

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)

бакалавра

(ступінь вищої освіти)

на тему РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПЛАНУВАННЯ
БЮДЖЕТУ ПОДОРОЖЕЙ
SOFTWARE DEVELOPMENT FOR TRAVEL BUDGET PLANNING

Виконав: студент(ка) 4 курсу, групи КНТ-137
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Моїсеєнко Д.Я.

(прізвище та ініціали)

Керівник Степаненко О.О.

(прізвище та ініціали)

Рецензент Голуб Т.В.

(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 бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

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

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Моїсєєнка Данила Ярославовича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка програмного забезпечення планування
бюджету подорожей
Software Development for Travel Budget Planning

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

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

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

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми.
3. Розробка програми. 4. Експлуатація, 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
1-4 Основна частина	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “15” березня 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 предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис) Моїсеєнко Д.Я.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 83 с., 29 рис., 1 табл., 3 дод., 15 джерел.

ПЛАНУВАННЯ БЮДЖЕТУ ПОДОРОЖІ, ВІДСЛІДКОВУВАННЯ ВИТРАТ ПОДОРОЖІ, ГРУПОВІ ПОДОРОЖІ, ВЕБЗАСТОСУНОК, МОВА PYTHON, ВЕБФРЕЙМВОРК DJANGO.

Об'єкт дослідження – процес планування бюджету подорожей.

Предмет дослідження – програмне забезпечення планування бюджету подорожей.

Мета роботи – розроблення програмного забезпечення планування бюджету подорожей для підвищення зручності процесу планування і подальшого відслідковування витрат як індивідуальними мандрівниками, так і мандрівниками у складі груп.

Матеріали, методи та технічні засоби: мова програмування Python, вебфреймворк Django, система керування базами даних MySQL.

Результати. Проведено аналіз програмних застосунків, що використовуються під час планування бюджету подорожей, на основі чого визначено концепцію створення програмного забезпечення. Виконано проєктування програмного забезпечення на основі створення діаграм варіантів використання, послідовностей, класів, схеми бази даних. Реалізовано програмне забезпечення.

Висновки. Створений вебзастосунок дозволяє працювати над плануванням бюджету як індивідуальних, так і групових подорожей, які можуть охоплювати і декілька країн, а також над відслідковуванням фактичних витрат.

Галузь використання – планування і відслідковування витрат подорожей.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 29 figures, 1 table, 3 appendixes, 15 sources.

TRAVEL BUDGET PLANNING, TRAVEL COST TRACKING, GROUP TRAVEL, WEB APPLICATION, PYTHON LANGUAGE, DJANGO WEB FRAMEWORK.

The object of research is the process of planning a travel budget.

The subject of research is travel budget planning software.

The purpose of the work is to develop travel budget planning software to increase the convenience of the planning process and further costs tracking for both individual travelers and travelers in groups.

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

Results. The analysis of software applications used during travel budget planning is carried out. On the basis of the performed analysis the concept of software creation is determined. The software was designed on the basis of creating diagrams of use cases, sequences, classes, database schema. The software was implemented.

Conclusions. The created web application allows to work on budget planning for both individual and group travelers, as well as on tracking actual costs. Such travels can cover several countries.

Field of usage is travel planning and tracking.

ЗМІСТ

С.

Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Програмні застосунки планування і відслідковування бюджету подорожей.....	10
1.2 Програмні застосунки відслідковування споживчих цін.....	14
1.3 Висновки за розділом 1	16
2 Розробка архітектури програми.....	17
2.1 Вибір засобів проєктування і розробки	17
2.2 Визначення функціональних вимог до програмного забезпечення	18
2.3 Визначення архітектури програмного забезпечення.....	23
2.4 Визначення структури бази даних	26
2.5 Висновки за розділом 2	31
3 Розробка програми	33
3.1 Організація шару роботи з даними	33
3.2 Організація шару відображення	36
3.3 Висновки за розділом 3	42
4 Експлуатація, тестування та експериментальне дослідження програми	43
4.1 Призначення програми	43
4.2 Умови виконання програми	43
4.3 Тестування програми	44
Висновки	51
Перелік джерел посилання	53
Додаток А Технічне завдання	55
Додаток Б Текст програми	60
Додаток В Слайди презентації.....	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVT – object-relational mapping;

ORM – Model-View-Template.

ВСТУП

Важко визначити туристичну галузь повністю і однозначно, оскільки, на відміну від інших галузей, вона не має жодного чіткого продукту і при цьому охоплює багато галузей, включаючи готельний бізнес, транспорт, сферу розваг, туристичні компанії тощо, у найширшому розумінні туризм визначається як процес, під час якого люди подорожують і перебувають у місцях, що перебувають поза їх звичним середовищем, менше одного року поспіль з метою дозвілля, бізнесу, здоров'я чи інших причин, у глобальному масштабі прямий внесок подорожей та туризму у валовий зовнішній продукт становив приблизно 2,9 трільйонів доларів США у 2019 році [1].

Зважаючи на величезний обсяг галузі туризму варто виділити, що в ній існують як організовані туристичними компаніями подорожі, так і подорожі, які створюються людьми без участі туристичних компаній. У такому випадку функція планування подорожі покладається на мандрівника або групу мандрівників, якщо це самотійна, але групова подорож. Для якісної реалізації даного етапу має бути зручний програмний інструмент, адже планувати подорож на папері можливо, а ось з фіксацією фактичних витрат таким же чином зазвичай виникають проблеми, які призводять до того, що часто людина виконує планування, починає фіксувати витрати, але в процесі губиться і цей процес нічим не завершується. Усе це вказує на актуальність розроблення програмного забезпечення планування бюджету подорожей.

Об'єкт дослідження – процес планування бюджету подорожей.

Предмет дослідження – програмне забезпечення планування бюджету подорожей.

Мета роботи полягає в розробленні програмного забезпечення планування бюджету подорожей для підвищення зручності процесу планування і подальшого відслідковування витрат як індивідуальними мандрівниками, так і мандрівниками у складі груп.

Для виконання мети роботи було визначено її завдання:

- аналіз існуючих програмних аналогів;
- визначення вимог до програмного забезпечення планування бюджету подорожей;
- проєктування програмного забезпечення;
- реалізація програмного застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмні застосунки планування і відслідковування бюджету подорожей

На даний момент можна виділити декілька програмних застосунків, які стосуються повністю або частково розв'язуваної проблеми.

TrabeePocket [2] (рис. 1.1) – це доволі простий у взаємодії застосунок, але один з найкращих програмних застосунків, який дозволяє відстежувати витрати за задану подорож:

- надаються версії для Android та iOS;
- перш ніж почати подорож, можна додати всі передплачені витрати на вкладку «підготовка»: це дозволяє враховувати свої рейси та будь-які тури, які були замовлені заздалегідь;
- після початку подорожі можна легко додати всі додаткові витрати на проїзд;
- при цьому можна класифікувати свої витрати на вісім різних категорій, а за допомогою платної версії можна додати скільки завгодно додаткових категорій, якщо є конкретні витрати, які потрібно відстежувати;
- безкоштовна версія дозволяє відстежувати лише одну валюту за поїздку, але платна версія дозволяє відстежувати кілька валют в одному файлі поїздки;
- TrabeePocket допомагає встановити щомісячний бюджет і відстежувати свої витрати відповідно до цієї загальної кількості, окрім того дохід, який було отримано під час подорожі, можна також внести;
- надається можливість фотографувати і прив'язувати фотографії до певних витрат: цей засіб може бути корисним для індивідуальних мандрівників або сімейних, які хочуть точно пам'ятати, на що витрачали кошти, або відстежувати пропозиції про поїздки, ділові мандрівники також можуть скористатися цією функцією, сфотографувавши квитанцію, якщо це потрібно для звіту про корпоративні витрати;

- безкоштовна версія обмежує кількість витрат, яку можна внести, в 25;
- платна версія надає можливість експортувати витрати в формат Excel;
- окрім можливості формувати звіти за категоріями витрат, надається також можливість поділитися ними з іншими користувачами;
- програмний засіб безкоштовний, але додаткові функції потребують сплати 1,99 американського долара США за версію для Android, а версія для iOS вартує 4.99 долара [3]-[4].

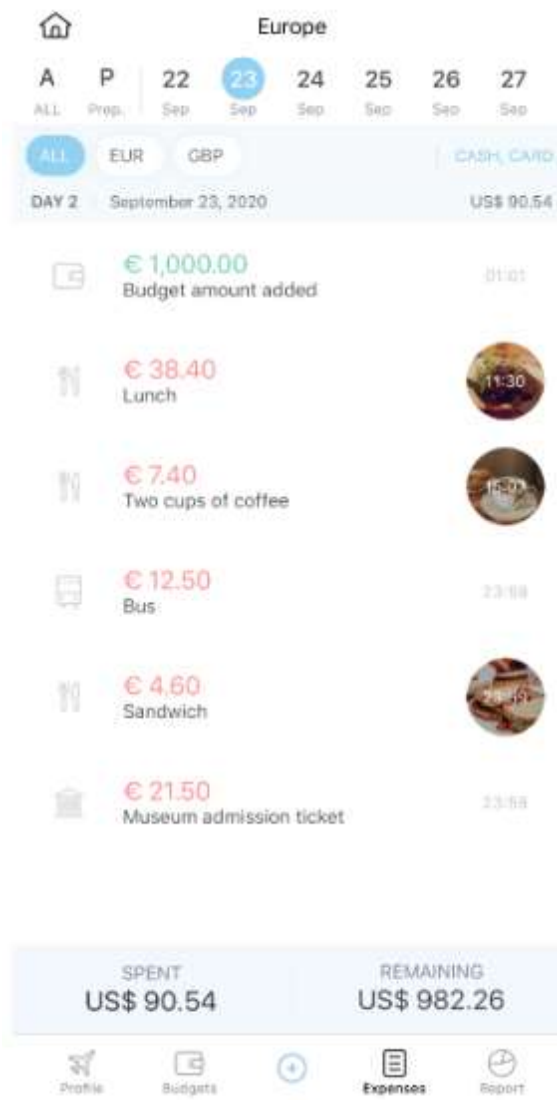


Рисунок 1.1 – Trabee Pocket [2]

При цьому програмний застосунок не надає можливість планування. Тобто фактично відслідковуються витрати, надається можливість аналізу власних результатів у різному розрізі, але при цьому процес планування полишений окремо від програми.

Застосунком, який на даний момент, часто використовуються саме для планування бюджету поїздки є програма Spendee [5] (рис. 1.2). Але насправді ця програма є універсальним інструментом, який дозволяє керувати витратами та планувати бюджет. Відповідно ніяких особливих засобів щодо роботи з витратами саме поїздок тут немає. Тобто фактично саме витрати на подорожі є одним з видів витрат, який дозволяє відслідковувати програма.

Щодо роботи з програмою можна відзначити наступне:

- можна змінювати валюту безпосередньо, додаючи доходи або витрати, отримувати доступ до різних категорій, зберігати фотографії квитанцій та отримувати презентабельну інфографіку, щоб переглянути витрати з часом;
- можна створити бюджет, який допоможе досягти фінансових цілей і який відповідно є основою планування подорожей;
- базова версія програми безкоштовна для iOS та Android, оновлення до преміум-класу дозволяє підключити свій банківський рахунок, тобто стати повноцінним фінансовим трекером, створити декілька гаманців для різних поїздок, що дозволяє реалізувати засоби подібні до групових подорожей, чи проектів та створити необмежений бюджет за 2,99 доларів на місяць [4], [6].

Таким чином, даний програмний застосунок є універсальним для роботи з витратами та доходами, особливостей організації саме поїздок при цьому не враховує.

Trail Wallet [7] є ще одним програмним застосунком з цієї сфери. Щодо нього можна відзначити, що він не безкоштовний, а доступний за 4,99 доларів, безкоштовною є лише пробна версія, її засобів недостатньо для тривалої подорожі, особливо якщо витрати, такі як проживання, розділяються на кожен день подорожі [8].

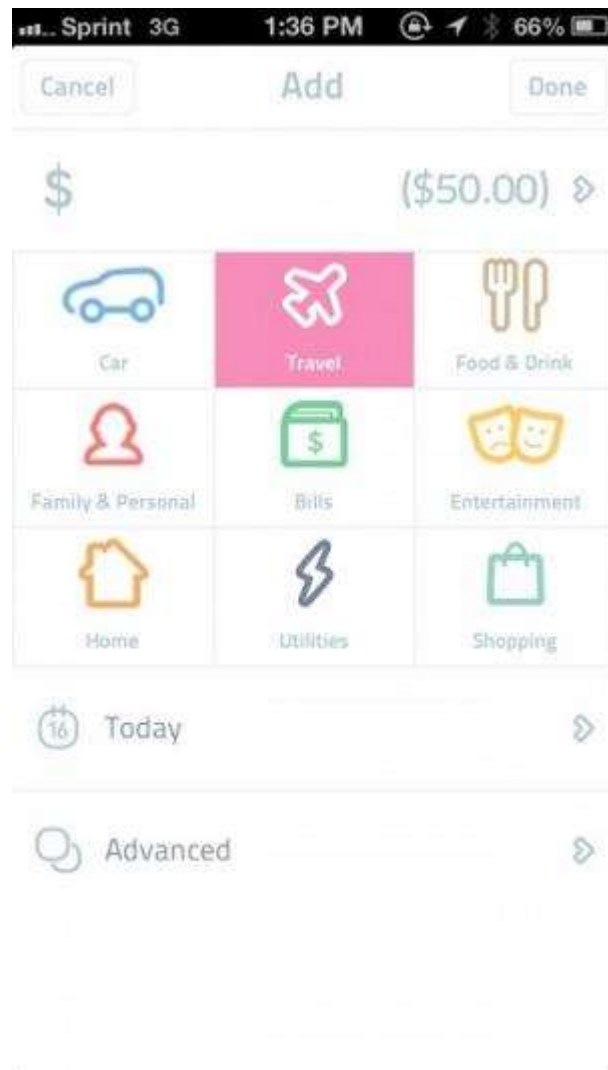


Рисунок 1.2 – Spendee [4]

Програма дозволяє створювати подорожі, визначати валюту подорожі, витрати можуть бути розподілені за днями, тобто щодо фіксації витрат це достатньо зручний інструмент, який до того ж дозволяє визначати план витрат. Але варто відзначити, що щодо фіксації витрат тривалістю в декілька днів, відбувається розбиття витрат на окремі дні, тобто створюються окремі записи, відповідно під час подальшого редагування далі можна загубити потрібні дані і змінити тільки один запис замість цілого набору, який фактично потрібно було змінювати [8].

Щодо кожної подорожі програма визначає (рис. 1.3):

- назву та дату подорожі;
- загальну суму витрачених до цього часу коштів;

- загальний бюджет поїздки;
- загальний бюджет поїздки, що залишився;
- щоденний бюджет;
- середні витрати на день;
- кількість днів у поїзді;
- відсоток витраченого бюджету [8].

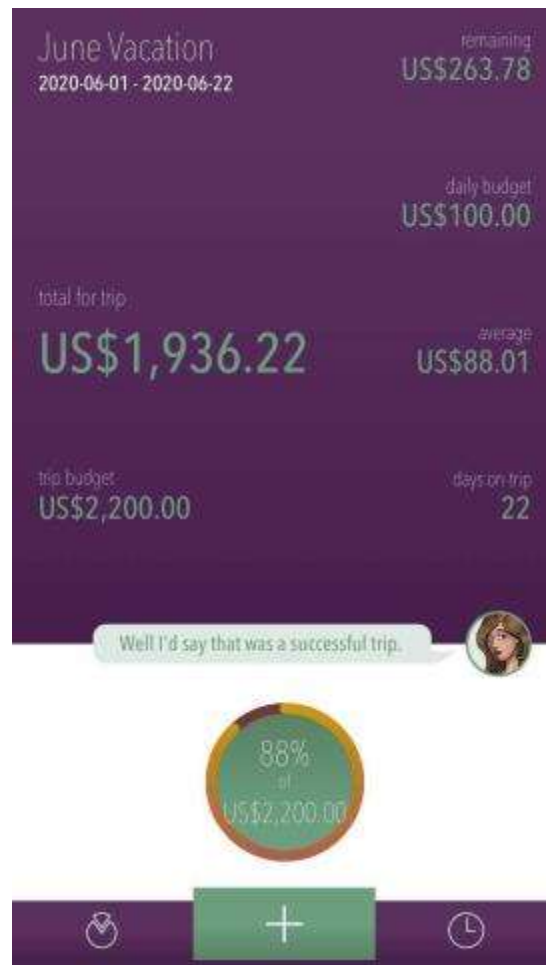


Рисунок 1.3 – Trail Wallet [8]

1.2 Програмні застосунки відслідковування споживчих цін

Окрім того існують вебзастосунки, які дозволяють дізнаватися витрати за статтями у відповідних місцях. Такі інструменти дуже важливі саме для планування бюджету подорожі, адже під час планування користувач

точно не знає свої витрати, відповідно завжди потрібний консультативний механізм, щоб дізнатися потрібні йому обсяги і внести у власний план.

Вебзастосунок Numbeo [9] (рис. 1.4) дозволяє дізнаватися про витрати у заданому місті або країні. Для цього потрібно задати назву міста, а далі буде сформовано перелік основних статей витрат на проживання і для кожної статті зазначено її вартість. Користувач може не тільки отримати інформацію, але і відредагувати її, таким чином користувачі беруть участь у зміні актуальних значень.

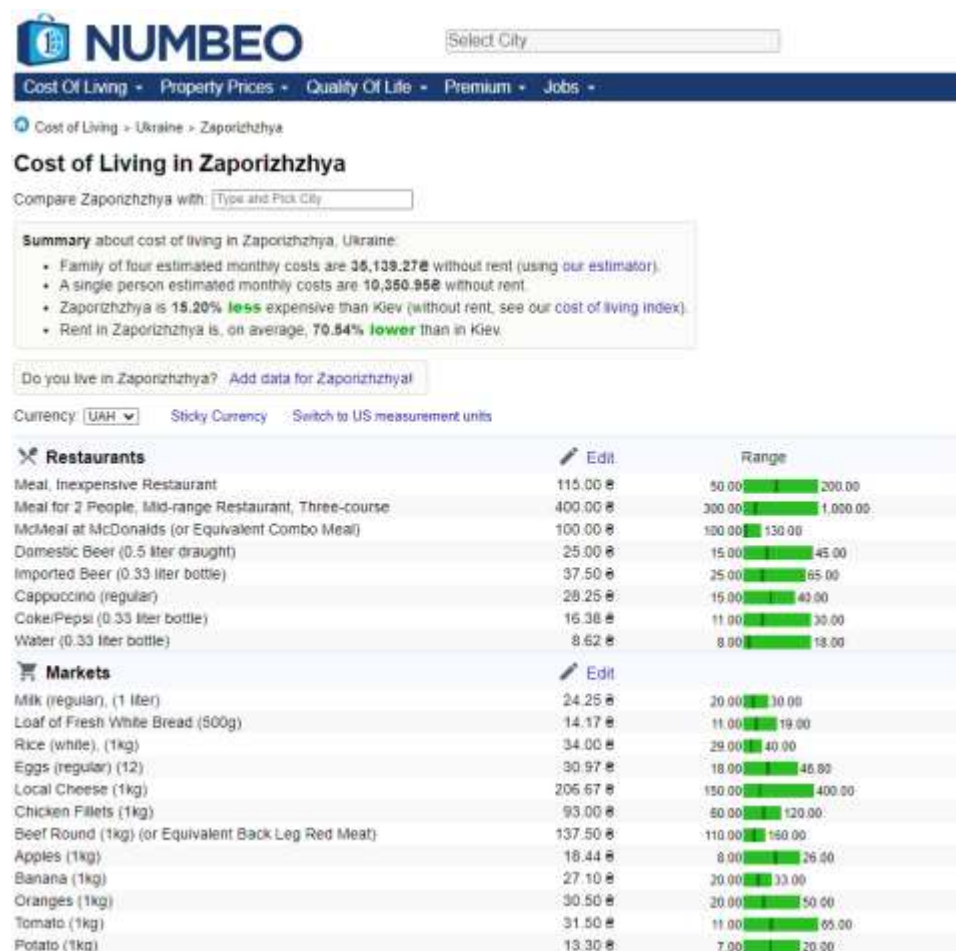


Рисунок 1.4 – Numbeo

Такі інструменти дуже корисні для планування подорожей, але варто відзначити, що виконаний огляд дозволив виявити, що існуючі програми направлені або на планування і відслідковування витрат (не завжди одночасно), або на отримання довідкової інформації щодо витрат на

проживання в місті, відповідно інструменту, який би об'єднував ці функції і спрощував процес планування для користувача, немає.

1.3 Висновки за розділом 1

У процесі аналізу предметної області було проаналізовано програмні застосунки, які можуть використовуватися під час планування бюджету подорожей. Для цього можуть використовуватися з одного боку засоби планування і відслідковування бюджету подорожі, що включають зокрема Trabee Pocket, Trail Wallet, Spendee, які серед інших особливостей відзначаються стягуванням сплати за користування, а з іншого боку – засоби відслідковування споживчих цін в місцях, куди буде здійснюватися подорож. На даний момент немає програмного засобу, який би об'єднував можливості цих двох областей. Його створення має полегшити процес планування для мандрівників.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування і розробки

Розробка програми передбачена у вигляді вебзастосунку для забезпечення доступу з усіх типів пристроїв з наявністю браузера. Одними з найпопулярніших мов програмування для веброзробки є мови С# [10]-[11] і Python [12]-[13], порівняння яких і було виконано (табл. 2.1).

Таблиця 2.1 – Порівняння мов програмування С# і Python

Критерій	С#	Python
Область використання	Веброзробка	Веброзробка
Безкоштовність	+	+
Широка спільнота розробників	+	+
Високорівнева розробка	+	+
Динамічна типізація	—	+
Простота розгортання на вебсервері	+—	+
Динамічна робота над проєктом (включаючи інтерпретатор)	—	+

Для реалізації програмного забезпечення було обрано мову програмування Python, адже вона надає засоби динамічної типізації, дозволяє більш динамічно працювати над проєктом за рахунок наявності

інтерпретатора, що призводить до швидшого створення загального рішення порівняно з необхідністю постійної компіляції компонентів проєкту та за рахунок можливості швидкої перевірки будь-якої частини коду, окрім того сам процес розгортання вебзастосунку значно простіший за використання засобів з платформи, характерної для Python, ніж за використання платформи .NET.

2.2 Визначення функціональних вимог до програмного забезпечення

Основною концепцією програмного забезпечення є забезпечення планування подорожей у вигляді визначення витрат на ці подорожі.

Концепцію можна сформулювати на основі наступних тез:

- витрати на подорож спочатку плануються, а потім мають фіксуватися фактичні значення, що в результаті призводить до створення не тільки плану, але і фактичного набору витрат;

- витрати мають накопичуватися за статтями, але для кожної статті може визначатися цілий набір значень, адже часом сума може розділитися на певні платежі, навіть якщо людина знаходиться в тому самому місці і займається тією ж справою;

- планування надається не тільки для індивідуальних подорожей, але і для подорожей групою людей: у такому випадку передбачається розподілення суми витрат між учасниками, при цьому окремі витрати можуть не розподілятися, а інші розподілятися, тобто не має бути такого, що всі витрати в подорожі підпадають тільки під один принцип;

- планування витрат може здійснюватися на основі значень, встановлених самим користувачем, а можуть використовуватися статистичні значення за даними витратами, збережені в програмі;

– планування і фіксація остаточних витрат має забезпечувати не тільки розподіл витрат між учасниками, але також і можливість поділитися з іншими користувачами своїм власним планом і його фактичною реалізацією.

Остання теза призводить до того, що потенційно з програмою можуть працювати неавторизовані користувачі. Таким користувачам не буде доступним планування, але буде доступним пошук витрат за подорожами користувачів та статистичних значень за місцями.

У результаті програма має забезпечувати виконання таких функцій:

- створення плану нової подорожі;
- додавання учасників до планування подорожі;
- перегляд планів подорожей, у яких користувач є учасником;
- пошук подорожей, у яких користувач є учасником;
- редагування учасників у плануванні подорожі;
- додавання запланованих витрат до подорожей;
- внесення фактичних витрат подорожі;
- визначення публічного доступу до реалізованої подорожі;
- перегляд плану подорожі;
- редагування і вилучення витрат подорожі;
- внесення статистичних витрат;
- редагування статистичних витрат;
- перегляд узагальнених користувацьких статистичних витрат;
- перегляд статистичних витрат;
- перегляд відкритих планів подорожей;
- пошук відкритих планів подорожей;
- обчислення індивідуальних сплат у груповій подорожі;
- авторизація і реєстрація користувачів;
- додавання категорій і груп категорій витрат;
- редагування категорій і груп категорій витрат;
- перегляд категорій і груп категорій витрат;
- перегляд фактичних витрат відкритої подорожі.

Неавторизований користувач (рис. 2.1) може виконувати:

- перегляд узагальнених користувацьких статистичних витрат;
- перегляд статистичних витрат;
- перегляд відкритих планів подорожей;
- пошук відкритих планів подорожей;
- реєстрація;
- перегляд фактичних витрат відкритої подорожі.



Рисунок 2.1 – Діаграма варіантів використання неавторизованого користувача

Адміністратор (рис. 2.2) може виконувати:

- внесення статистичних витрат;
- перегляд статистичних витрат;
- додавання категорій і груп категорій витрат;
- редагування категорій і груп категорій витрат;
- перегляд категорій і груп категорій витрат;
- авторизація.

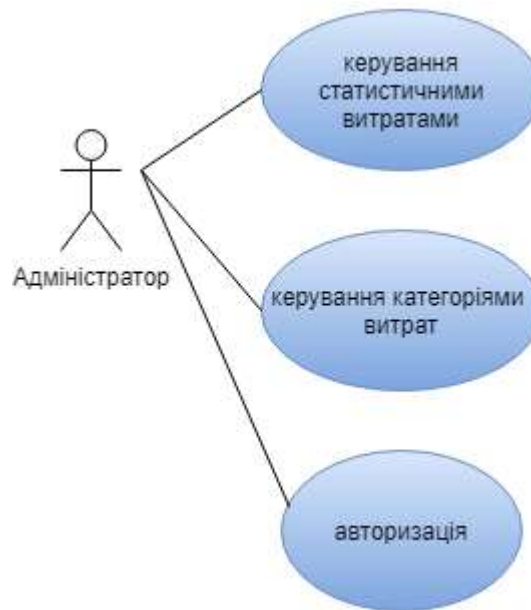


Рисунок 2.2 – Діаграма варіантів використання адміністратора

Авторизований мандрівник, тобто авторизований користувач зі звичайними правами (не адміністратора) може виконувати (рис. 2.3):

- створення плану нової подорожі;
- додавання учасників до планування подорожі;
- перегляд планів подорожей, у яких користувач є учасником;
- пошук подорожей, у яких користувач є учасником;
- редагування учасників у плануванні подорожі;
- додавання запланованих витрат до подорожей;
- внесення фактичних витрат подорожі;
- визначення публічного доступу до реалізованої подорожі;
- перегляд плану подорожі;
- редагування і вилучення витрат подорожі;
- перегляд узагальнених користувацьких статистичних витрат;
- перегляд статистичних витрат;
- перегляд відкритих планів подорожей;
- пошук відкритих планів подорожей;
- обчислення індивідуальних сплат у груповій подорожі;
- авторизація;

– перегляд фактичних витрат відкритої подорожі.

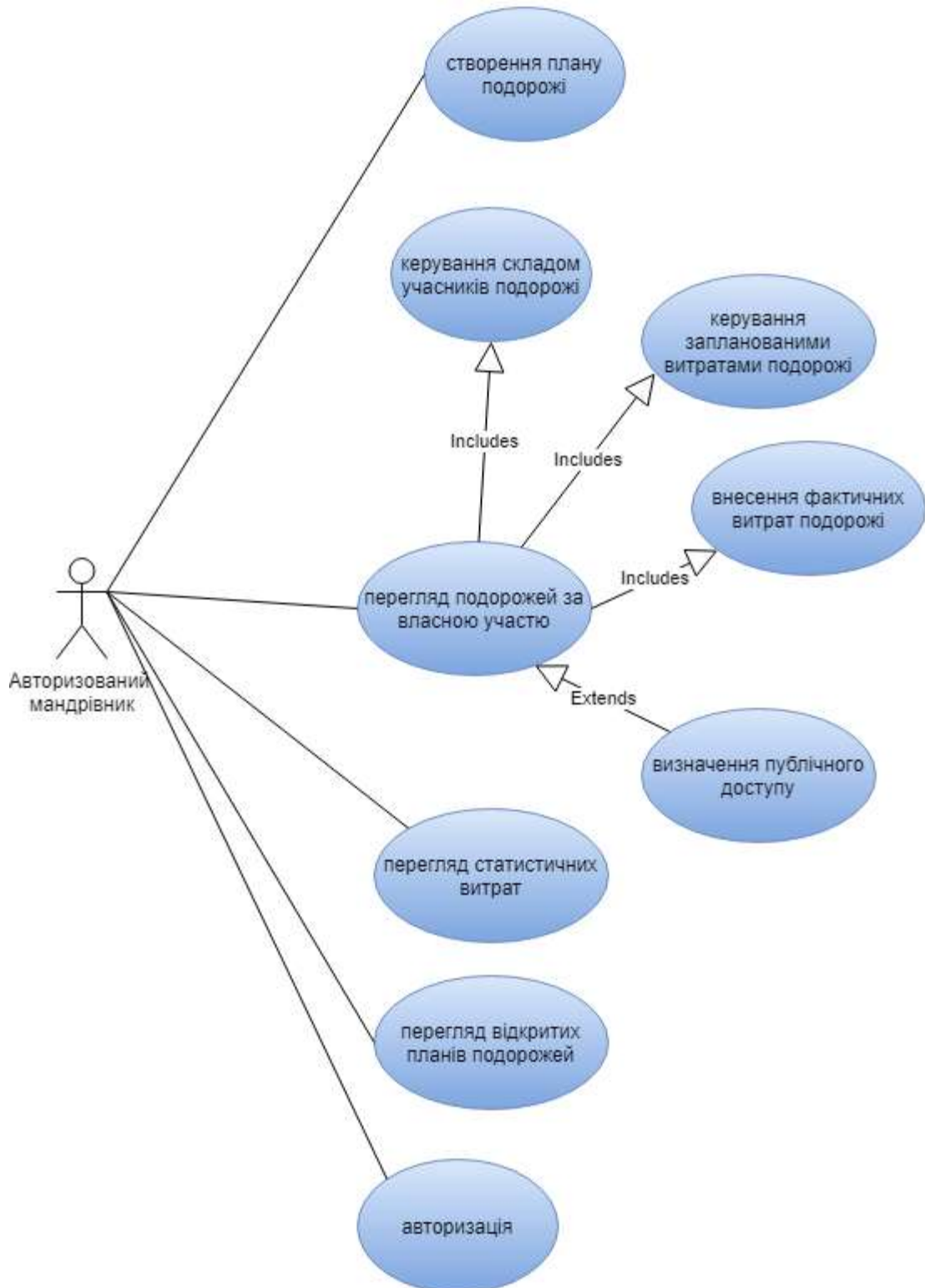


Рисунок 2.3 – Діаграма варіантів використання авторизованого мандрівника

2.3 Визначення архітектури програмного забезпечення

В якості основи для побудови вебзастосунку мовою програмування Python було використано фреймворк веброзробки Django. Вибір саме даного вебфреймворка обумовлено наступними його перевагами:

а) простий: використання фреймворку для розробки є швидким і простим, підключення бібліотек та багаторазове використання означають, що фреймворк дозволяє розробникам використовувати готовий код та перепрофілювати його, за допомогою цього в результаті розробники можуть заощадити час на розробку;

б) легкий: Django побудований на Python, а тому переносить на себе переваги цієї мови, філософія Python наголошує на читабельності коду;

в) всебічний: документація до фреймворку є повною, а також корисною для подальшого застосування, добре написана документація є визначальною характеристикою чудового програмного забезпечення, Django є всебічним тому, що він оснащений більшістю або навіть всіма бібліотеками та інструментами, які можуть коли-небудь знадобитися в процесі розробки, включаючи механізм шаблонів, Django object-relational mapping (ORM), підтримку декількох сайтів, автентифікацію, бібліотеки HTTP тощо;

г) захищений: кібербезпека є великою проблемою для технічних фахівців у наш час, адже втрата персональних даних є критичною, система безпеки Django може запобігати подібним випадкам, міжсайтовим сценаріям та ін'єкціям SQL, своєчасно випускаючи нові виправлення безпеки відносно наступних випадків:

1) clickjacking – акт обману користувача, щоб натиснути щось не те, що здається, що зазвичай призводить до втрати конфіденційної інформації або завантаження шкідливого програмного забезпечення;

2) міжсайтовий сценарій – виникає, коли шкідливі сценарії вводяться на надійні вебсайти;

3) SQL-ін'єкція – техніка введення коду, при якій шкідливі оператори SQL вставляються в застосунки, керовані даними;

д) масштабований: набір інструментів безпеки та повного завантаження Django робить платформу хорошим вибором для підприємств, які хочуть в якийсь момент масштабувати свої програми Django, можна використовувати кластеризацію або балансування навантаження для розподілу програми між серверами:

1) кластеризація – це різновид багатопроцесорної обробки, коли група виконавців, процеси Python, об'єднуються та обробляють завдання, це підвищує ефективність;

2) балансування навантаження призводить до того ж ефекту: у програмі Django балансування навантаження спонукає до методичного розподілу трафіку застосунків на декілька серверів [14].

Django у свою чергу визначає архітектуру вебзастосунку, що складається з наступних компонентів:

- модель (model);
- представлення (view);
- шаблон (template);
- URL-менеджер.

Модель отримала назву Model-View-Template (MVT) на основі взаємодіючих компонентів.

Взаємодія в межах моделі (рис. 2.4) реалізується наступним чином:

- користувач через браузер звертається до Django з запитом;
- Django відправляє URL-запит URL-менеджеру;
- URL-менеджер співставляє запит з шаблонами розпізнавання його, обирає представлення, необхідне для його обробки, та формує до нього HTTP-запит;
- представлення звертається до моделі для отримання необхідних для його роботи даних;

- модель формує запит на зчитування або запис даних до бази даних на основі використання ORM;
- база даних повертає необхідні дані;
- модель перенаправляє отримані дані представленню;
- представлення передає отримані контекстні дані, оброблені ним необхідним чином, шаблону;
- шаблон формує відповідну HTML-сторінку і передає її представленню;
- представлення повертає отриманий результат URL-менеджеру, який перенаправляє його Django, а той у свою чергу відправляє отриману сторінку користувачу для відображення отриманих даних у браузері.

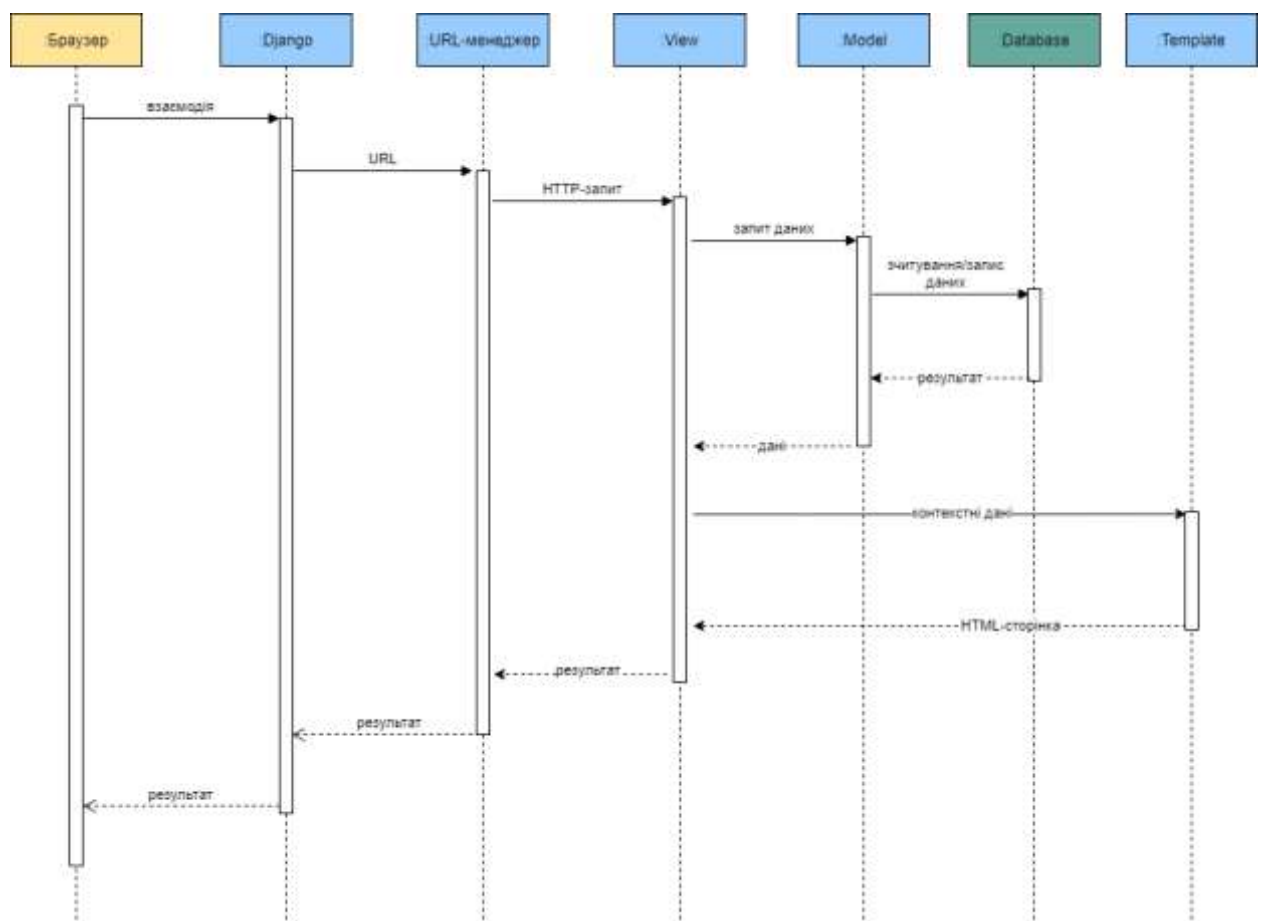


Рисунок 2.4 – Діаграма послідовностей MVT

2.4 Визначення структури бази даних

Системою керування базою даних було обрано MySQL, оскільки вона дозволяє створювати проекти рівня виробництва, є достатньо надійною.

Для структуризації даних було виділено категорії витрат та групи витрат. Групи витрат визначають загальну класифікацію, а категорії більш конкретну. Наприклад, групою витрат може бути «харчування», а категорією – «ресторан», вже «вечеря в ресторані» стане відповідно конкретною витратою.

Таблиця Place представляє місце витрат (місто, селище тощо) і включає поля:

- uniqueId – унікальний номер;
- countryId – країна;
- name – назва місця.

Таблиця Country представляє країну, у подорожі в яку здійснюються витрати, і включає поля:

- uniqueId – унікальний номер;
- name – назва країни.

Таблиця User представляє користувача вебзастосунку (подорожуючий) і включає поля:

- uniqueId – унікальний номер;
- currencyId – базова валюта для користувача;
- login – логін;
- password – пароль;
- profilepic – зображення профілю.

Таблиця Category представляє категорію витрат на подорож і включає поля:

- uniqueId – унікальний номер;
- categGroupId – група категорії витрати;
- name – назва категорії.

Таблиця CategoryGroup представляє групу категорій витрат і включає поля:

- uniqueId – унікальний номер;
- name – назва групи категорії.

Таблиця Trip представляє конкретну подорож, планування якої відбувається, і включає поля:

- uniqueId – унікальний номер;
- name – назва подорожі;
- startDate – дата початку подорожі;
- endDate – дата завершення подорожі;
- membernum – кількість учасників подорожі;
- description – опис;
- mode – режим доступу (необхідний для забезпечення публічного доступу).

Таблиця UserTrip представляє учасників подорожі, планування якої відбувається, реалізує зв'язок багато-до-багатьох між таблицями User і Trip та включає поля:

- uniqueId – унікальний номер;
- userId – користувач (мандрівник);
- tripId – подорож.

Таблиця Costs представляє безпосередньо витрати подорожі, планування якої відбувається, і включає поля:

- uniqueId – унікальний номер;
- categoryId – категорія витрат;
- placeId – місце витрат;
- currencyId – валюта витрати;
- name – назва статті витрат;
- plancosts – заплановані витрати за статтею;
- actcosts – фактичні сумарні витрати за статтею;
- description – опис;

- displaymode – режим відображення витрат;
- calcmode – режим обчислення витрат;
- begDate – дата початку витрат;
- endDate – дата завершення витрат.

Таблиця CostsDetailing представляє конкретні витрати за статтею. Вона необхідна зокрема для тих випадків, коли виконується декілька витрат за статтею. Окрім того таким чином забезпечується підтримка груп подорожуючих, адже таким чином подорожуючий може вносити власні витрати, а вони потім узагальнюються для всіх учасників за статтею. Окрім того таким чином можна потім загальну суму розподілити між учасниками.

Таблиця включає поля:

- uniqueId – унікальний номер;
- costsId – стаття витрат;
- userTripId – користувач (подорожуючий, який сплатив дану витрату);
- currencyId – валюта витрати;
- price – сума витрати;
- mode – режим (дозволяє контролювати, чи потрібно цю суму розподіляти між іншими учасниками, чи це індивідуальна витрата);
- perfDate – дата і час фіксації витрати;
- description – додатковий опис;
- begDate – дата початку витрат;
- endDate – дата завершення витрат.

Таблиця TransportCosts представляє транспортні витрати. Виділення транспортних витрат у окрему таблицю знадобилося, оскільки транспортні витрати передбачають переміщення з одного місця в інше, відповідні така таблиця передбачає два зв'язки з таблицею Place. Звичайні витрати містять тільки один зв'язок.

Таблиця включає поля:

- uniqueId – унікальний номер;

- categoryId – категорія витрат;
- begPlaceId – початковий пункт;
- endPlaceId – кінцевий пункт;
- currencyId – валюта витрати;
- name – назва статті транспортних витрат;
- planCosts – заплановані витрати;
- actCosts – фактичні сумарні витрати;
- displayMode – режим відображення;
- calcMode – режим обчислення;
- description – опис;
- begDate – дата початку витрат;
- endDate – дата завершення витрат.

Таблиця TCOSTSDetailing представляє конкретні транспортні витрати за статтею.

Таблиця включає поля:

- uniqueId – унікальний номер;
- tCostsId – стаття транспортних витрат;
- userTripId – учасник подорожі;
- currencyId – валюта витрати;
- price – сума витрати;
- mode – режим (дозволяє контролювати, чи потрібно цю суму розподіляти між іншими учасниками, чи це індивідуальна витрата);
- perfDate – дата і час фіксації витрати;
- description – додатковий опис;
- begDate – дата початку витрат;
- endDate – дата завершення витрат.

Наявність полів begDate і endDate дозволяє працювати з багатоденними подорожами, уникаючи розбиття по дням, характерного для існуючих програм.

Схема бази даних представлена на рис. 2.5.

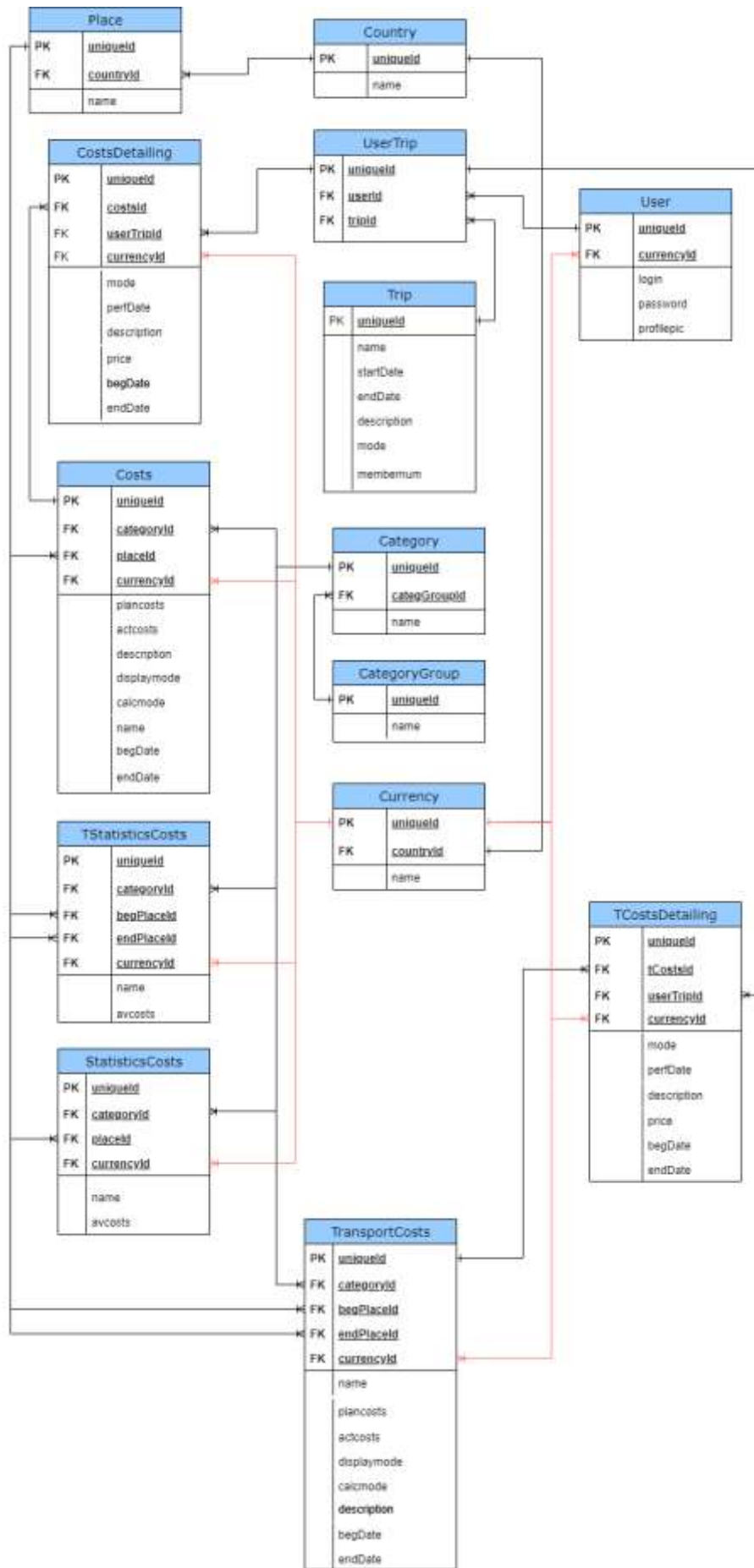


Рисунок 2.5 – Схема бази даних

Таблиця TStatisticsCosts представляє статистику величини транспортних витрат.

Таблиця включає поля:

- uniqueId – унікальний номер;
- categoryId – категорія витрат;
- begPlaceId – початковий пункт;
- endPlaceId – кінцевий пункт;
- currencyId – валюта витрати;
- name – назва статті;
- avcosts – середнє значення витрат (задане статично адміністратором, обчислення з внесених витрат виконується автоматично і в базі даних не зберігається).

Таблиця StatisticsCosts представляє статистику величини витрат і включає поля:

- uniqueId – унікальний номер;
- categoryId – категорія витрат;
- placeId – місце витрат;
- currencyId – валюта витрати;
- name – назва статті;
- avcosts – середнє значення витрат.

Таблиця Currency представляє валюту, в якій обчислюються витрати, і включає поля:

- uniqueId – унікальний номер;
- countryId – країна валюти;
- name – скорочене позначення валюти.

2.5 Висновки за розділом 2

У даному розділі було прийнято рішення обрати мову програмування Python для реалізації програми, яку було вирішено розробляти у вигляді

вебзастосунку. Виконано вибір системи керування базами даних MySQL та вебфреймворку Django для проектування та реалізації. Визначено структуру бази даних.

3 РОЗРОБКА ПРОГРАМИ

3.1 Організація шару роботи з даними

Одними з основних класів моделей, які реалізують основні сутності предметної області, є класи моделей Trip, Costs, CostsDetailing.

Клас моделі Trip визначає план подорожі, як індивідуальної, так і групової, і включає такі поля:

- name сформоване як текстове поле на основі models.CharField максимальною довжиною 200 символів і визначає назву подорожі;
- startDate сформоване як поле дати models.DateField і визначає дату початку подорожі;
- endDate сформоване як поле дати models.DateField і визначає дату завершення подорожі;
- membernum сформоване як цілочисельне поле models.IntegerField і визначає кількість учасників подорожі;
- description сформоване як текстове поле models.CharField максимальною довжиною 255 символів і визначає додатковий опис плану подорожі;
- mode сформоване як цілочисельне поле models.IntegerField і визначає режим відображення плану подорожі;
- currency є зовнішнім ключем models.ForeignKey, що представляє зв'язок з таблицею Currency, тобто визначає базову валюту подорожі, до якою будуть зводитися всі загальні обчислення;
- users визначає перелік користувачів, які беруть участь у плануванні подорожі, представляє собою перелік, реалізований шляхом зв'язку багато-до-багатьох models.ManyToManyField на основі класу User.

Поле mode може дорівнювати одному зі значень: 0 – не призначена для відображення публічно, 1 – відкрита.

Поле membernum необхідне з одного боку для того, щоб не обчислювати кожен раз кількість учасників подорожі, а з іншого боку

оскільки інші учасники можуть бути не представлені в програмі. Відповідно кількість зв'язаних учасників через поле `users` може бути меншою ніж їх фактична кількість, тому для виконання коректних обчислень введено дане поле.

Окрім полів даної моделі було реалізовано методи.

Метод `__str__(self)` дозволяє працювати з об'єктом даного класу шляхом передачі параметру до функції `str`, тобто шляхом переведення до текстового вигляду, в результаті повертає назву подорожі.

Метод `get_absolute_url(self)` повертає абсолютний шлях до сторінки подорожі, включаючи її номер.

Клас моделі `Costs` визначає заплановані статті витрат подорожі і включає такі поля:

- `name` сформоване як текстове поле `models.CharField`, що визначає назву статті витрат, максимальною довжиною 200 символів;
- `trip` є зовнішнім ключем `models.ForeignKey`, що представляє зв'язок з таблицею `Trip`, тобто визначає конкретну подорож, до якої відноситься витрата;
- `category` є зовнішнім ключем `models.ForeignKey`, що представляє зв'язок з таблицею `Category`, тобто визначає конкретну категорію, до якої відноситься витрата;
- `place` є зовнішнім ключем `models.ForeignKey`, що представляє зв'язок з таблицею `Place`, тобто визначає конкретне місце, де було виконано витрату (місто, курорт, селище тощо);
- `currency` є зовнішнім ключем `models.ForeignKey`, що представляє зв'язок з таблицею `Currency`, тобто визначає валюту статті витрат, до якої має бути зведено всі фактичні витрати;
- `description` є текстовим полем `models.CharField` максимальною довжиною 255 символів для формування додаткового опису витрат;
- `begDate` є полем дати `models.DateField` і визначає дату початку витрати (для багатоденної витрати) або день витрати;

- `endDate` є полем дати `models.DateField` і визначає дату завершення витрати (для багатоденної витрати);
- `calcmode` є цілочисельним полем `models.IntegerField`, яке визначає режим обчислення витрати;
- `displaymode` є цілочисельним полем `models.IntegerField`, яке визначає режим відображення цієї статті витрат;
- `currplancosts` є дійним полем `models.DecimalField`, що включає 2 знаки після коми, представляє заплановані витрати, виражені у валюті цієї статті;
- `curractcosts` є дійним полем `models.DecimalField`, що включає 2 знаки після коми, представляє фактичні витрати, виражені у валюті цієї статті;
- `plancosts` є дійним полем `models.DecimalField`, що включає 2 знаки після коми, представляє заплановані витрати, тобто заповнюється на етапі планування подорожі;
- `actcosts` є дійним полем `models.DecimalField`, що включає 2 знаки після коми, представляє сумарні фактичні витрати, при цьому в суму включаються тільки витрати, які розділяються між групою, а персональні сюди не рахуються.

Поле `displaymode` може приймати значення: 0 – витрати видні тільки всередині групи, 1 – витрати видні тільки самому користувачу, який вніс фактичні витрати, 2 – витрати видні всім.

Поле `calcmode` може приймати значення: 0 – поділити витрати між всіма учасниками, 1 – персональні витрати, 2 – залежить від показника, вказаного в деталях (фактичних витратах).

Клас моделі `CostsDetailing` визначає фактичні витрати подорожі і включає такі поля:

- `costs` – зовнішній ключ `models.ForeignKey` до таблиці `Costs`, що визначає статтю запланованих витрат, до якої належать фактична трата;

- user – зовнішній ключ models.ForeignKey до таблиці User, що визначає користувача, який здійснив трату (для групових подорожей визначає, що фактично саме цей учасник оплатив дану суму);
- currency – зовнішній ключ models.ForeignKey до таблиці Currency, що визначає валюту, в якій здійснено витрату;
- currprice – вартість у валюті, визначеній полем currency;
- price – дійсне поле models.DecimalField з точністю до 2 знаків після коми, що визначає розмір трати в базовій валюті подорожі;
- mode – цілочисельне поле models.IntegerField(), що визначає режим обчислення витрати;
- perfDate поле дати models.DateField() – дата здійснення витрати;
- description – символічне поле опису трати models.CharField максимальною довжиною 255 символів;
- begDate – поле дати початку трати models.DateField;
- endDate – поле дати завершення трати models.DateField.

Поле mode може приймати значення: 0 – не ділити витрату (персональна), 1 – ділити між учасниками групи, 2 – не відображати трату взагалі (персональна прихована).

Аналогічно передбачені класи моделей для транспортних витрат TransportCosts, TCostsDetailing. Також створено класи моделей статистики транспортних витрат TStatisticsCosts, статистики інших витрат StatisticsCosts, країни Country, місця Place, групи категорій CategoryGroup, категорії Category, користувача User, валюти Currency.

3.2 Організація шару відображення

Для створення відображення даних користувачу важливим елементом є представлення, оскільки вони здійснюють підготовку даних для відображення користувачу.

Клас представлення `PublicTripsView` визначає підготовку даних для взаємодії користувача через відповідний шаблон з переліком відкритих планів подорожей. Клас створений шляхом наслідування класу `ListView`, що визначає відображення моделей у вигляді переліків.

Оскільки для відображення даних подорожей потрібно не тільки дані подорожі мати, але і дані, представлені моделями витрат, а окрім того ще відображення залежить від параметрів пошуку, заданих користувачем, тобто спочатку відображаються всі відкриті подорожі, а потім їх перелік формується таким, щоб відповідати обраному місту або країні. Для цього було перевизначено метод `get_queryset(self, *args, **kwargs)`.

Спочатку відбувається виділення параметрів пошуку через POST-запит. Наприклад, визначення місця, через яке мала відбуватися подорож, реалізується наступним чином:

```
plc = self.request.POST.get('place')
plc = po.get(name = plc)
```

Далі реалізується виділення подорожей, витрати яких містять задане через виділений вище параметр місце:

```
po = Place.objects.all()
ptrips2 = []
for t in ptrips:
    jd = False
    csts = Costs.objects.all()
    csts = csts.filter(trip = t)
    for c in csts:
        if c.place == plc:
            jd = True
    if jd:
```

`ptrips2.append(t)`

Клас представлення `PublicTripView` готує дані для детального відображення відкритої подорожі та її витрат, створений за допомогою наслідування класу `DetailView`. У даному випадку забезпечується групування всіх витрат за групами категорій, а всередині них – за категоріями. Це дозволяє в результаті відображати не тільки загальну інформацію про кожну подорож, але і мати згруповані за категоріями витрати. При цьому до витрат відносяться тільки витрати, які дозволені для публічного доступу, а загальні суми відображаються у відповідності зі збереженими для розподілу в групі, тобто в таблиці `Costs`.

Клас представлення `StatCostsView` визначає підготовку даних для відображення статистичних витрат за статтями на подорож у визначене користувачем місце, створений шляхом наслідування класу `ListView`.

Клас представлення `OwnTripsView` визначає підготовку даних для взаємодії користувача через відповідний шаблон з переліком власних планів подорожей, тобто подорожей у яких він визначений як учасник. Клас створений шляхом наслідування класу `ListView`, що визначає відображення моделей у вигляді переліків, та класу `LoginRequiredMixin`, що визначає необхідність авторизації користувача.

Клас представлення `OwnTripView` готує дані для детального відображення власної подорожі та її витрат, створений за допомогою наслідування класів `DetailView` та `LoginRequiredMixin`.

Клас представлення `TripNewView` виконує роботу з даними щодо створення нового плану подорожі за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TripEditView` виконує роботу з даними щодо редагування даних існуючого плану подорожі за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `CategoryGroupNewView` виконує роботу з даними щодо створення нової групи категорій адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `CategoryGroupEditView` виконує роботу з даними щодо редагування існуючої групи категорій адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `CategoryNewView` виконує роботу з даними щодо створення нової категорії адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `CategoryEditView` виконує роботу з даними щодо редагування існуючої категорії адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `TStatCostsNewView` виконує роботу щодо внесення нових даних про статистику транспортних витрат між деякими містами адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TStatCostsEditView` виконує роботу щодо редагування даних про статистику транспортних витрат між деякими містами адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `StatCostsNewView` виконує роботу щодо внесення нових даних про статистику витрат у деякому місті адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `StatCostsEditView` виконує роботу щодо редагування даних про статистику витрат у деякому місті адміністратором за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `CategoryGroupView` визначає підготовку даних для взаємодії адміністратора з переліком раніше внесених груп категорій витрат, створений шляхом наслідування класів `ListView` і `LoginRequiredMixin`.

Клас представлення `CategoryView` визначає підготовку даних для взаємодії адміністратора з переліком раніше внесених категорій витрат, створений шляхом наслідування класів `ListView` і `LoginRequiredMixin`.

Клас представлення `ProfileView` готує дані для детального відображення профілю авторизованого користувача, створений за допомогою наслідування класів `DetailView` та `LoginRequiredMixin`.

Клас представлення `AuthorizeView` забезпечує роботу над формою авторизації облікового запису користувача, побудований за допомогою наслідування класу `FormView`.

Клас представлення `UserNewView` виконує роботу щодо внесення даних про новий обліковий запис (реєстрацію) за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `ProfileEditView` виконує роботу щодо редагування даних власного профілю користувачем за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `TripCostNewView` виконує роботу з даними щодо внесення нової статті витрат до плану учасником подорожі за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TripCostEditView` виконує роботу з даними щодо редагування існуючої статті витрат до плану учасником подорожі за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `TripTCostNewView` виконує роботу з даними щодо внесення нової статті транспортних витрат у план подорожі учасником подорожі за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TripTCostEditView` виконує роботу з даними щодо редагування існуючої статті транспортних витрат у плані подорожі учасником подорожі за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `TripDetailCostNewView` виконує роботу з даними щодо внесення нових фактичних витрат за запланованою статтею витрат учасником подорожі, організований за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TripDetailCostEditView` виконує роботу з даними щодо редагування раніше внесених фактичних витрат за запланованою статтею витрат учасником подорожі, організований за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `TripDetailTCostNewView` виконує роботу з даними щодо внесення нових фактичних транспортних витрат за запланованою статтею витрат учасником подорожі, організований за допомогою наслідування класів `LoginRequiredMixin` і `CreateView`.

Клас представлення `TripDetailTCostEditView` виконує роботу з даними щодо редагування раніше внесених фактичних транспортних витрат за запланованою статтею витрат учасником подорожі, організований за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

Клас представлення `UserTripChangeView` виконує роботу з даними щодо змінювання учасників подорожі, додаючи або вилучаючи учасника з подорожі, якщо він вже був доданий раніше, організований за допомогою наслідування класів `LoginRequiredMixin` і `UpdateView`.

На рис. 3.1 представлена діаграма класів основних представлень, які було розроблено, а також класів, які були наслідувані при їх створенні. У наслідуваних класів приведено основні поля та методи, оскільки кожен з них має велику кількість таких полів та методів. У розроблених класах приведено тільки ті поля і методи, які були для них перевизначені.

Безпосередньо шаблони відповідають за відображення отриманих з представлення даних у конкретних позиціях розмітки вебсторінки, яку по суті і репрезентує шаблон. Шаблони створено на основі використання безкоштовного HTML-шаблону Wave Cafe [15].

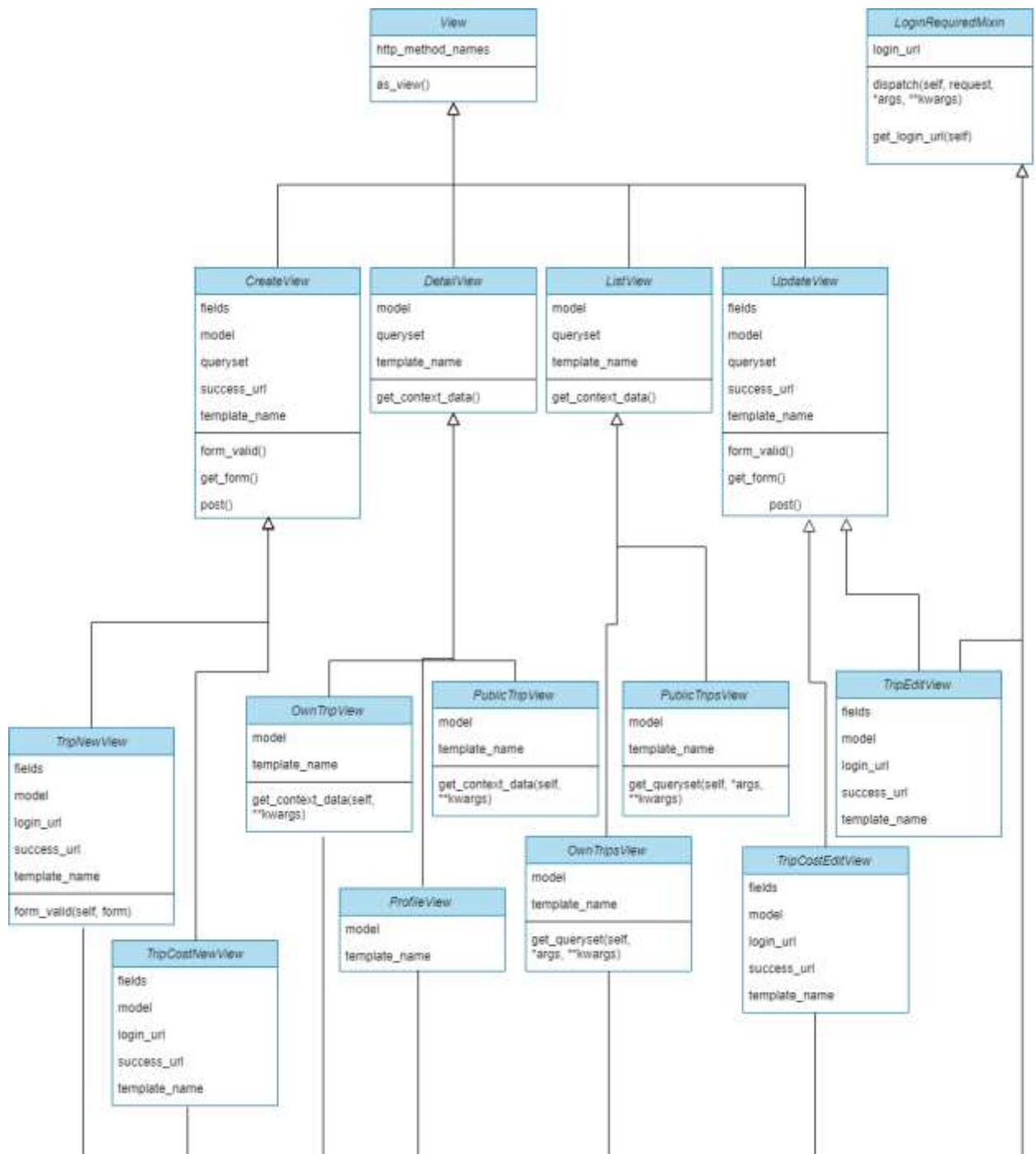


Рисунок 3.1 – Діаграма класів представлень вебзастосунку

3.3 Висновки за розділом 3

У даному розділі представлено і описано розроблені класи моделей, представлень вебзастосунку, принципи створених шаблонів, представлено діаграму класів розроблених представлень.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Призначенням програми є планування бюджету подорожей та відслідковування їх фактичних витрат, для цього забезпечується можливість працювати над плануванням бюджету як індивідуальних, так і групових подорожей, які можуть охоплювати і декілька країн, а також над відслідковуванням фактичних витрат.

4.2 Умови виконання програми

Для роботи клієнтської сторони програмної системи має бути забезпечено роботу сучасного браузера, зокрема:

- Chrome;
- Opera;
- Firefox;
- Internet Explorer версії не нижче 10+.

Також вимагається наявність підключення до мережі інтернет для з'єднання з сервером.

Мінімальні вимоги до вебсервера:

- двохядерний процесор;
- обсяг оперативної пам'яті – 1 Гб;
- вільний обсяг на жорсткому диску – 1 Гб.

Рекомендовані вимоги до вебсервера:

- чотирьохядерний процесор;
- обсяг оперативної пам'яті – 16 Гб;
- вільний обсяг на жорсткому диску – 2 Гб.

4.3 Тестування програми

Тестування програми було реалізовано на пристроях під керуванням операційної системи Windows 10 (персональний комп'ютер), Android 11 (смартфон) та емульованих пристроях Android версій з 8 по 12 в браузерх Chrome, Mozilla Firefox, Internet Explorer, Opera.

Проводились наступні види тестування:

- тестування верстки;
- функціональне тестування;
- тестування юзабіліті;
- конфігураційне тестування.

Проведене тестування на даних пристроях не виявило проблем: уся функціональність була в запланованому вигляді коректно відображена користувачу з усіх цих пристроїв.

Під час виконання тестування проводилась перевірка за основним ланцюжків взаємодії користувача з програмою, що передбачав розміщення вебзастосунку на сервері під керуванням вебсервера Apache. Після його запуску виконувались запити за адресою вебсервера.

При початковому доступі до вебсервера за замовчуванням відображається сторінка для неавторизованого користувача (рис. 4.1).

Сторінка неавторизованого користувача включає наступні пункти меню:

- «Плани» для відображення та пошуку всіх відкритих подорожей;
- «Статистика» для відображення статистики щодо витрат;
- «Авторизація» для авторизації в застосунку.

Пункт меню «Плани» відображає усі відкриті подорожі, розташовані в алфавітному порядку за назвою. На створений перелік можна вплинути, визначаючи назву країни та міста, через яке проходила подорож, тобто в якому була визначена хоча би одна витрата. Для підтвердження вибору країни потрібно після введення в полі натиснути на кнопку «Країна», потім

вказати назву міста і натиснути на кнопку «Місто», тоді перелік подорожей нижче зміниться.

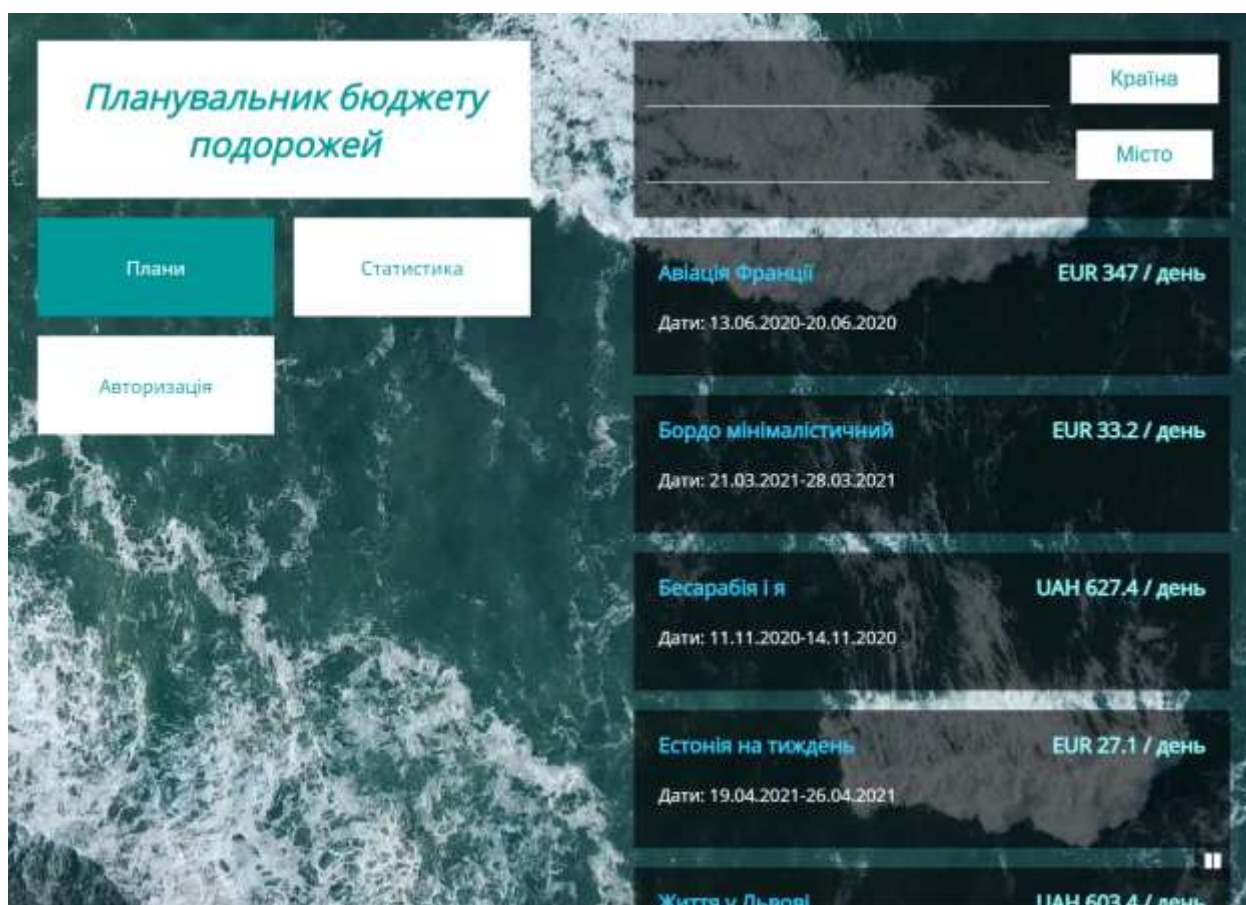


Рисунок 4.1 – Пошук відкритих подорожей

Для кожної подорожі відображається її назва, дати та обсяг витрат на день (всі групові витрати діляться на кількість днів та кількість учасників).

Будь-яку подорож можна деталізувати (до рівня витрат, без застосування деталей), натиснувши на її назву, після цього відкриється окрема сторінка.

При виборі пункту меню «Статистика» відкривається сторінка (рис. 4.2), де можна переглянути дані щодо витрат за категоріями. Для цього потрібно спочатку обрати варіант відображення витрат (розширений – відображати також обчислені з фактичних подорожей витрати або загальний – відображати тільки значення внесені адміністратором в базу даних). Потім потрібно задати країну та місто, де відбувались витрати,

обрати групу категорій витрат, натиснути на кнопку «Обрати», тоді в переліку нижче відобразяться всі категорії для даного міста з визначенням значення витрат з бази даних та обчисленими витратами з розрахунку на день на одну людину.

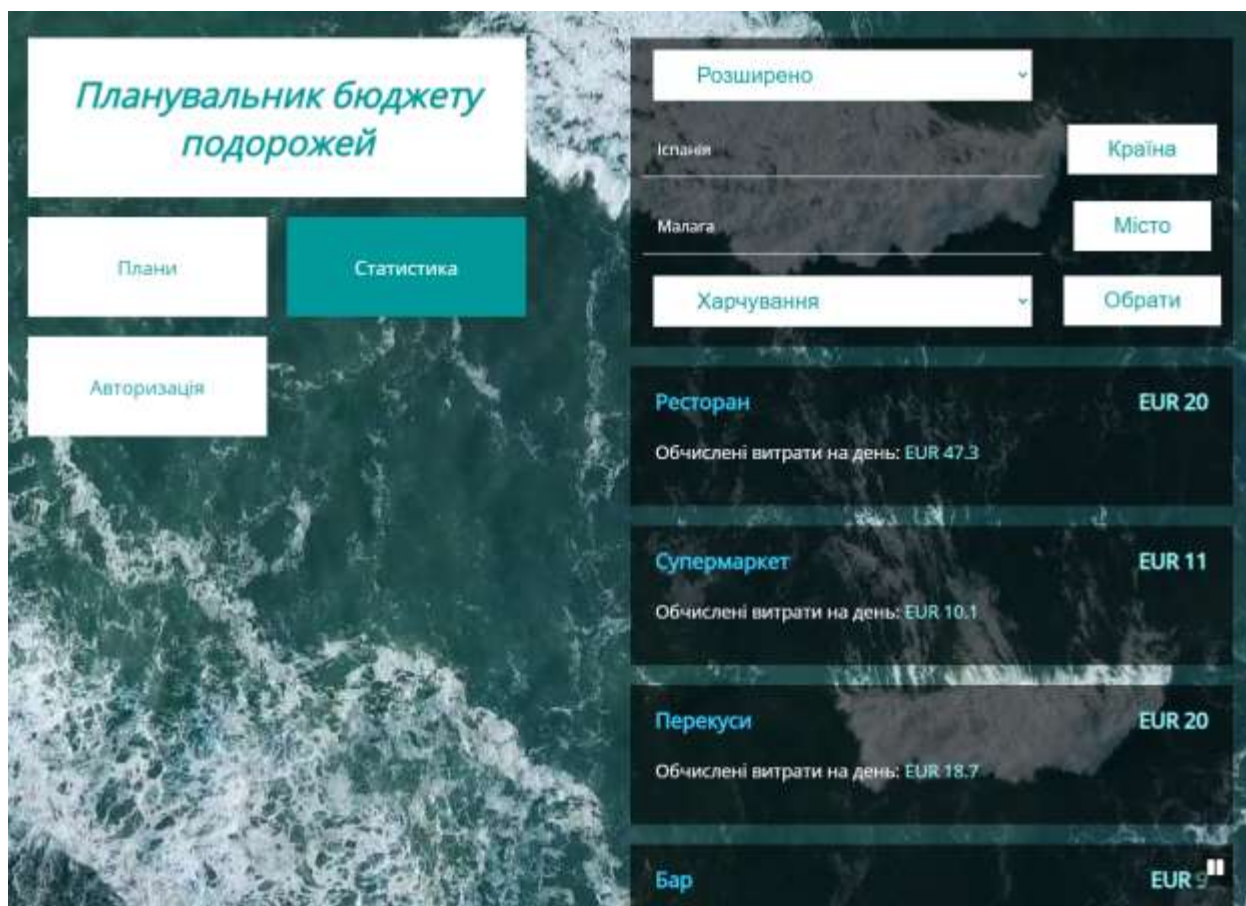


Рисунок 4.2 – Статистика витрат

Після авторизації адміністратора доступні пункти меню:

- «Категорії» для внесення категорій та груп категорій, а також редагування існуючих;
- «Статистика» для внесення статистичних даних за витратами в містах, які потім відображаються основними (не обчисленими) в розділі «Статистика» для користувачів.

Після авторизації звичайному користувачу доступні пункти меню:

- «Мої подорожі», де можна переглянути перелік власних планів подорожей та керувати ними;

- «Відкриті плани», де можна ознайомитись з планами інших користувачів, відкритими для інших;
- «Статистика» для відображення статистики щодо витрат;
- «Профіль», де можна редагувати власні дані, та вийти з облікового запису;
- «Створити подорож» для створення нового плану подорожі.

Після авторизації користувачу спочатку відображається сторінка перегляду власних подорожей за пунктом меню «Мої подорожі» (рис. 4.3).

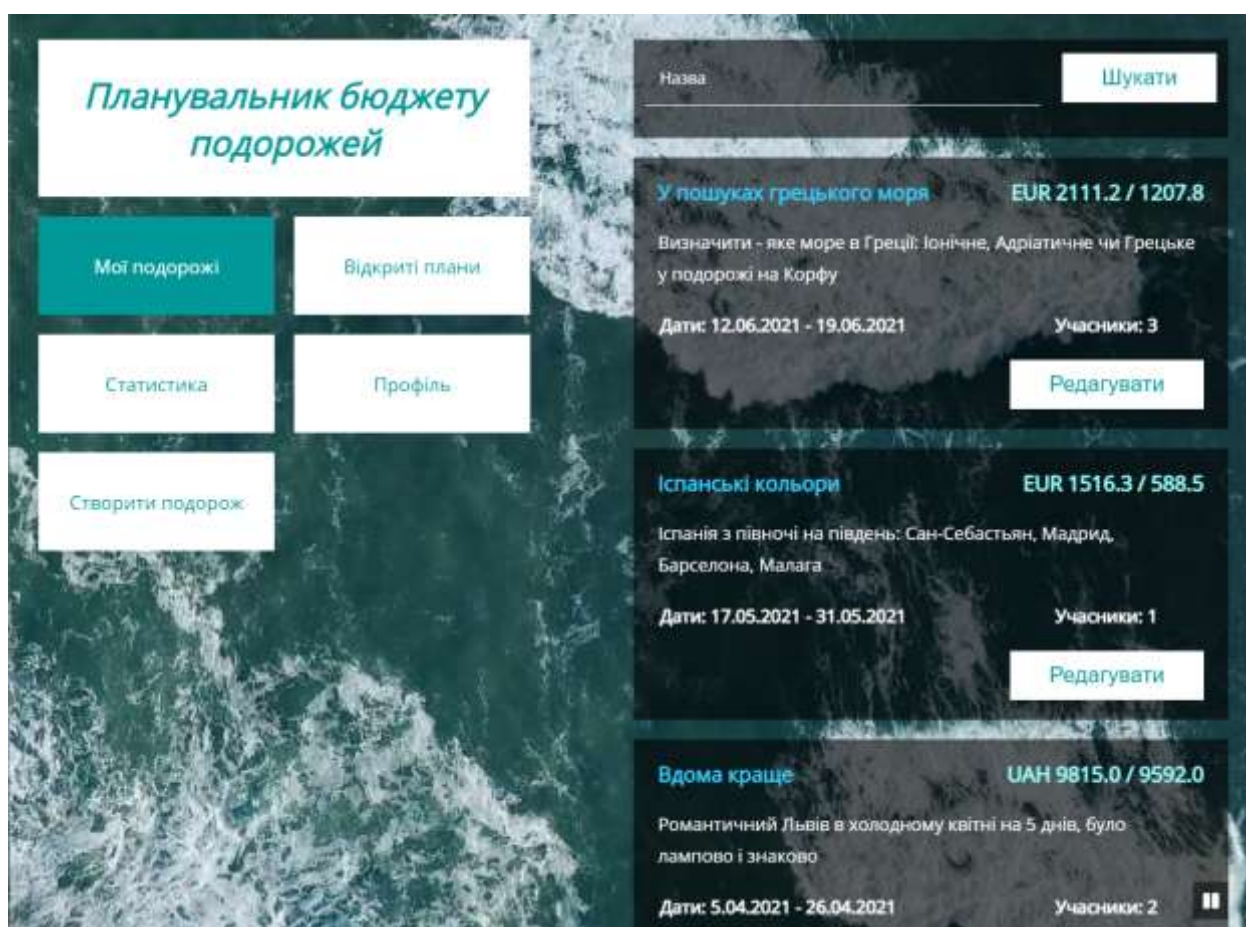


Рисунок 4.3 – Перегляд власних подорожей

Вводячи назву подорожі або її частину можна відобразити відповідні знайдені плани в переліку. Для кожного плану відображається його назва, опис, дати подорожі, кількість учасників та кнопка для виконання редагування (загальних елементів – назви, опису, дат). Якщо між внесеними

фактичними витратами користувача та розрахованими шляхом розподілення загальних між всіма учасниками є різниця і вона негативна, то відображається рядок з інформацією про те, яку суму залишилось сплатити користувачу за цією подорожжю.

Якщо натиснути на назву плану подорожі, то відкриється окрема сторінка (рис. 4.4).

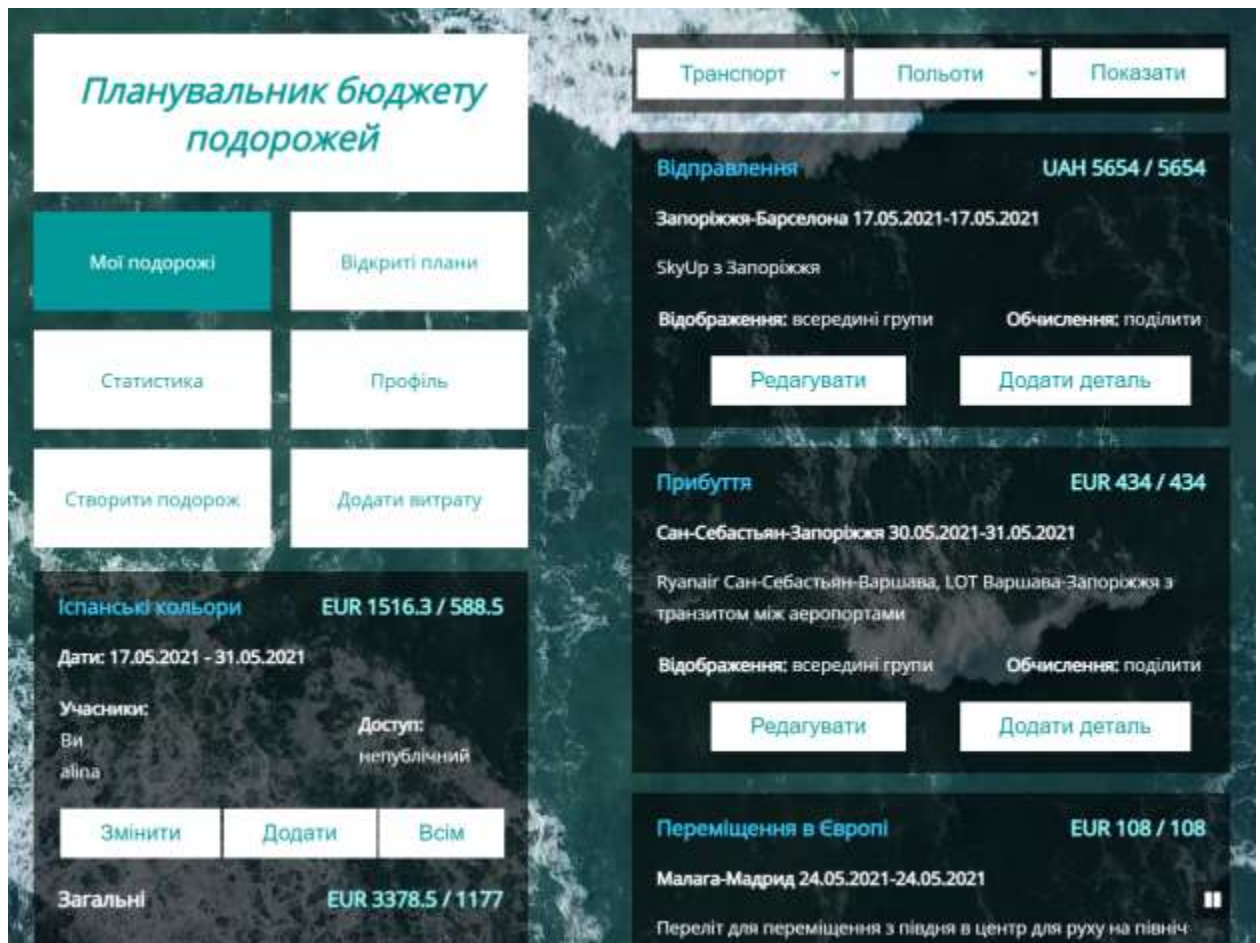


Рисунок 4.4 – Робота з витратами власної подорожі за категоріями

На цій сторінці в лівій нижній частині відображається інформація про обрану подорож. У даному випадку до подорожі було додано другого учасника. В цій частині у верхньому рядку відображаються індивідуальні заплановані та через дріб фактичні витрати, а у останньому рядку – загальні витрати, тобто витрати всіх учасників подорожі.

Кнопка «Змінити» дозволяє вилучити учасника з подорожі. Кнопка «Додати» дозволяє додати нового учасника. Кнопка «Всім» (або «Мені») змінює рівень доступу до плану (публічний або непублічний).

У правій частині відображається перелік витрат. Для цього потрібно спочатку зверху обрати групу витрат з переліку, потім категорію витрат і натиснути на кнопку «Показати». За кожною витратою відображається загальна інформація. Якщо натиснути на кнопку «Редагувати», то відкриється сторінка редагування даних витрати. Звідти можна за необхідності перейти до редагування деталей, тобто окремих фактичних витрат, які сумарно формують дану витрату. Якщо натиснути на кнопку «Додати деталь», то відкриється сторінка внесення фактичної витрати.

На сторінці перегляду витрат подорожі доступна також в меню кнопка «Додати витрату», яку дозволяє внести нову заплановану витрату (рис. 4.5).

The screenshot displays the 'Планувальник бюджету подорожей' (Travel Budget Planner) application. The interface is divided into several sections:

- Header:** 'Планувальник бюджету подорожей' (Travel Budget Planner).
- Navigation Menu (Left):**
 - Мої подорожі (My Trips)
 - Відкриті плани (Open Plans)
 - Статистика (Statistics)
 - Профіль (Profile)
 - Створити подорож (Create Trip)
 - Додати витрату (Add Expense)
- Trip Details (Bottom Left):**
 - Іспанські кольори** (Spanish Colors) - EUR 1516.3 / 588.5
 - Дати:** 17.05.2021 - 31.05.2021
 - Учасники:** Ви, alina
 - Доступ:** непублічний
 - Buttons:** Змінити (Change), Додати (Add), Всім (All)
 - Summary:** Загальні (Total) - EUR 3378.5 / 1177
- Expense Form (Right):**
 - Location:** Іспанія (Spain) - Країна (Country); Мадрид (Madrid) - Місто (City).
 - Category Selection:** Житло (Accommodation), Транспорт (Transport), Харчування (Food), Розваги (Entertainment), Сувеніри (Souvenirs), Додаткові (Additional).
 - Expense Details:**
 - Готель** (Hotel) - всередині групи (within group) - поділити (split)
 - Назва** (Name)
 - Заплановані витрати** (Planned expenses)
 - Дата початку** (Start date) - **Дата завершення** (End date)
 - Опис** (Description)
 - Buttons:** Зберегти (Save), До деталі (To details)

Рисунок 4.5 – Створення запланованої витрати власної подорожі

Під час створення нової витрати плану подорожі потрібно спочатку задати місце, де відбудеться ця витрата, аналогічно описаному вище способу, потім обрати групу категорій витрат (житло, транспорт, харчування, розваги, сувеніри або додаткові). Потім потрібно обрати конкретну категорію витрат (у даному випадку «Готель»), рівень видимості витрати, спосіб урахування витрати в плані, задати назву витрати, вказати запланований обсяг витрат, дату початку та дату завершення, а також за необхідності опис. Для підтвердження збереження внесених даних необхідно натиснути на кнопку «Зберегти».

Якщо натиснути на кнопку «До деталі», то виконається збереження витрати та відкриється сторінка внесення фактичної витрати за цією статтею.

ВИСНОВКИ

Дипломну кваліфікаційну роботу бакалавра присвячено проблемі планування бюджету подорожей.

Проведено аналіз програмних застосунків, що використовуються під час планування бюджету подорожей, на основі чого визначено концепцію створення програмного забезпечення, що ґрунтується на поєднанні можливостей планування і відслідковування витрат з можливостями відслідковування вартості життя в місцях подорожі.

Виконано проєктування програмного забезпечення на основі створення діаграм варіантів використання, послідовностей, класів, схеми бази даних. Для проєктування і розроблення було використано такі засоби як мова програмування Python, вебфреймворк Django та система керування базами даних MySQL.

Реалізовано програмне забезпечення з виділенням основних класів, створених під час визначення моделей, представлень, а також використаних засобів під час створення шаблонів, URL-менеджера, які були основними компонентами програмного забезпечення.

Створений вебзастосунок дозволяє працювати над плануванням бюджету як індивідуальних, так і групових подорожей, додаючи учасників у подорож та надаючи їм можливості внесення як персональних, так і групових витрат у цих подорожах, в результаті програма автоматично розподіляє суми за необхідності між всіма учасниками. Подорожі, планування яких відбувається в програмі, можуть охоплювати і декілька країн, що можливо за рахунок використання засобів визначенні курсу валют. Подорожі можуть бути тривалими, адже для витрат враховано можливість тривалості протягом декількох днів. Програма дозволяє не тільки планувати бюджет, але і відслідковувати фактичні витрати. Витрати розділені на категорії, а категорії об'єднані в групи, що дозволяє потім достатньо універсально фіксувати витрати з розширеними можливостями отримання довідки щодо суми статей

витрат на етапі планування. При цьому мандрівник може використовувати як довідкові дані про витрати, так і вносити власні значення, програма його в цьому не обмежує. Важливим є також те, що довідкові дані можуть і розраховуватися на основі узагальнення досвіду, накопичено іншими користувачами.

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lock, S. Global tourism industry – statistics & facts [Electronic resource] / S. Lock. – Access mode : <https://www.statista.com/topics/962/global-tourism/>
2. Trabee Pocket [Electronic resource]. – Access mode : <https://trabepocket.com/>
3. Yellin, J. 8 apps that will help track your travel expenses [Electronic resource] / J. Yellin. – Access mode : <https://thepointsguy.com/news/apps-track-travel-expense/>
4. Ayoub, C. A. Top 5 Travel Budget Apps to track your Trip Expenses [Electronic resource] / C. A. Ayoub. – Access mode : <https://www.travelevil.com/travel-budget-apps-trip-expenses/>
5. Money Manager & Budget Planner | Spendee [Electronic resource]. – Access mode : <https://www.spendee.com/>
6. Winkler, C. The Best Budgeting and Expense-Tracking Apps For Travelers [Electronic resource] / C. Winkler. – Access mode : <https://toomanyadapters.com/best-budgeting-expense-tracking-apps-travel/>
7. Fairbairn, S. Trail Wallet Travel Budget App [Electronic resource] / S. Fairbairn. – Access mode : <https://apps.apple.com/us/app/trail-wallet-travel-budget-app/id547171665>
8. Nottrodt, J. Review [Electronic resource] : Trail Wallet Travel Budget App for iOS / J. Nottrodt. – Access mode : <https://toomanyadapters.com/review-trail-wallet-app-ios/>
9. Cost of Living [Electronic resource]. – Access mode : <https://www.numbeo.com/cost-of-living/>
10. Skeet, J. C# in Depth [Text] : Fourth Edition / J. Skeet. – New York : Manning Publications, 2019. – 528 p.
11. .NET is Free [Electronic resource]. – Access mode : <https://dotnet.microsoft.com/platform/free>

12. Lutz, M. Programming Python [Text] : Powerful Object-Oriented Programming / M. Lutz. – Sebastopol, CA : O'Reilly Media, 2011. – 1632 p.
13. Lubanovic, B. Introducing Python [Text] : Modern Computing in Simple Packages / B. Lubanovic. – Sebastopol, CA : O'Reilly Media, 2019. – 630 p.
14. What Is Django Used For? 6 Examples of Global Companies Using Django [Electronic resource]. – Access mode : <https://trio.dev/blog/django-applications>
15. Wave Cafe – Free HTML Templates [Electronic resource]. – Access mode : <https://www.tooplate.com/view/2121-wave-cafe>

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення, що розробляється в роботі, має бути представлене вебзастосунком для забезпечення уніфікованого доступу до програми з будь-якого типу пристроїв.

Витрати мають бути розділені на категорії, а категорії об'єднані в групи, що дозволить потім достатньо універсально фіксувати витрати з розширеними можливостями отримання довідки щодо суми статей витрат на етапі планування. Мандрівник повинен мати змогу використовувати як довідкові дані про витрати, так і вносити власні значення, не обмежуючи вибір програмно.

A.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Розробка програмного забезпечення планування бюджету подорожей» і затверджене у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Призначенням розробки є планування бюджету подорожей та відслідковування їх фактичних витрат, для цього має забезпечуватися можливість працювати над плануванням бюджету як індивідуальних, так і групових подорожей, які можуть охоплювати і декілька країн, а також над відслідковуванням фактичних витрат.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма повинна забезпечувати такі функціональні характеристики:

- створення плану нової подорожі;
- додавання учасників до планування подорожі;
- перегляд планів подорожей, у яких користувач є учасником;
- пошук подорожей, у яких користувач є учасником;
- редагування учасників у планування подорожі;
- додавання запланованих витрат до подорожей;
- внесення фактичних витрат подорожі;
- визначення публічного доступу до реалізованої подорожі;
- перегляд плану подорожі;
- редагування і вилучення витрат подорожі;
- внесення статистичних витрат;
- редагування статистичних витрат;
- перегляд узагальнених користувацьких статистичних витрат;
- перегляд статистичних витрат;
- перегляд відкритих планів подорожей;
- пошук відкритих планів подорожей;
- обчислення індивідуальних сплат у груповій подорожі;
- додавання категорій і груп категорій витрат;
- редагування категорій і груп категорій витрат;
- перегляд категорій і груп категорій витрат;
- авторизація і реєстрація користувачів.

A.3.2 Вимоги до надійності

Надійність програмного забезпечення має забезпечуватися шляхом перевірки тих даних, які задає програмі користувач, відповідно потрібні поля

мають бути заповнені, у них мають бути введені коректні значення, включаючи до такого типу перевірок зокрема і SQL-ін'єкції.

A.3.3 Умови експлуатації

Експлуатація програмного забезпечення має виконуватися на основі клієнт-серверної моделі. Для цього потрібно виділити окремий комп'ютер або скористатися виділеним сервером, який буде забезпечувати підтримку реалізації вебсервера. Вебсервер має бути доступним для запитів користувачів постійно.

A.3.4 Вимоги до складу та параметрів технічних засобів

Для роботи клієнтської сторони програмної системи має бути забезпечено роботу сучасного браузера:

- Chrome;
- Opera;
- Firefox;
- Internet Explorer версії не нижче 10.

Також вимагається наявність підключення до мережі інтернет для з'єднання з сервером.

Мінімальні вимоги до вебсервера:

- двохядерний процесор;
- обсяг оперативної пам'яті – 1 Гб;
- вільний обсяг на жорсткому диску – 1 Гб.

Рекомендовані вимоги до вебсервера:

- чотирьохядерний процесор;
- обсяг оперативної пам'яті – 16 Гб;
- вільний обсяг на жорсткому диску – 2 Гб.

А.4 Порядок контролю та приймання

Контроль виконання роботи має здійснюватися за календарним планом. Остаточне приймання дипломної кваліфікаційної роботи має виконуватися після 9 тижнів.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```

from django.db import models
from django.urls import reverse

class Country (models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Place (models.Model):
    name = models.CharField(max_length=100)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

    def __str__(self):
        return self.name

    def full(self):
        return self.name+', '+self.country.name

class CategoryGroup (models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Category (models.Model):
    name = models.CharField(max_length=150)
    categorygroup = models.ForeignKey (CategoryGroup,
on_delete=models.CASCADE)

    def __str__(self):
        return self.name

```

```

def full(self):
    return self.name+' ('+self.category.name+')'

class Trip (models.Model):
    name = models.CharField(max_length=200)
    startDate = models.DateField()
    endDate = models.DateField()
    membernum = models.IntegerField()
    description = models.CharField(max_length=255)
    mode = models.IntegerField()
    users = models.ManyToManyField(User)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('trip', args=[self.id])

class User (models.Model):
    login = models.CharField(max_length=100)
    password = models.CharField(max_length=100)
    currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
    profilepic = models.CharField(max_length=255)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('user', args=[self.id])

class Currency (models.Model):
    name = models.CharField(max_length=3)
    country = models.ForeignKey (Country,
on_delete=models.CASCADE)

    def __str__(self):
        return self.name

```

```

def full(self):
    return self.name+' ('+self.country.name+')'

class Costs (models.Model):
    name = models.CharField(max_length=200)
    trip = models.ForeignKey (Trip, on_delete=models.CASCADE)
    category = models.ForeignKey (Category,
on_delete=models.CASCADE)
    place = models.ForeignKey (Place, on_delete=models.CASCADE)
    currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
    description = models.CharField(max_length=255)
    begDate = models.DateField()
    endDate = models.DateField()
    calcmode = models.IntegerField()
    displaymode = models.IntegerField()
    plancosts = models.DecimalField(max_digits=9,
decimal_places=2)
    actcosts = models.DecimalField(max_digits=9,
decimal_places=2)

    def __str__(self):
        return self.name

class CostsDetailing (models.Model):
    costs = models.ForeignKey (Costs, on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=9, decimal_places=2)
    mode = models.IntegerField()
    perfDate = models.DateField()
    description = models.CharField(max_length=255)
    begDate = models.DateField()
    endDate = models.DateField()

class TransportCosts (models.Model):
    name = models.CharField(max_length=200)

```

```

        trip = models.ForeignKey (Trip, on_delete=models.CASCADE)
        category = models.ForeignKey (Category,
on_delete=models.CASCADE)
        begPlace = models.ForeignKey (Place,
on_delete=models.CASCADE)
        endPlace = models.ForeignKey (Place,
on_delete=models.CASCADE)
        currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
        plancosts = models.DecimalField(max_digits=9,
decimal_places=2)
        actcosts = models.DecimalField(max_digits=9,
decimal_places=2)
        calcmode = models.IntegerField()
        displaymode = models.IntegerField()
        description = models.CharField(max_length=255)
        begDate = models.DateField()
        endDate = models.DateField()

    def __str__(self):
        return self.name

class TCostsDetailing (models.Model):
    tcosts = models.ForeignKey (TransportCosts,
on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
    price = models.DecimalField(max_digits=9, decimal_places=2)
    mode = models.IntegerField()
    perfDate = models.DateField()
    description = models.CharField(max_length=255)

class TStatisticsCosts (models.Model):
    name = models.CharField(max_length=200)
    category = models.ForeignKey (Category,
on_delete=models.CASCADE)
    begPlace = models.ForeignKey (Place,
on_delete=models.CASCADE)

```

```

        endPlace = models.ForeignKey (Place,
on_delete=models.CASCADE)
        currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
        avcosts = models.DecimalField(max_digits=6,
decimal_places=2)

    def __str__(self):
        return self.name + ', '+self.category.name

class StatisticsCosts (models.Model):
    name = models.CharField(max_length=200)
    category = models.ForeignKey (Category,
on_delete=models.CASCADE)
    place = models.ForeignKey (Place, on_delete=models.CASCADE)
    currency = models.ForeignKey (Currency,
on_delete=models.CASCADE)
    avcosts = models.DecimalField(max_digits=6,
decimal_places=2)

    def __str__(self):
        return self.name + ', '+self.category.name

```

Б.2 Текст файла /views/trip.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
from django.views.generic.detail import DetailView
from django.views.generic.list import ListView
import tripplanner.models

class PublicTripsView (ListView):

    model = Trip
    template_name = 'publictrips.html'

    def get_queryset(self, *args, **kwargs):

```

```

        ptrips = super(PublicTripsView,
self).get_queryset(*args, **kwargs)
        ptrips = ptrips.filter(mode = 1)
        if self.request.POST:
            try:
                cntr = self.request.POST.get('country')
                if cntr:
                    po = Country.objects.all()
                    cntr = po.filter(name = cntr)
                    po = Place.objects.all()
                    plcs = po.filter(country = cntr)
                    ptrips2 = []
                    for plc in plcs:
                        for t in ptrips:
                            jd = False
                            csts = Costs.objects.all()
                            csts = csts.filter(trip = t)
                            for c in csts:
                                if c.place == plc:
                                    jd = True
                            if jd:
                                ptrips2.append(t)
                            else:
                                csts =
TransportCosts.objects.all()

                                csts = csts.filter(trip = t)
                                for c in csts:
                                    if c.begPlace == plc or
c.endPlace == plc:

                                        jd = True
                                if jd:
                                    ptrips2.append(t)
                    ptrips = ptrips2
            else:
                plc = self.request.POST.get('place')
                po = Place.objects.all()
                plc = po.get(name = plc)
                ptrips2 = []
                for t in ptrips:
                    jd = False
                    csts = Costs.objects.all()

```

```

        csts = csts.filter(trip = t)
        for c in csts:
            if c.place == plc:
                jd = True
        if jd:
            ptrips2.append(t)
        else:
            csts = TransportCosts.objects.all()
            csts = csts.filter(trip = t)
            for c in csts:
                if c.begPlace == plc or
c.endPlace == plc:
                    jd = True
            if jd:
                ptrips2.append(t)
            ptrips = ptrips2
        ptrips = ptrips.order_by("name")
        return ptrips

class PublicTripView (DetailView):

    model = Trip
    template_name = 'ptrip.html'

    def get_context_data(self, **kwargs):
        tdata = super(TripView, self).get_context_data(**kwargs)
        tr = Trip.objects.get(id=self.kwargs['pk'])
        css = Costs.objects.filter(trip=tr)
        lo = []
        for c in css:
            if c.displaymode == 2:
                lo.append(c)
        catgroups = CategoryGroup.objects.all()
        fullres = []
        for catgroup in catgroups:
            categories = Category.objects.filter(categorygroup =
catgroup)

            reslist = []
            for categor in categories:
                res = lo.filter(category = categ)

```

```

        if res:
            reslist[str(category)] = res
        if reslist:
            fullres[str(catgroup)] = reslist
    if fullres:
        tdata ['costs'] = fullres
    csts = TransportCosts.objects.all()
    csts = csts.filter(trip = tr)
    lo = []
    for c in csts:
        if c.displaymode == 2:
            lo.append(c)
    catgroups = CategoryGroup.objects.all()
    fullres = []
    for catgroup in catgroups:
        categories = Category.objects.filter(categorygroup =
catgroup)

        reslist = []
        for categor in categories:
            res = lo.filter(category = categ)
            if res:
                reslist[str(category)] = res
        if reslist:
            fullres[str(catgroup)] = reslist
    if fullres:
        tdata ['tcosts'] = fullres
    return tdata

```

```

class OwnTripView (LoginRequiredMixin, DetailView):

```

```

    model = Trip

```

```

    template_name = 'otrip.html'

```

```

    def get_context_data(self, **kwargs):

```

```

        tdata = super(TripView, self).get_context_data(**kwargs)

```

```

        tr = Trip.objects.get(id=self.kwargs['pk'])

```

```

        people = len(tr.users)

```

```

        topay = 0

```

```

        paied = 0

```

```

perspaied = 0

css = Costs.objects.filter(trip=tr)
lo = []
for c in css:
    if c.calcmode == 0:
        topay = topay + c.actcosts
        cds = CostsDetailing.objects.filter(costs = c)
        locd = []
        for cd in cds:
            if not (c.displaymode == 1 and cd.user !=
self.request.user) and not (cd.mode == 2 and cd.user !=
self.request.user):
                locd.append(cd)
                if (c.calcmode == 0 or c.calcmode == 2) and
cd.user == self.request.user and cd.mode == 1:
                    paied = paied + cd.price
                    if cd.user == self.request.user and cd.mode == 0
or cd.mode == 2:
                        perspaied = perspaied + cd.price
                c['details'] = locd
                lo.append(c)

catgroups = CategoryGroup.objects.all()
fullres = []
plist2 = []
for catgroup in catgroups:
    categories = Category.objects.filter(categorygroup =
catgroup)

    reslist = []
    plist = []
    sumgcosts = 0
    for categor in categories:
        res = lo.filter(category = categ)
        sumcosts = 0
        if res:
            reslist[str(category)] = res
            for r in res:
                vs = r['details']
                if r.calcmode == 2
                    for v in vs:

```



```

        lo.append(c)

catgroups = CategoryGroup.objects.all()
fullres = []
plist2 = []
for catgroup in catgroups:
    categories = Category.objects.filter(categorygroup =
catgroup)

    reslist = []
    plist = []
    sumgcosts = 0
    for categor in categories:
        res = lo.filter(category = categ)
        sumcosts = 0
        if res:
            reslist[str(category)] = res
            for r in res:
                vs = r['details']
                if r.calcmode == 2:
                    for v in vs:
                        if v.mode == 0 or v.mode == 2:
                            sumcosts = sumcosts +
v.price

                        else:
                            sumcosts = sumcosts +
v.price/people

                else:
                    if r.calcmode == 1:
                        sumcosts = sumcosts + r.price
                    else:
                        sumcosts = sumcosts +
r.price/people

            plist[str(category)] = sumcosts
            sumgcosts = sumgcosts + sumcosts
        if reslist:
            fullres[str(catgroup)] = reslist
            plist2[str(catgroup)] = sumgcosts
if fullres:
    tdata ['tcosts'] = fullres
    tdata ['tottcosts'] = plist
    tdata ['tottgrcosts'] = plist2

```

```

topay = topay / people - paied
tdata ['topay'] = topay
tdata ['paied'] = paied
tdata ['perspaied'] = perspaied
return tdata

```

```
class TripNewView(LoginRequiredMixin, CreateView):
```

```

    template_name = 'tripnew.html'
    model = Trip

```

```

    fields = [
        'name',
        'startDate',
        'endDate',
        'membernum',
        'description',
        'mode',
        'users'
    ]

```

```

    login_url = 'authorizeview'
    success_url = 'tripview'

```

```

    def form_valid(self, form):
        form.instance.users = [self.request.user]
        return super(TripNewView, self).form_valid(form)

```

```
class TripEditView (LoginRequiredMixin, UpdateView):
```

```

    template_name = 'tripedit.html'
    model = Trip

```

```

    fields = [
        'name',
        'startDate',
        'endDate',
        'membernum',
        'description',

```

```

        'mode',
        'users'
    ]

    login_url = 'authorizeview'
    success_url = 'tripview'

class OwnTripsView (LoginRequiredMixin, ListView):

    model = Trip
    template_name = 'owntrips.html'

    def get_queryset(self, *args, **kwargs):
        ptrips = super(OwnTripsView, self).get_queryset(*args,
**kwargs)
        us = self.request.user
        triplist = []
        tripname = self.request.POST.get('name')
        for t in ptrips:
            if us in t.users and (tripname.lower() in
t.name.lower()):
                triplist.append(t)
        return triplist

```

Б.3 Текст файла /views/categories.py

```

from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
from django.views.generic.list import ListView
import tripplanner.models

```

```

class CategoryGroupNewView(LoginRequiredMixin, CreateView):

    template_name = 'gcattegnew.html'
    model = CategoryGroup

    fields = [
        'name'
    ]

```

```
]
```

```
login_url = 'authorizeview'  
success_url = 'categorygroupview'
```

```
class CategoryGroupEditView (LoginRequiredMixin, UpdateView):
```

```
    template_name = 'gcatgedit.html'  
    model = CategoryGroup
```

```
    fields = [  
        'name'  
    ]
```

```
    login_url = 'authorizeview'  
    success_url = 'categorygroupview'
```

```
class CategoryNewView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'gcatgnew.html'  
    model = CategoryGroup
```

```
    fields = [  
        'name',  
        'categorygroup'  
    ]
```

```
    login_url = 'authorizeview'  
    success_url = 'categoryview'
```

```
class CategoryEditView(LoginRequiredMixin, CreateView):
```

```
    template_name = 'gcatgedit.html'  
    model = CategoryGroup
```

```
    fields = [  
        'name',  
        'categorygroup'
```

```
]

```

```
login_url = 'authorizeview'
success_url = 'categoryview'
```

```
class CategoryGroupsView (LoginRequiredMixin, ListView):
```

```
    model = CategoryGroup
    template_name = 'catgroups.html'
```

```
class CategoriesView (LoginRequiredMixin, ListView):
```

```
    model = Category
    template_name = 'categories.html'
```

```
    def get_queryset(self, *args, **kwargs):
        categs = super(CategoriesView, self).get_queryset(*args,
**kwargs)
        if self.request.POST:
            try:
                gr = self.request.POST.get('group')
                categs = categs.filter(categorygroup = gr)
            return categs
```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Слайд № 1

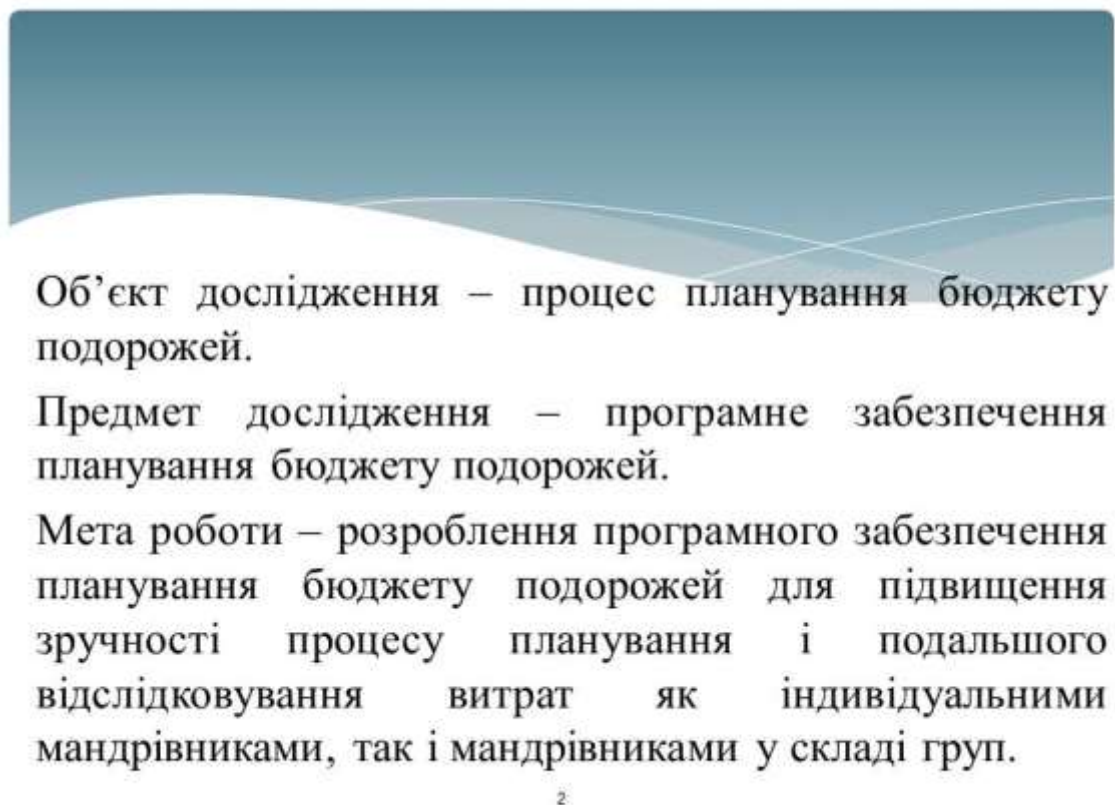


Рисунок В.2 – Слайд № 2

Завдання

- аналіз існуючих програмних аналогів;
- визначення вимог до програмного забезпечення планування бюджету подорожей;
- проєктування програмного забезпечення;
- реалізація програмного застосунку.

3

Рисунок В.3 – Слайд № 3

Порівняння мов програмування C# і Python		
Критерій	C#	Python
Область використання	Веброзробка	Веброзробка
Безкоштовність	+	+
Широка спільнота розробників	+	+
Високорівнева розробка	+	+
Динамічна типізація	-	+
Простота розгортання на вебсервері	+ -	+
Динамічна робота над проектом (включаючи інтерпретатор)	-	+

4

Рисунок В.4 – Слайд № 4



Рисунок В.5 – Слайд № 5



Рисунок В.6 – Слайд № 6

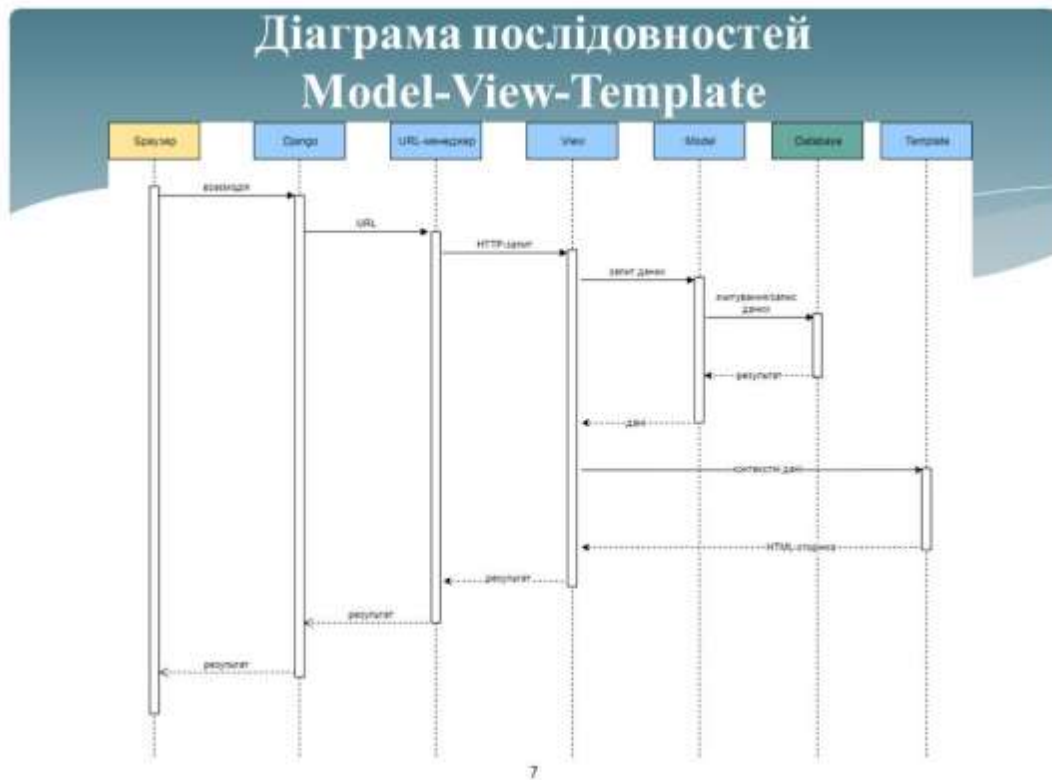


Рисунок В.7 – Слайд № 7



Рисунок В.8 – Слайд № 8

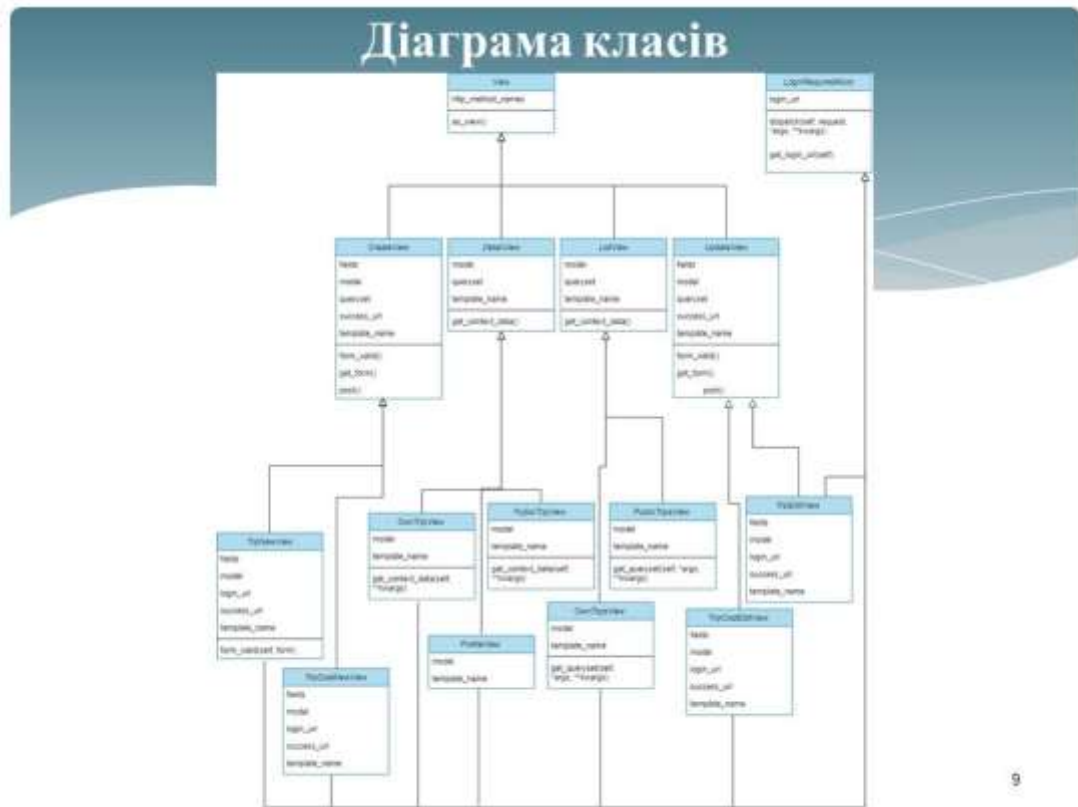


Рисунок В.9 – Слайд № 9

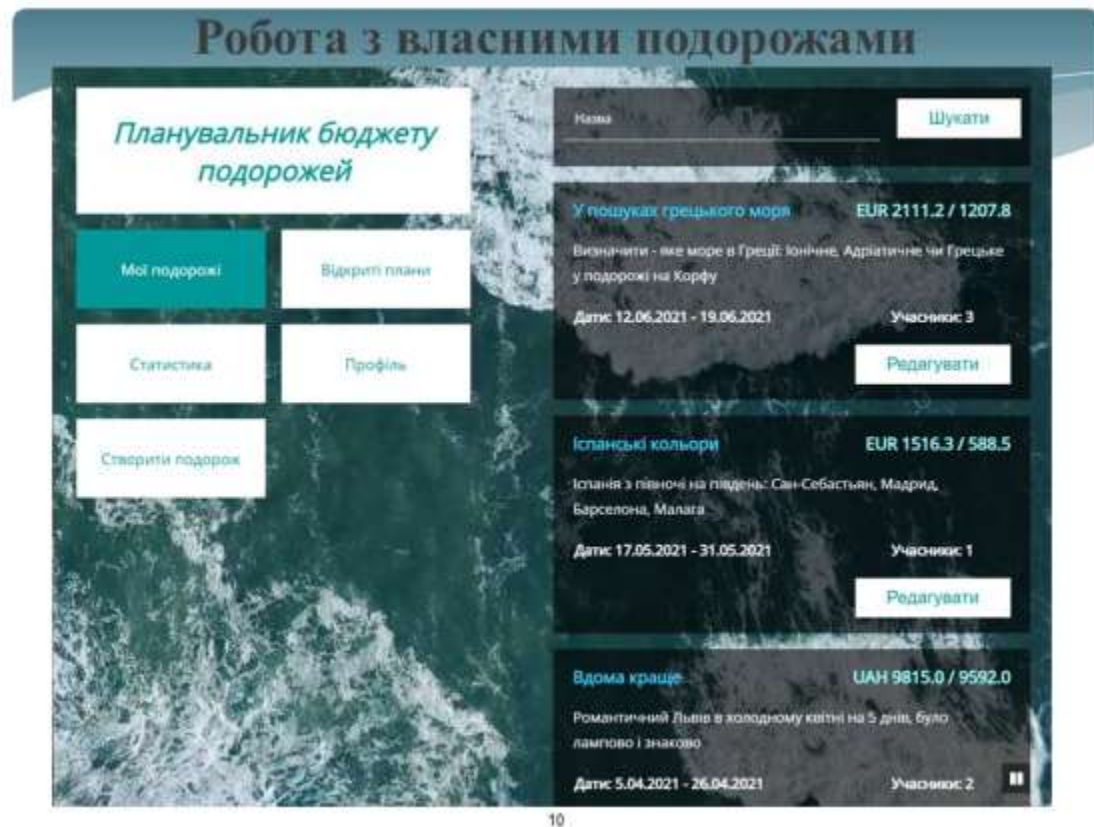
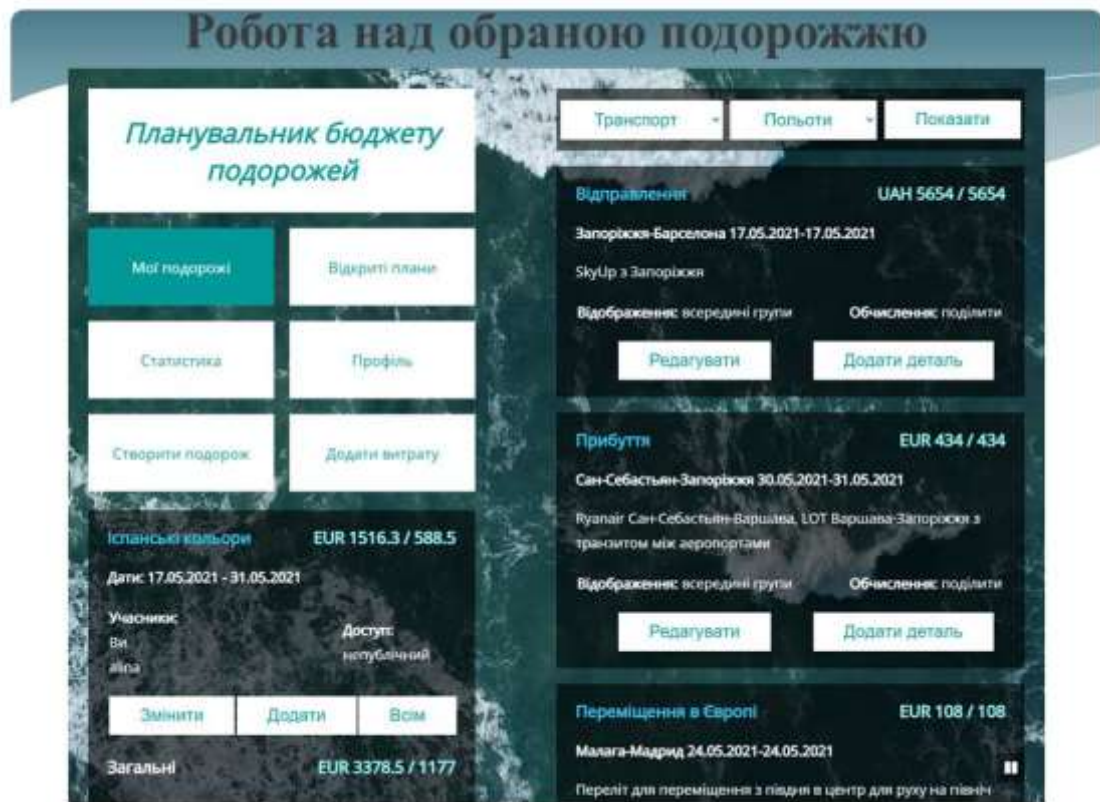
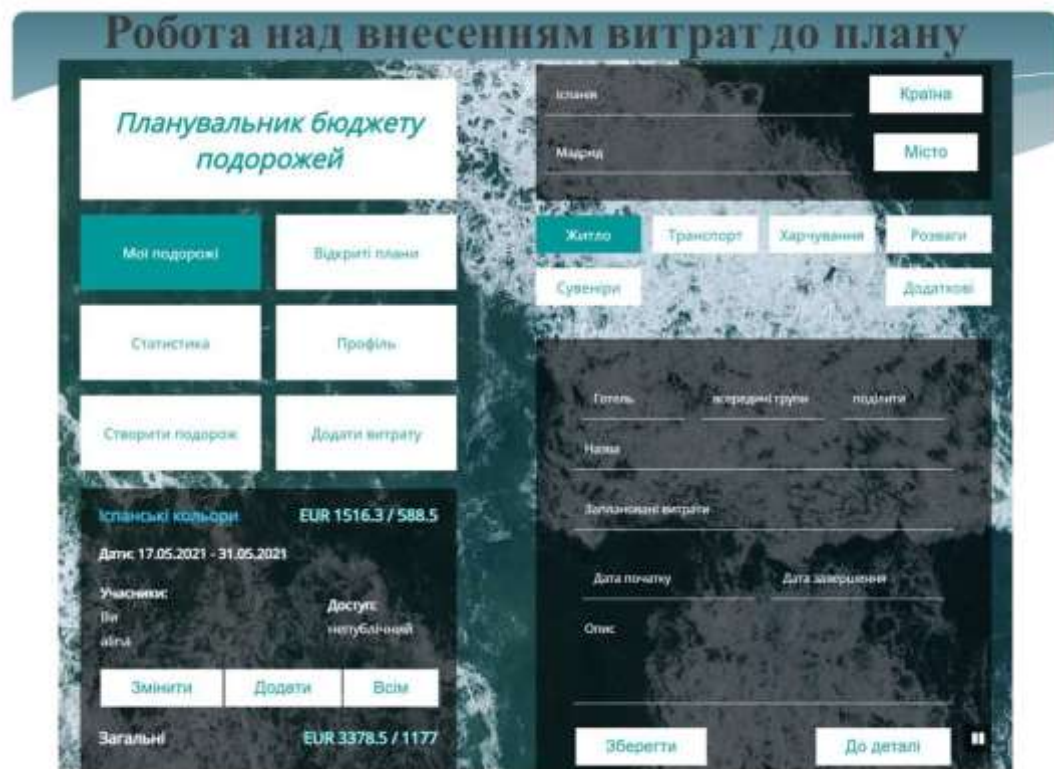


Рисунок В.10 – Слайд № 10



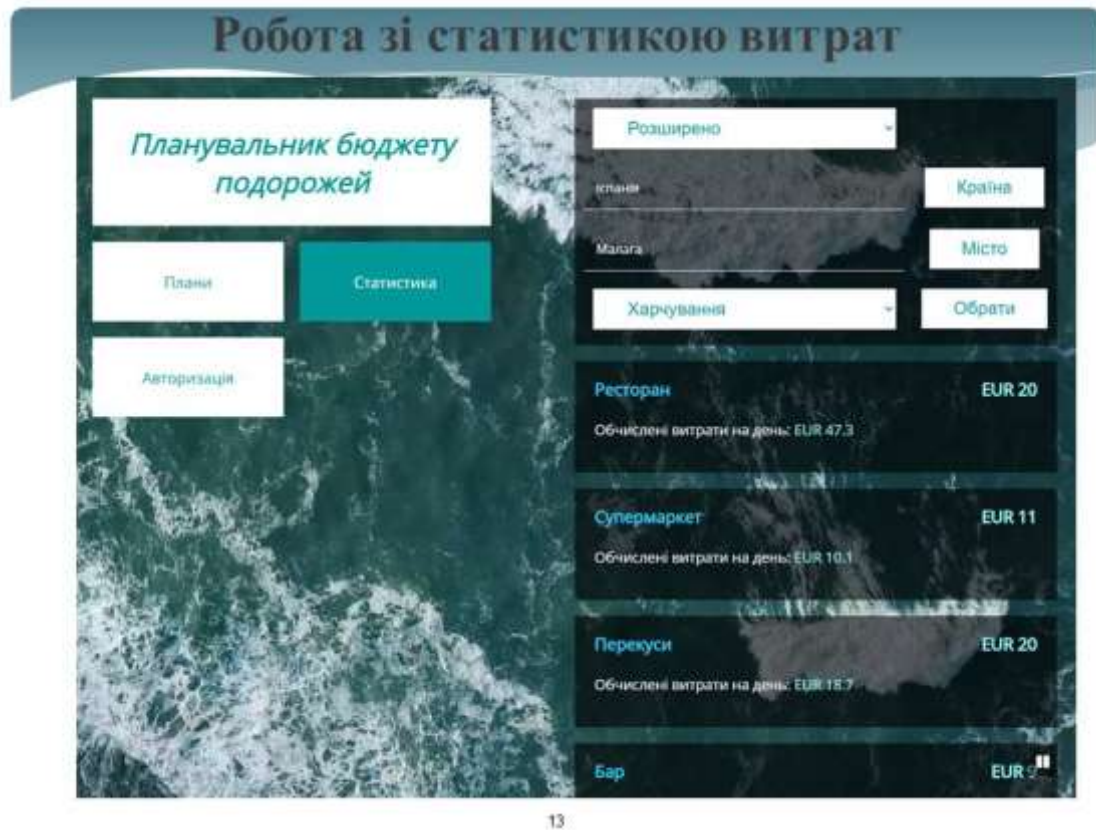
11

Рисунок В.11 – Слайд № 11



12

Рисунок В.12 – Слайд № 12



13

Рисунок В.13 – Слайд № 13



14

Рисунок В.14 – Слайд № 14