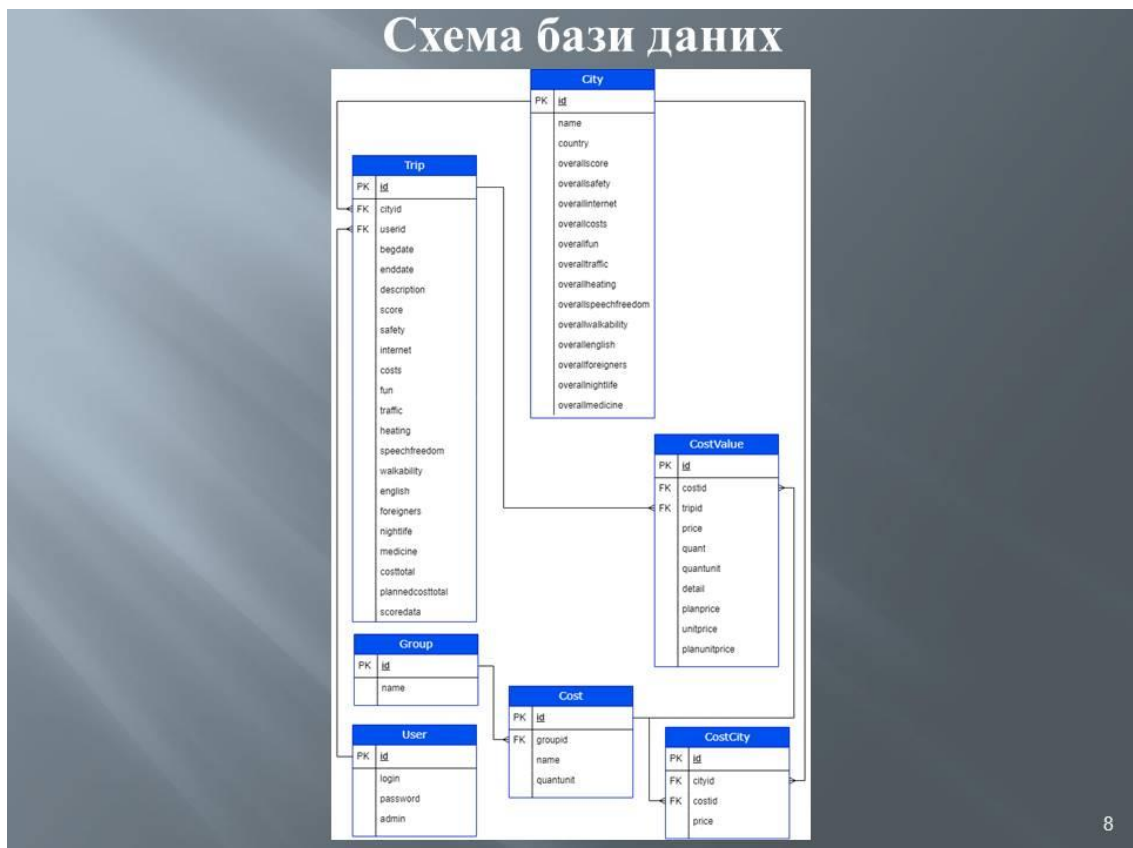


Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9



Рисунок В.10 – Слайд 10

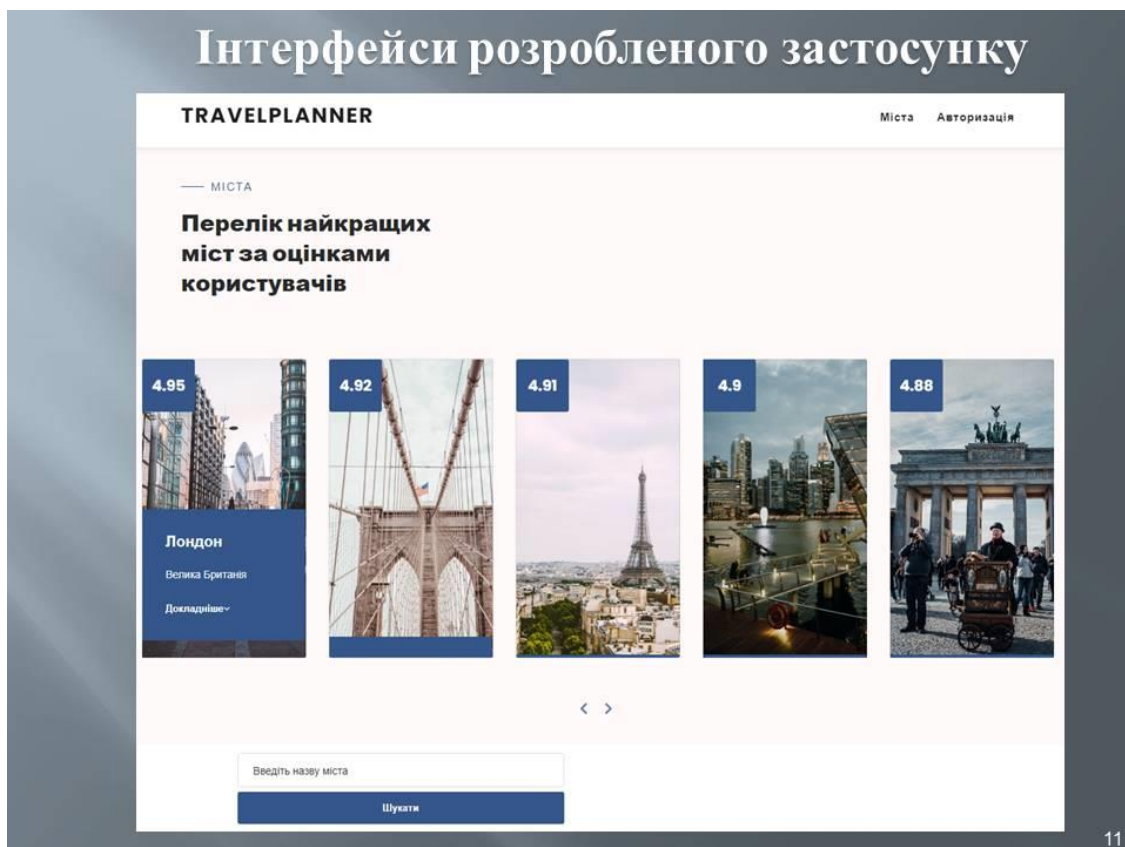


Рисунок В.11 – Слайд 11



Рисунок В.12 – Слайд 12

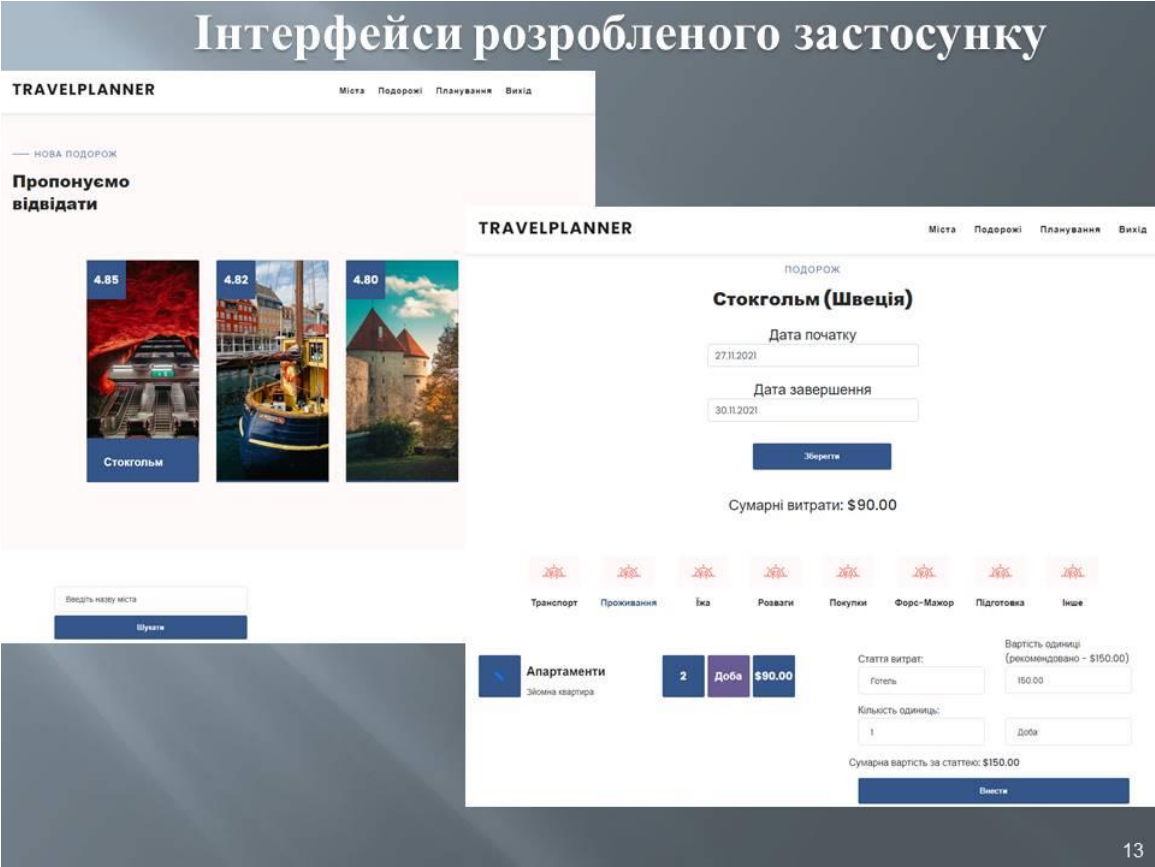


Рисунок В.13 – Слайд 13

Результати експериментального дослідження	
Метод	Точність
Метод k-найближчих сусідів	0,71
Метод підтримки планування подорожей	0,84

Рисунок В.14 – Слайд 14

Висновки

Розроблене програмне забезпечення дозволяє створювати і планувати подорожі, редагувати параметри подорожі, оцінювати подорож, переглядати власні подорожі, план подорожі, вносити фактичні витрати подорожі, вилучати власні подорожі, переглядати найкращі міста загалом і за думкою конкретного користувача, переглядати власні найкращі подорожі, шукати статті витрат, переглядати оцінки міста, переглядати статті витрат за заданою групою, переглядати витрати за статтями у місті, визначати статистику витрат за статтями у містах, обчислювати середні оцінки за подорожами у місто, а також адміністратору вносити статті витрат, групувати їх, внести міста, у які можуть здійснюватися подорожі.

Наукова новизна роботи полягає в запропонованому методі підтримки планування подорожей, який виконує визначення міст в якості альтернативних варіантів для подорожі на основі методу k-найближчих сусідів з врахуванням уподобань інших користувачів і віддаленості такого уподобання в часі, виконує фінансове планування подорожі шляхом визначення плану на основі статей витрат, суми за якими обчислюються шляхом усереднення значень, отриманих іншими користувачами, що дозволяє самостійно організовувати подорожі в міста, відвідувані вперше.