

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

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

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ
ПРОЦЕСУ АВТОМАТИЗАЦІЇ ДІЯЛЬНОСТІ
ТУРИСТИЧНОЇ АГЕНЦІЇ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
AUTOMATING THE ACTIVITY OF A TRAVEL
AGENCY

Виконав(ла): студент(ка) 2 курсу, групи КНТ-122м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

МЕЛЬНИКОВ Д.А.

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
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.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

МЕЛЬНІКОВА Данила Андрійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація процесу
автоматизації діяльності туристичної агенції. Research and Software Implementation of
the Process of Automating the Activity of a Travel Agency

керівник проєкту (роботи) д.т.н., доцент, СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 12 грудня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області. 2. Проєктування архітектури програмного забезпечення.

3. Основні рішення щодо реалізації компонентів програмного забезпечення.

4. Реалізація компонентів програмного забезпечення. 5. Тестування та

експериментальне дослідження програмного забезпечення. 6. Керівництво програміста.

7. Керівництво користувача.

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-7 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст. викладач		

7. Дата видачі завдання “ 15 ” вересня 2023 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 та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Опис керівництва програміста та користувача	9 тиждень	Розділ 6, 7
8	Оформлення пояснювальної записки та документів до неї.	10-11 тижні	Додатки
9	Нормоконтроль та рецензування.	12 тиждень	
10	Захист роботи.	12 тиждень	

Студент(ка)

_____ Данило МЕЛЬНІКОВ
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
237 с., 49 табл., 114 рис., 5 дод., 39 джерела.

УПРАВЛІННЯ, ДІАГРАМА, MVC, ФРЕЙМВОРК, PHP, Yii2, IDEF, BPMN, UML, БД, IC, ER, CRUD, CRM.

Об'єкт дослідження – інформаційні системи в сфері туризму.

Предмет дослідження – інноваційні підходи до розвитку інформаційних систем у сфері туризму.

Мета роботи – розробка та впровадження інноваційного підходу до розробки інформаційної системи для вдосконалення бізнес-процесів у туристичних агентствах.

Метод дослідження – комплексний аналіз, методи моделювання, дослідження вимог та впровадження інновацій у сфері розробки програмного забезпечення.

Матеріали, методи та технічні засоби: IDEF0, IDEF3, BPMN 2.0, ООП. Мови програмування PHP та MySQL для зберігання та обробки даних. Фреймворк Yii2, а також методологію розробки Transfer та паттерн Model-View-Controller (MVC).

Результати. Досягнуто значний прогресу в удосконаленні та розширенні можливостей клієнт/серверних інформаційних систем туристичних агентств.

Висновки. Розроблено і впроваджена програмна система для туристичного агентства, підвищило ефективність роботи та забезпечило швидке документування, бронювання послуг та ефективний зворотний зв'язок із клієнтами та постачальниками.

Галузь використання – застосовування в сфері туризму та гостювого обслуговування туристичними агентствами, туроператорами, та іншими учасниками галузі.

ABSTRACT

Explanatory note to the master's thesis: 237 pages, 49 tables, 114 figures, 5 appendices, 39 sources.

MANAGEMENT, DIAGRAM, MVC, FRAMEWORK, PHP, Yii2, IDEF, BPMN, UML, DB, IS, ER, CRUD, CRM.

The object of research is information systems in the field of tourism.

The subject of research is Innovative approaches to the development of information systems in the field of tourism.

The purpose of the work is to develop and implement an innovative approach to the development of an information system for improving business processes in travel agencies.

The research method is comprehensive analysis, modeling methods, requirements research and the introduction of innovations in the field of software development.

Materials, methods and technical means: IDEF0, IDEF3, BPMN 2.0, OOP. PHP and MySQL programming languages for data storage and processing. The Yii2 framework, as well as the Transfer development methodology and the Model-View-Controller (MVC) pattern.

The results. Significant progress has been made in improving and expanding the capabilities of client/server information systems of travel agencies.

Conclusions. Developed and implemented a software system for a travel agency, increased work efficiency and provided quick documentation, booking of services and effective feedback to customers and suppliers.

Field of use – application in the field of tourism and guest services by travel agencies, tour operators, and other industry participants.

ЗМІСТ

С.

Перелік скорочень та умовних позначок	11
Вступ	12
1 Аналіз предметної області.....	13
1.1 Основні поняття	13
1.2 Огляд існуючих методів для оцінки ефективності впроваджених програмних продуктів	17
1.2.1 Метод параметричної лінійної комбінації даних	17
1.2.2 Метод параметричного порівняння експериментальних даних А/В тестування	19
1.3 Огляд існуючих програмних продуктів для ведення інформаційно туристичної діяльності	22
1.3.1 Tourplan.....	22
1.3.2 Lemax.....	22
1.3.3 Dolphin Dynamics.....	23
1.4 Порівняльна характеристика програмних продуктів для ведення інформаційно туристичної діяльності	24
1.5 Висновок до розділу 1	26
2 Проєктування архітектури програмного забезпечення	27
2.1 Вибір мови та платформ програмування	27
2.1.1 Python	27
2.1.2 PHP	28
2.1.3 JavaScript	29
2.1.4 Java	29
2.2 Порівняння обраних мов програмування.....	30
2.3 Вибір середовища розробки програмного продукту.....	32
2.3.1 PHPStorm.....	32
2.3.2 Visual Studio Code.....	33

2.3.3 Eclipse.....	34
2.3.4 Sublime Text.....	34
2.4 Порівняння характеристика обраних середовищ розробки	35
2.5 Перелік обраних бібліотек для розробки програмного продукту	37
2.5.1 Yii2.....	37
2.5.2 Dektrium Yii2 User та RBAC.....	37
2.5.3 JavaScript та Bootstrap.....	38
2.5.4 Реляційна база даних	39
2.5.5 Laravel.....	39
2.5.6 React	40
2.5.7 Symfony	40
2.6 Вибір архітектури розробки програмного продукту.....	41
2.7 Опис застосованих інструментів розробки	44
2.8 Процес створення модулів	44
2.9 Створення контролера.....	45
2.10 Опис бібліотек та плагінів.....	46
2.11 Ієрархія даних проєкту.....	47
2.12 Структурна схема програмного продукту.....	47
2.13 Структурна схема програмного продукту.....	50
2.14 Діаграма класів.....	51
2.15 Діаграма компонентів	52
2.16 Діаграма розгортання.....	52
2.2 Висновок до розділу 2	53
3 Основні рішення щодо реалізації компонентів програмного забезпечення ..	55
3.1 Попередній огляд критеріїв оцінки.....	55
3.2 Застосування обраних методів	58
3.3 Висновок до розділу 3	68
4 Реалізація компонентів програмного забезпечення.....	70
4.1 Структура та зміст контролерів	70

4.2 Реалізовані програмні класи	72
4.3 Реалізація графічного інтерфейсу користувача	72
4.4 Висновки за розділом 4.....	75
5 Тестування та експериментальне дослідження програмного забезпечення ..	77
5.1 Тестування.....	77
5.2 Чек-ліст функціоналу WEB-додатку	82
5.3 Висновок до розділу 5.....	85
6 Керівництво програміста.....	86
6.1 Призначення й умови використання	86
6.2 Архітектура	86
6.3 Структура коду.....	87
6.4 Використані технології.....	88
6.5 Інструкція зі збирання та встановлення	89
6.6 Документація з API	90
6.7 Ієрархія папок та шляхи до основних файлів	90
7 Керівництво користувача	92
7.1 Загальна інформація	92
7.2 Область застосування.....	93
7.3 Рівень підготовки користувача	94
7.4 Призначення та умови застосування	94
7.5 Вимоги до програмного забезпечення.....	94
7.6 Підготовка до роботи	95
7.7 Перевірка працездатності системи.....	96
7.8 Склад та структура сайту	96
7.9 Реєстрація на сайті	99
7.10 Авторизація та розділи сайту	101
7.11 Функціонал користувача	103
7.12 Функціонал адміністратора.....	105
7.13 Створення акаунта користувача.....	105

7.14 Створення ролей.....	107
7.15 Створення правил.....	108
7.16 Функціонал менеджера.....	110
7.17 Створення турів.....	110
7.18 Попередній документ тура клієнта.....	112
7.19 Функціонал менеджера-постачальника.....	114
7.20 Заповнення даних країн.....	115
7.21 Заповнення даних міст.....	116
7.22 Заповнення даних готелів.....	117
7.23 Заповнення даних вимог.....	118
7.24 Заповнення даних вимог.....	119
7.25 Заповнення даних аеропортів.....	120
7.26 Заповнення даних типів турів.....	120
Висновки.....	122
Перелік джерел посилання.....	123
Додаток А Технічне завдання.....	127
А.1 Загальні вимоги до розробки програмного продукту для ведення інформаційно туристичної діяльності.....	128
А.1.1 Загальні відомості.....	128
А.1.2 Підстави на розробку.....	128
А.1.3 Призначення і цілі створення системи.....	128
А.1.4 Вимоги до системи.....	129
А.1.5 Вимоги до способів і засобів зв'язку компонентів системи.....	131
А.1.6 Вимоги до надійності.....	132
А.1.7 Вимоги до безпеки.....	132
А.1.8 Вимоги до завдань, що виконуються системою.....	133
А.1.9 Вимога до видів забезпечення.....	134
А.1.10 Характеристика об'єктів автоматизації.....	137
А.1.11 Вимоги до документування.....	138

А.1.12 Етапи розробки програмного продукту	138
А.1.13 Огляд етапів розробки	139
А.1.14 Зміст робіт за етапами	139
А.2 Порядок контролю і приймання системи	141
А.2.1 Методи випробування системи і її складових частин	141
А.2.2 Загальні вимоги до приймання робіт по стадіям	141
А.2.3 Статус приймальної комісії	141
А.3 Постановка завдання до роботи	141
Додаток Б Опис фізичної бази.....	143
Додаток В Текст програми.....	179
Додаток Г Контекстні діаграми, діаграми декомпозиції, дерева вузлів, діаграми послідовності, діаграма BPMN 2.0, діаграми потоків, IDEF3	215
Додаток Д Слайди презентації	227

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
BPMN	– Business Process Model and Notation;
CRUD	– Create, Read, Update, Delete;
CRM	– Customer Relationship Management;
DB	– Database;
ER	– Entity-Relationship;
MVC	– Model-View-Controller;
UML	– Unified Modeling Language;
KPI	– Key Performance Indicator;
UTP	– Unshielded Twisted Pair;
RBAC	– Role-Based Access Control;
HTTP	– Hypertext Transfer Protocol;
ОС	– операційна система;
ООП	– об'єктно-орієнтоване програмування;
ЛОМ	– Локальна обчислювальна мережа;

ВСТУП

Сучасний ринок туристичних послуг вимагає від гравців галузі операційно ефективних рішень для забезпечення високої якості обслуговування клієнтів. Враховуючи ці виклики, розробка та впровадження інформаційних систем для автоматизації ключових бізнес-процесів туристичних агентств стає все більш актуальним. Завдяки різноманітності існуючих рішень для автоматизації туристичного бізнесу, попит на комплексні, гнучкі та ефективні інструменти для надання високого рівня обслуговування та задоволення потреб сучасних туристів зростає з кожним днем.

Метою даного проєкту є створення окремого програмного продукту, що автоматизує ключові процеси туристичних агентств. Проєкт передбачає комплексний підхід до розвитку, який враховує аналіз сучасних тенденцій розвитку галузі, аналіз бізнес-процесів та впровадження передових технологій у туристичній сфері.

Ця робота спрямована на вирішення важливих завдань, таких як:

- забезпечення зручної взаємодії між клієнтами та агентствами;
- оптимізування внутрішніх процесів та підвищення загальної ефективності туристичного бізнесу;
- підвищення конкурентоспроможності та задоволення потреб сучасних туристів в умовах постійних змін у туристичній сфері.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Туризм – це галузь, пов'язана з подорожами та проживанням часу поза звичайним середовищем проживання з метою відпочинку, розваги, знайомства з новими місцями та культурами.

Турист – це особа, яка здійснює поїздку в інший регіон або країну з туристичною метою.

Туристична промисловість – це економічна сфера, що включає всі види діяльності, пов'язані з обслуговуванням туристів.

Гід – це фахівець, який проводить екскурсії та надає туристам інформацію про відвідувані місця.

Інформаційний туризм – це сфера, де використовуються інформаційні технології для організації та просування туристичних послуг.

Туристичний веб-сайт – це інтернет-ресурс з інформацією про туристичні напрямки та можливості онлайн-бронювання.

Мобільні програми для туризму – це програми для смартфонів, що надають інформацію про місця, карти, рейтинги готелів та інші сервіси.

Електронні бронювання – це онлайн системи для бронювання готелів, авіаквитків та інших послуг.

Віртуальні тури – це засоби, що дозволяють подорожувати віртуально через комп'ютер чи віртуальну реальність.

Складемо схему функціонування туристичного бізнесу (рис. 1.1) [1].

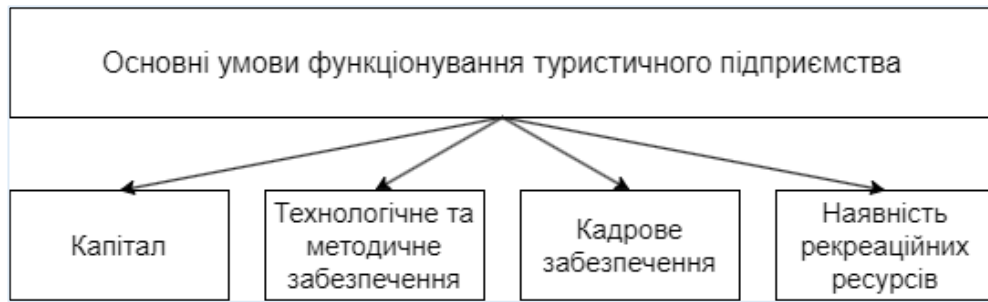


Рисунок 1.1 – Схема функціонування туристичного бізнесу [1]

Згідно рис. 1.1. схема функціонування туристичного бізнесу, що відображає взаємозалежні процеси та складові, що беруть участь у його роботі.

Складемо схему функціонування туристичної агенції, (рис. 1.2) [2].

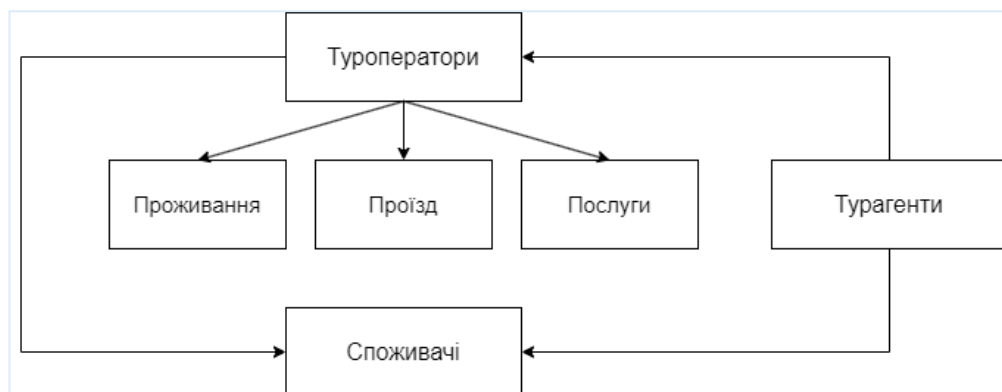


Рисунок 1.2 – Схема функціонування туристичної агенції [2]

Згідно рис. 1.2. схема функціонування туристичної агенції, що вказує на ключові елементи та процеси, що входять у її структуру та визначають її робочі функції.

Складемо схему функціонування туристичного сайту, (рис. 1.3) [3].

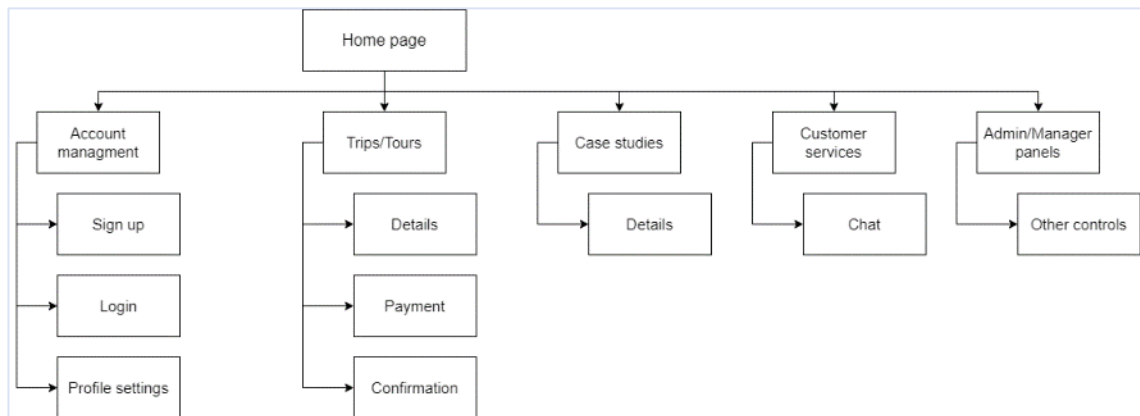


Рисунок 1.3 – Схема функціонування туристичного сайту [3]

Згідно рис. 1.3. схема функціонування туристичного сайту, що відображає основні елементи та взаємодії, що відбуваються на веб-ресурсі туристичної агенції.

Розглянемо основні критерії, які мають бути присутніми в інформаційному програмному продукті управління та функціонування туристичної агенції:

а) бронювання та білети (Reservation and Ticketing):

- 1) бронювання готелів: використання програмних систем для резервування розміщення для клієнтів;
- 2) бронювання транспорту: організація бронювання квитків на транспортні засоби (авіаквитки, поїзди, автобуси);
- 3) улектронні квитки: процес створення та обробки електронних квитків для зручності мандрівників;

б) управління клієнтської базою даних (Customer Relationship Management – CRM):

- 1) зберігання інформації про клієнтів: використання систем CRM для відстеження та керування даними про клієнтів, їх переваги та історію подорожей;
- 2) персоналізація послуг: використання даних про клієнтів для надання персоналізованих туристичних пропозицій;

в) фінансове управління:

- 1) бухгалтерія – програмні рішення для обліку фінансових операцій, включаючи виставлення рахунків, оплату та ведення бухгалтерії;
- 2) онлайн-платежі – інтеграція з платіжними системами для забезпечення безпечних онлайн-транзакцій;

г) інтернет-маркетинг та реклама:

- 1) сайт та мобільні програми: розробка та підтримка веб-сайту та мобільних додатків для залучення клієнтів та надання інформації про туристичні послуги;
- 2) цифровий маркетинг: використання соціальних мереж, контекстної реклами та інших цифрових каналів для просування туристичних пропозицій;

д) управління інвентарем (Inventory Management):

- 1) відстеження наявності послуг: управління та контроль за наявністю місць у готелях, квитків на транспорт та інших туристичних послуг;
- 2) інтеграція з постачальниками: обмін інформацією з постачальниками послуг для актуалізації інвентарю;

е) аналітика та звітність:

- 1) звіти про продаж: генерація звітів про продаж, бронювання та інші ключові показники для прийняття стратегічних рішень;
- 2) аналіз даних: використання інструментів аналізу даних для виявлення тенденцій та розуміння споживчої поведінки;

є) захист інформації (Information Security):

- 1) кібербезпека: реалізація заходів для захисту конфіденційних даних клієнтів, бронювань та фінансових операцій.

Відповідно до проаналізованої предметної області, постає питання у розробці індивідуального програмного забезпечення для ефективного функціонування туристичного агентства.

Однак, для оцінки розробленого програмного забезпечення необхідно використати розроблену методику оцінки ефективності розробленого програмного забезпечення, яка базується на використанні методу лінійної комбінації параметрів (Середнє арифметичне зважене, The min-max scaling method) з вагами, які відображають їхню відносну важливість у кінцевій оцінці [4]-[5].

Додатково до цього необхідно використовувати метод А/В тестування для порівняння двох або більше версій якогось елемента з метою визначення, яка версія ефективніша. У контексті впровадження програмного продукту в туристичній агенції, ми можемо провести А/В тестування, щоб оцінити ефективність нового клієнт-серверного програмного продукту.

Далі розглянемо огляд існуючих методів для оцінки ефективності впроваджених програмних продуктів.

1.2 Огляд існуючих методів для оцінки ефективності впроваджених програмних продуктів

1.2.1 Метод параметричної лінійної комбінації даних

Для оцінки розробленого програмного забезпечення розглянемо використання методики оцінки ефективності розробленого програмного забезпечення, яка базується на використанні методу лінійної комбінації параметрів (Середнє арифметичне зважене, The min-max scaling method) з вагами, які відображають їхню відносну важливість у кінцевій оцінці [4]-[5].

Даний метод дозволяє формувати зважені оцінки для визначення найбільш вагомих у аналізі даних, багатокритерність даних, підстроювання даних під конкретні умови, використання тестових даних для будування прогностичних оцінок, гнучкість моделі для адаптування різноманітних ситуацій та критеріїв для виконання оцінки.

Для нормалізації обраних параметрів будемо приводити дані до виду min-max нормалізації для перерахунку отриманих значень параметрів в діапазоні від 0 до 1. Для кожного параметра x значення нормалізованого параметра x розрахуємо наступним чином (1.1):

$$a = \frac{a - \min(a)}{\max(a) - \min(a)} \quad (1.1)$$

де a – вихідне значення параметра;

$\min(a)$ – мінімальне значення параметра;

$\max(a)$ – максимальне значення параметра.

Додаткового до кожного параметра встановлюються вагові коефіцієнти w_i , які в сумі не повинні перевищувати 1.

Після нормалізації параметрів можна застосувати вагові коефіцієнти для врахування їх відносної важливості. Вагові коефіцієнти можуть бути задані на основі експертних оцінок або інших методів оцінки важливості кожного параметра.

Загальна оцінка (1.2) обчислюється як зважена сума нормалізованих параметрів представлених вище:

$$S = w_1 * a_1 + w_2 * a_2 + w_3 * a_3 \dots + w_n * a_n \quad (1.2)$$

де S – загальна оцінка;

$w_1, w_2 \dots w_n$ – вагові коефіцієнти;

$a_1, a_2, \dots a_n$ – нормалізовані значення параметрів.

Отже, цей підхід дозволяє врахувати відносну важливість параметрів та надає більш об'єктивну оцінку ефективного впровадження програмного засобу.

Як діапазон оцінки ефективного впровадження програмного продукту (S) констатуємо такі значення:

- діапазон $S = 0.5 - 0.21$, вказує на критичний рівень ефективності розробленого програмного продукту, вимоги та параметри до якого потребують повного перепроєктування;
- діапазон $S = 0.22 - 0.32$, вказує на низький рівень ефективності розробленого програмного продукту, який виконує мінімальний набір функцій, та потребує повного перепроєктування;
- діапазон $S = 0.33 - 0.54$, вказує на середній рівень ефективності розробленого програмного продукту, який виконує більшість основних функцій, та потребує невеликий об'єм доробки програмного засобу;
- діапазон $S = 0.55 - 0.78$, вказує на достатній рівень ефективності розробленого програмного продукту, який виконує основні функції та має додатковий функціонал різнитися з іншими програмними продуктами;
- діапазон $S = 0.79 - 1$, свідчить про ідеальний рівень ефективності розробленого програмного продукту, який відрізняється від обраного вузького сегменту обраної предметної області, та займає лідируючі позиції серед клієнтів.

1.2.2 Метод параметричного порівняння експериментальних даних А/В тестування

А/В тестування – це метод порівняння двох або більше версій якогось елемента з метою визначення, яка з версій ефективніша. У контексті впровадження програмного продукту в туристичній агенції, ми можемо

провести А/В тестування, щоб оцінити ефективність нового клієнт-серверного програмного продукту.

Визначимо параметри та створимо математичну модель для оцінки ефективності на основі А/В тестування:

В якості загального показника для аналізу даних виступає KPI_{old} та KPI_{new} , які визначають найбільш важливі критерії ефективності впровадженого програмного забезпечення.

В якості загального показника для об'єму вибірки даних виступає N_{old} та N_{new} , які визначають мінімальний та максимальний діапазон аналізованої вибірки даних.

В якості загального показника для визначення кількості даних виступає X_{old} та X_{new} , які визначають кількість успішності відпрацьованих подій.

Також для достовірних розрахунків виділимо параметри для визначення α – рівень значимості (вірогідність помилок першого роду) та β – потужність тесту ($1 - \beta$ – вірогідність помилок другого роду).

Визначивши параметри та критерії оцінки, сформуємо математичну модель оцінки різниці між старим та новим продуктом та сформуємо інтерпретацію формулу наступним чином, див. формулу (1.3):

$$\begin{aligned} H_0 : KPI_{old} &= KPI_{new} \\ H_1 : KPI_{old} &\neq KPI_{new} \end{aligned} \quad (1.3)$$

де H_0 – нульова гіпотеза (відсутності різниці);

H_1 – альтернативна гіпотеза (з наявною різницею).

Для розрахунку різниці аналізованих параметрів покращення або погіршення ефективності розроблено програмного продукту, див. формулу (1.4):

$$\begin{aligned} KPI_{old} &= \frac{X_{old}}{N_{old}} \\ KPI_{new} &= \frac{X_{new}}{N_{new}} \end{aligned} \quad (1.4)$$

де N_{old} та N_{new} – об'єм вибірки даних;

X_{old} та X_{new} – кількість успішності відпрацьованих подій.

Якщо, результат тесту дозволяє спростувати нульову гіпотезу, це може вказувати на статистичну значимість покращення або погіршення ефективності процесу автоматизації діяльності розробленого програмного продукту для туристичної агенції.

Метод параметричної комбінації є більш гнучким, адаптується до різних умов та критеріїв оцінки, у той час як А/В тестування зосереджується переважно на порівнянні конкретних версій продуктів. З точки зору об'єктивності оцінки, метод параметричної комбінації пропонує більш об'єктивний підхід, враховуючи відносну важливість різних параметрів, в той час як результати А/В тестування можуть залежати від конкретних характеристик вибірок та умов тестування. Щодо практичного застосування в туристичних агентствах, А/В тестування може бути більш підходящим, оскільки воно дозволяє безпосередньо порівнювати різні версії програмного продукту в реальних умовах використання. Однак, обидва методи мають свої переваги та недоліки, і вибір між ними слід робити, базуючись на конкретних вимогах завдання оцінки та умовах, в яких вони будуть використовуватися. Для комплексної оцінки програмного продукту, особливо у сфері туризму,

рекомендується використовувати обидва методи для отримання повної картини його ефективності.

1.3 Огляд існуючих програмних продуктів для ведення інформаційно туристичної діяльності

1.3.1 Tourplan

Tourplan – це програмний продукт, спрямований на автоматизацію та управління бізнес-процесами туристичних агентств. Він надає засоби для створення, зберігання та обробки інформації про турпакети, бронювання послуг, облік клієнтів та фінансовий облік. Tourplan відомий своєю ефективністю та надійністю, що робить його популярним серед туроператорів та туристичних агентств [6].

Переваги Tourplan включають простий та зручний інтерфейс, високий рівень кастомізації під потреби конкретного бізнесу, а також широкий функціонал для управління турпакетами та роботи з клієнтами. Система підтримує автоматичний облік фінансів, що спрощує фінансовий облік туристичних компаній.

Недоліками Tourplan є високі вартості ліцензій та обмеження в розгортанні на різних платформах. Крім того, для ефективного використання системи може знадобитися навчання персоналу.

1.3.2 Lemax

Lemax – це інтегрована система для управління туристичним бізнесом, яка включає в себе модулі для обліку турпакетів, бронювання, фінансового обліку та звітності. Lemax визначається своєю гнучкістю та можливістю адаптації під потреби різних туристичних компаній [7].

Серед переваг Lемах слід відзначити його хороший функціонал для роботи з турпакетами, можливість інтеграції з іншими системами та сервісами, а також зручний інтерфейс для користувачів. Система підтримує онлайн-бронювання, що полегшує взаємодію з клієнтами.

Недоліками Lемах є високі витрати на впровадження та обслуговування, а також потребу в технічній підтримці. Деякі користувачі можуть відзначити складність в освоєнні системи, особливо для новачків у галузі.

1.3.3 Dolphin Dynamics

Dolphin Dynamics – це програмна система для автоматизації та управління туристичним бізнесом, яка охоплює весь цикл обробки замовлень, від пошуку турів до фінансового обліку. Dolphin Dynamics славиться своєю комплексністю та інтегрованістю модулів [8].

Серед переваг Dolphin Dynamics варто відзначити його широкий функціонал, здатність працювати в реальному часі, а також можливість інтеграції з іншими системами та сервісами. Система надає інструменти для ефективного управління клієнтами та обліку фінансів.

Недоліки Dolphin Dynamics включають високі витрати на впровадження та технічну підтримку. Деякі користувачі можуть відзначити складність інтерфейсу та необхідність навчання персоналу перед використанням системи.

1.4 Порівняльна характеристика програмних продуктів для ведення інформаційно туристичної діяльності

Сформуємо порівняльну характеристику продуктів для ведення інформаційно туристичної діяльності, яка наведена у табл. 1.1. Вони можуть бути використані для дослідження та розробки програмних рішень для турагентств [6]-[8].

Таблиця 1.1 – Порівняльна характеристика програмних продуктів для ведення інформаційно туристичної діяльності.

Назва критерію	Tourplan	Lemax	Dolphin Dynamics
Ліцензія	Так	Так	Так
Операційна система	Крос-платформенний	Крос-платформенний	Крос-платформенний
Доступність	Закритий код	Закритий код	Закритий код
Пробна версія	Так	Так	Так
Адміністрування	Так	Так	Так
Імпорт/Експорт даних	Стандартні опції імпорту/експорту	Зручний обмін даними між модулями	Інтеграція з різними форматами, можливість експорту та імпорту
API	Обмежена можливість	Є, але з обмеженнями	Інтеграції з іншими системами
Доступ багатьом користувачам	Так	Так	Так
Сумісність з сучасними технологіями	Так	Так	Так
Рівень користувацької підтримки	Високий	Середній	Високий

Продовження таблиці 1.1

Назва критерію	Tourplan	Lemax	Dolphin Dynamics
Масштабованість системи	Так	Так	Так
Доступність	Закритий код	Закритий код	Закритий код
Пробна версія	Так	Так	Так
Адміністрування	Так	Так	Так
Імпорт/Експорт даних	Стандартні опції імпорту/експорту	Зручний обмін даними між модулями	Інтеграція з різними форматами, можливість експорту та імпорту
API	Обмежена можливість	Є, але з обмеженнями	Інтеграції з іншими системами
Доступ багатьом користувачам	Так	Так	Так
Сумісність з сучасними технологіями	Так	Так	Так
Рівень користувацької підтримки	Високий	Середній	Високий
Масштабованість системи	Так	Так	Так
Системні вимоги	Середні	Середні	Середні
Можливість індивідуального налаштування	Так	Так	Так
Забезпечення конфіденційності даних	Так	Так	Так
Система звітності	Так	Так	Так

Порівнюючи існуючі інформаційні системи в галузі туризму та акцентуючи на таких функціях, як бронювання та продаж квитків, управління відносинами з клієнтами (CRM), фінансове управління, інтернет-маркетинг, управління запасами, аналітика та інформаційна безпека. Програмна реалізація процесу автоматизації діяльності туристичної агенції відрізнятиметься завдяки розширеним функціям, яких немає в існуючих системах, таким як більш складна аналітика даних для прогнозування тенденцій, обслуговування клієнтів, покращених заходів кібербезпеки та більш інтуїтивно зрозумілого інтерфейсу для туристів і співробітників агентств. Програма покликана заповнити прогалини в існуючих системах, пропонуючи більш комплексне, ефективне та безпечне рішення для галузі туризму.

1.5 Висновок до розділу 1

У цьому розділі було проведено детальний аналіз галузі, що дозволило визначити важливі поняття, пов'язані з галуззю та предметом дослідження.

Ретельно розглянуто основні аспекти індустрії туризму та фактори, що впливають на сучасні туристичні технології.

Розглянуто різні методи, які можна використовувати для оптимізації процесів у туристичній сфері.

У рамках дослідження також проведено порівняльний аналіз існуючих програмних продуктів для здійснення інформаційно-туристичної діяльності.

Оцінюються їх можливості, функціональні характеристики та характеристики, що дозволяють визначити сильні та слабкі сторони кожної системи.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови та платформ програмування

Для розробки системи автоматизації туристичного агентства важливо вибрати мову програмування, яка відповідає конкретним потребам і характеристикам проєкту.

У туристичній індустрії використовуються різні технології для оптимізації та автоматизації бізнес-процесів.

Найпоширенішими мовами програмування для розробки систем туристичних агентств є PHP, JavaScript, Python і Java.

2.1.1 Python

Python – це мова програмування загального призначення, яка набула широкого застосування у різних галузях, включаючи розробку веб-додатків, аналіз даних та штучний інтелект. Вона відзначається простотою синтаксису та великою кількістю бібліотек, що сприяє швидкому розвитку та впровадженню проєктів [9].

Характеристики та переваги:

- Python є мовою програмування загального призначення, що дозволяє використовувати її для різних завдань, включаючи розробку веб-додатків, обробку даних, інтеграцію з базами даних та реалізацію алгоритмів штучного інтелекту;

- Python має обширну екосистему бібліотек, таких як NumPy, Pandas, та Scikit-learn для ефективного використання в аналізі даних та машинному навчанні;

- синтаксис Python лаконічний та легко читається, що сприяє розробці та підтримці коду;

- Python користується великою спільнотою розробників, що означає активний розвиток та високий рівень підтримки.

Недоліки:

- у порівнянні з мовами, такими як C++ чи Java, Python може бути менш ефективним щодо швидкодії, особливо у великих обчислювальних завданнях;
- Python використовує глобальний інтерпретатор, що може створювати проблеми при роботі з багатозадачними програмами;
- Python може бути менш ефективним для розробки мобільних додатків.

2.1.2 PHP

PHP – це скриптова мова програмування, спеціально створена для розробки веб-додатків. Вона використовується для створення динамічних веб-сайтів та забезпечення взаємодії з базами даних [10].

Характеристики та переваги:

- PHP спеціально розроблено для веб-розробки, що робить його потужним інструментом для серверної частини веб-додатків;
- PHP вважається легким для вивчення, що дозволяє швидко створювати веб-застосунки;
- більшість хостинг-провайдерів підтримують PHP, що дозволяє легко розгортати веб-проекти;
- завдяки різноманітним фреймворкам, таким як Laravel та Symfony, PHP може бути використаний для розробки складних веб-додатків.

Недоліки:

- PHP може виявитися менш ефективним для великих та складних проєктів порівняно з іншими мовами;
- У порівнянні з іншими мовами, такими як Java чи C++, PHP може працювати повільніше.

2.1.3 JavaScript

JavaScript – це високорівнева мова програмування, широко використовуваною для створення динамічних веб-сайтів та реалізації взаємодії на стороні клієнта веб-додатків [11].

Характеристики та переваги:

- JavaScript використовується для взаємодії з користувачем на стороні клієнта, що робить його невід’ємною частиною розробки веб-додатків;
- JavaScript використовується для розробки не лише клієнтської частини веб-сайтів, але і серверних додатків, що забезпечує уніфікацію розробки;
- JavaScript може використовувати асинхронний підхід для ефективної обробки багатьох одночасних запитів;
- JavaScript має велику та активну спільноту розробників, а також використовує потужний пакетний менеджер для управління залежностями проєкту.

Недоліки:

- асинхронний підхід дозволяє ефективно обробляти багато запитів, JavaScript є однопоточним, що може обмежити продуктивність у деяких випадках;
- асинхронний стиль програмування може бути важким для розуміння для новачків.

2.1.4 Java

Java – це високопродуктивна, мультипарадигмальна мова програмування, яка використовується для розробки різноманітних застосунків, включаючи веб-додатки, мобільні додатки та корпоративні системи [12].

Характеристики та переваги:

- Java є платформонезалежною мовою, що дозволяє використовувати код на різних платформах без перекомпіляції;
- Java підтримує об’єктно-орієнтований підхід, що сприяє покращенню модульності та повторного використання коду;
- мова має вбудовану підтримку багатопоточності, що дозволяє ефективно використовувати ресурси системи та поліпшує продуктивність додатків;
- Java має обширну екосистему бібліотек і фреймворків, що полегшує розробку та розширення функціональності додатків;
- мова має вбудовані механізми безпеки, що робить її популярною для розробки безпечних корпоративних систем.

Недоліки:

- розробники вважають, що Java може генерувати зайвий обсяг коду порівняно з іншими мовами;
- додатки на Java можуть запускатися повільніше порівняно з деякими іншими мовами програмування.

2.2 Порівняння обраних мов програмування

Проаналізувавши мови програмування [9]–[12], було створено порівняльний аналіз мов, який наведений у табл. 2.1.

Таблиця 2.1 – Порівняльна характеристика обраних мов програмування.

Назва критерію	Python	Java	JavaScript	PHP
Ліцензія	Python Software Foundation License	GNU GPL	-	PHP License

Продовження таблиці 2.1

Назва критерію	Python	Java	JavaScript	PHP
Простота навчання та написання коду	+	+	+	+
Кросплатформність	+	+	+	+
Можливість обробляти операції з великим обсягом даних	+	+	+	+
Паралельне програмування	+	+	+	-
Наявність бібліотек для машинного навчання	+	±	+	-
Інтеграція з іншими мовами	+	+	+	+
Наявність зрозумілої та повної документації	+	+	+	+
Висока візуалізація даних	+	-	+	-
Підвищена продуктивність	+	+	+	+
Великий список предметно-орієнтованих пакетів з відкритим вихідним кодом	+	+	+	+
Підтримка спільноти	+	+	+	+
Сумісні платформи	Windows, macOS, Linux, Unix	Крос-платформенний	Крос-платформенний	Крос-платформенний
Призначення	Загальне програмування, веб-розробка, аналіз даних	Загальне програмування, корпоративні застосунки	Веб-розробка, створення інтерактивних інтерфейсів	Веб-розробка

Як видно з табл. 2.1, найбільші переваги в контексті розробки турагентських систем має мова програмування PHP та Javascript. PHP відзначається простотою навчання та написання коду, що робить його ефективним інструментом для веб-розробки. Мова підтримує кросплатформність, що важливо для різних операційних систем, і має велику спільноту розробників, що сприяє обміну досвідом та вирішенню проблем.

Javascript також визначається простотою та гнучкістю, особливо у контексті взаємодії з клієнтами на стороні веб-додатків. Здатність обробляти операції з великим обсягом даних та використання асинхронного програмування робить її ефективним інструментом для реалізації інтерактивних веб-інтерфейсів. Мова також користується підтримкою великої та активної спільноти, що сприяє розвитку та пошуку оптимальних рішень.

Отже, для розробки та оптимізації турагентських систем та автоматизації процесів у туристичній галузі обрано мови програмування PHP та Javascript.

2.3 Вибір середовища розробки програмного продукту

Середовища розробки відіграють ключову роль у спрощенні процесу написання, тестування та налагодження коду, забезпечуючи програмістам необхідні інструменти для ефективної роботи. Для розробки веб-додатків та виконання завдань у мовах програмування PHP та JavaScript, використовуються різноманітні середовища зручного програмування.

2.3.1 PHPStorm

PHPStorm – це інтегроване середовище розробки, призначене спеціально для мови PHP. Це має повнофункціональний інтерфейс з великою кількістю інструментів для автоматизації рутинних задач та полегшення

написання коду. Однією з його переваг є можливість взаємодії з широким спектром фреймворків, таких як Laravel, Symfony та інші, що дозволяє розробникам швидко та ефективно створювати веб-додатки на PHP [13].

Характеристики та переваги:

- повнофункціональний інтерфейс для комфортної розробки на PHP;
- інтеграція з різноманітними фреймворками для покращення продуктивності;
- інструменти для роботи з базами даних та вбудований відлагоджувач PHP.

Недоліки:

- платний продукт, але доступний пробний період для тестування.

2.3.2 Visual Studio Code

Visual Studio Code – це легкий, але потужний редактор коду, який підтримує роботу як з мовою PHP, так і з JavaScript. Для розробки на мові PHP він використовує розширення, яке дозволяє розробникам ефективно писати, тестувати та налагоджувати PHP-код. З іншого боку, для розробки веб-додатків на мові JavaScript, включаючи використання Node.js, Visual Studio Code надає зручне інтегроване середовище розробки з такими функціями, як IntelliSense, вбудоване відлагодження та інструменти для роботи з пакетами Node.js та npm [14].

Характеристики та переваги:

- безкоштовний та легкий у використанні;
- велика кількість розширень для роботи з PHP та JavaScript;
- підтримка роботи з системою контролю версій Git;
- велика кількість розширень для роботи з PHP;
- широкий спектр функцій для роботи з пакетами Node.js та npm;
- активна спільнота розробників.

Недоліки:

- в порівнянні з іншими інтегрованими середовищами розробки, може вимагати налаштувань для повноцінної роботи з PHP та Node.js.

2.3.3 Eclipse

Eclipse – це інтегроване середовище розробки, яке підтримує мову програмування PHP завдяки розширенню PDT (PHP Development Tools). Eclipse надає розробникам велику кількість інструментів для створення веб-додатків та підтримує взаємодію з різними фреймворками. Це безкоштовне середовище, яке володіє різноманітними можливостями для роботи з PHP-проектами [15].

Характеристики та переваги:

- безкоштовне та потужне середовище розробки;
- велика кількість розширень для роботи з PHP;
- підтримка роботи з різними фреймворками.

Недоліки:

- вимагає деяких налаштувань та знань для повноцінної роботи.

2.3.4 Sublime Text

Sublime Text – це легкий, швидкий та універсальний текстовий редактор, який можна розширювати за допомогою плагінів. Для роботи з PHP та JavaScript рекомендується встановити відповідні плагіни, які додають підтримку цих мов. Sublime Text відрізняється своєю швидкістю та легкістю використання, він підтримує відлагодження та автодоповнення коду [16].

Характеристики та переваги:

- швидкий та легкий текстовий редактор;
- велика кількість плагінів для розширення функціоналу;

– підтримка PHP та JavaScript за допомогою відповідних плагінів.

Недоліки:

- за замовчуванням не має таких широких можливостей як інтегровані середовища розробки;
- потрібно вручну встановлювати та налаштовувати плагіни для роботи з мовами програмування.

2.4 Порівняння характеристика обраних середовищ розробки

Проаналізувавши середовища програмування [13]–[16], було створено порівняльний аналіз середовищ розробки, який наведений у табл. 2.2.

Таблиця 2.2 – Порівняльна характеристика обраних середовищ розробки.

Назва критерію	PHPStorm	Visual Studio Code	Eclipse	Sublime Text
Ліцензія	Proprietary	MIT	Eclipse Public License	Proprietary
Кросплатформність	+	+	+	+
Автозавершення коду	+	+	+	+
Підтримка репозиторіїв	+	+	+	+
Швидкий та безпечний рефакторинг коду	+	-	+	-
Перевірка та аналіз коду	+	+	+	+
Простота встановлення	-	+	-	+
Наявність інтерактивної консолі	+	+	+	+
Потребує малої кількості пам'яті для зберігання	-	+	-	+

Продовження таблиці 2.2

Назва критерію	PHPStorm	Visual Studio Code	Eclipse	Sublime Text
Підсвічування синтаксису	+	+	+	+
Вартість	Платний	Безкоштовний	Безкоштовний	Безкоштовний
Можливості розширення	+	+	+	+
Розширення для роботи з конкретними мовами програмування	+ (основна підтримка PHP)	+ (розширення для PHP)	+ (за допомогою плагінів)	+

Як видно з табл. 2.2, в контексті вибору середовища розробки для роботи з мовами програмування PHP та Javascript, особливо варто звернути увагу на Visual Studio Code. Це легке та потужне інтегроване середовище розробки, яке підтримує роботу з обома мовами, забезпечуючи широкі можливості для ефективної та зручної розробки.

Visual Studio Code отримало визнання завдяки своїй безкоштовності, легкості використання та розширюваності за допомогою різноманітних плагінів, включаючи ті, що сприяють розробці на PHP та Javascript. Інтеграція з локальними репозиторіями Git, можливість розширеного відлагодження коду та активна спільнота розробників роблять його привабливим вибором для тих, хто працює з веб-розробкою та мовами, такими як PHP та Javascript.

Наявність функції статичного аналізу коду, підтримка репозиторіїв, а також зручний користувацький інтерфейс роблять Visual Studio Code інструментом, який відповідає вимогам для продуктивної та комфортної розробки в турагентській галузі. Таким чином, для оптимізації та розробки турагентських систем обрано середовище Visual Studio Code.

2.5 Перелік обраних бібліотек для розробки програмного продукту

Для реалізації програмного продукту було обрано низку бібліотек, що ефективно підтримують вибрані мови програмування та відповідають потребам розробки в контексті вибраного фреймворку та паттерну проектування.

2.5.1 Yii2

Yii2 є основним фреймворком для реалізації серверної логіки програмного продукту, працюючи за принципом MVC (model, view, controller) [17].

Характеристики та переваги:

- простота та ефективність у використанні, особливо з урахуванням принципів MVC;
- широкі можливості для відділення логіки додатку від представлення та управління;
- зручна робота з базою даних через активний запис та інші вбудовані засоби.

Недоліки:

- вимагає певного часу та зусиль для освоєння та повноцінного використання;
- деяка складність в конфігуруванні для специфічних потреб проєкту.

2.5.2 Dektrium Yii2 User та RBAC

Для реалізації системи управління користувачами та контролю доступу, використовуються плагін Dektrium Yii2 User та RBAC (role based

access control). Це дозволяє ефективно керувати правами доступу та ідентифікацією користувачів [17]-[18].

Характеристики та переваги:

- дектриум Yii2 User спрощує інтеграцію системи управління користувачами;
- RBAC дозволяє гнучко налаштовувати та керувати ролями та доступом користувачів;
- зручна робота з базою даних через активний запис та інші вбудовані засоби.

Недоліки:

- можливість виникнення конфліктів при інтеграції додаткових плагінів;
- потреба в уважному налаштуванні та плануванні для забезпечення безпеки та ефективності системи.

2.5.3 JavaScript та Bootstrap

JavaScript використовується як допоміжний інструмент для реалізації взаємодії з клієнтами на стороні веб-додатків. Використання компонента Bootstrap спрощує розробку та надає інтерактивні можливості для користувачів.

Характеристики та переваги:

- JavaScript дозволяє ефективно взаємодіяти з елементами веб-сторінки, що поліпшує користувацький досвід;
- використання Bootstrap забезпечує однорідний та привабливий дизайн додатку [19];
- зручність в реалізації асинхронних запитів та відправки даних без перезавантаження сторінки.

Недоліки:

- залежно від обсягу взаємодії з клієнтом, складність у вивченні ряду асинхронних концепцій JavaScript;
- можливість виникнення проблем з кросбраузерною сумісністю.

2.5.4 Реляційна база даних

В програмних продуктах використовується реляційна база даних, яка дозволяє зберігати та обробляти дані залежностей через зв'язку в БД. Моделі бази даних відображаються в логічній моделі додатку [20].

Характеристики та переваги:

- забезпечення структурованого та ефективного зберігання даних;
- зручний та масштабований доступ до інформації через мову запитів SQL.

Недоліки:

- можливість виникнення проблем з продуктивністю при недостатній оптимізації запитів;
- потреба в дотриманні нормалізації бази даних для уникнення проблем з цілісністю та зайвими зв'язками.

2.5.5 Laravel

Laravel є одним з популярних фреймворків для PHP, спрямованим на розробку веб-додатків. Він відзначається елегантним синтаксисом, високою продуктивністю та широким спектром вбудованих інструментів [21].

Характеристики та переваги:

- зручна робота з базою даних через Eloquent ORM;
- вбудована система маршрутизації для легкого керування шляхами додатку;
- швидкий розвиток завдяки активній спільноті та широкому спектру

розширень.

Недоліки:

- можливість виникнення конфліктів при інтеграції деяких розширень;
- потреба у вивченні специфічних особливостей фреймворка для максимальної продуктивності.

2.5.6 React

React є бібліотекою для розробки інтерфейсів користувача веб-додатків. Вона дозволяє створювати компоненти, які автоматично оновлюються при зміні даних, що робить її ефективною для інтерактивних інтерфейсів [22].

Характеристики та переваги:

- висока продуктивність завдяки віртуальному DOM;
- можливість використання для створення односторінкових додатків (SPA);
- широка підтримка від спільноти та наявність багатьох розширень.

Недоліки:

- необхідність використання додаткових бібліотек або фреймворків для роботи з іншими частинами додатку;
- можливість виникнення проблем з оптимізацією для пошукових систем при неправильному використанні.

2.5.7 Symfony

Symfony є високопродуктивним фреймворком для розробки веб-додатків на PHP. Він базується на компонентах, що робить його дуже гнучким та розширюваним [23].

Характеристики та переваги:

- використання компонентної архітектури для максимальної гнучкості;
- широкий спектр готових компонентів для різних завдань;
- зручна робота з базою даних та вбудованим відлагоджувачем.

Недоліки:

- великий обсяг документації, що може потребувати більше часу для освоєння;
- можливість виникнення проблем з продуктивністю при необережному використанні деяких компонентів.

2.6 Вибір архітектури розробки програмного продукту

Основними аспектами при виборі архітектури побудови ІС є швидкодія, надійність, масштабованість і безпеку.

В даний час найбільш поширеними архітектурою є:

- файл-сервер;
- клієнт-сервер;
- хмарний-сервер;
- багаторівнева архітектура.

Файл-серверна архітектура має на увазі під собою те, що сервер покладає на себе лише функцію зберігання даних, а обробка проводиться на клієнтських машинах. Це означає, що дані необхідно передавати по мережі, що призведе до сильної завантаженні мережевого трафіку. А це в свою чергу призведе до зниження продуктивності при збільшенні числа користувачів. Також при реалізації архітектури файл-сервер, проблема цілісності, узгодженості і одночасного доступу до даних вирішується децентралізовано: дані зберігаються на сервері, а обробляються на клієнті. Внаслідок цього знижується надійність програми. Ще одним недоліком є високі витрати на модернізацію і супровід сервісів бізнес – логіки на кожній клієнтській робочій станції. Однак дана архітектура має і низку переваг, таких як низька вартість

розробки, висока швидкість розробки і невисока вартість оновлення та зміни програмного забезпечення [24].

Архітектура клієнт-сервер позбавлена недоліків вищеописаної архітектури, тому що сервер баз даних не тільки забезпечує доступ до загальних даних, але і виконує їх обробку. Клієнт посилає на сервер запити, на мові "зрозумілою" сервера, а він в свою чергу обробляє запит, контролюючи при цьому цілісність і узгодженість даних, і повертає на клієнт результат відпрацьованого запиту. В результаті навантаження на мережу знижується: клієнтові більше не потрібно обробляти проміжні дані. Зберігання та обробка проводиться централізовано, тому дана архітектура надійніше архітектури файл-сервер. До недоліків клієнт-серверної архітектури відносяться, по-перше, достатня складність розробки системи через необхідність виконувати бізнес-логіку і забезпечувати інтерфейс з користувачем в одній програмі і високі вимоги до робочих станцій з тієї ж причини [25].

Наступною сходинкою розвитку архітектур ІС стала багаторівнева архітектура, в якій бізнес-логіка виконується на сервері додатків. Багаторівнева архітектура володіє наступними перевагами:

- масштабованість;
- конфігурованість – ізольованість рівнів один від одного дозволяє швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів;
- висока безпека;
- висока надійність;
- низькі вимоги до швидкості каналу (мережі) між терміналами і сервером додатків;
- низькі вимоги до продуктивності і технічним характеристикам терміналів, як наслідок зниження їх вартості.

Однак, незважаючи на незаперечні переваги, дана система не набула поширення, з наступних причин:

– складність розробки систем на основі багаторівневої архітектури, т.к дуже складно "зістикувати" різні модулі, особливо якщо вони написані різними групами. А зміна в одному модулі, як правило, викликає лавиноподібні зміни в інших, і з цієї точки зору навіть просту систему, засновану на багаторівневій архітектурі, буде складніше виконати в 2 рази;

– високі вимоги до продуктивності серверів додатків і сервера бази даних, а, значить, і висока вартість серверного обладнання;

– високі вимоги до швидкості каналу (мережі) між сервером бази даних і серверами додатків;

– висока складність адміністрування.

Розглянувши всі достоїнства і недоліки кожної з архітектур, для реалізації системи "AIC All Tours" вибираємо архітектуру клієнт-сервер. Дана архітектура дозволяє оптимально розподілити роботу між клієнтськими і серверними частинами системи: додаток, що працює на робочій станції, не читає записи бази даних "прямо", а посилає запити на сервер, де вони послідовно обробляються, а результати обробки відсилаються на робочу станцію. А це суттєво скорочує інформаційні потоки в ЛВС. Схема функціонування та побудови інформаційної системи, (рис. 2.1) [25].



Рисунок 2.1 – Організаційна модель клієнт-сервер [25]

2.7 Опис застосованих інструментів розробки

Для реалізації був використаний PHP фреймворк Yii2. Yii2 працює за принципом MVC – model, view, controller [26].

CRM розроблена у відповідність із цим Патерном, логічна модель може бути перенесена на іншу платформу

Javascript використовувався як допоміжний інструмент, залежні один від одного списки працюють за принципом відправки і поновлення запитів за допомогою JavaScript і компонент Bootstrap, який так само був використаний під час реалізації.

CRM складається з залежних через зв'язку в БД, які відображені в моделі, CRUD модулів (модуль користувачів йде окремо, через плагін Dektrium Yii2 User і RBAC role based access control): CRUD міст, готелів, пропозицій, вимог, заявок [27]-[28].

2.8 Процес створення модулів

Процес створення модуля:

- створення таблиці в БД і зв'язів;
- створення моделі (в папці models).

В моделі прописується зв'язок за базою даних, встановлюється зв'язок псевдонімів, правила валідації, вказуються зв'язки.

Процес створення моделі:

- підключити namespace app\models й use Yii;
- створити клас Country extends \yii\db\ActiveRecord;
- в класі описати методи;
- валідація зберігається в методі rules();
- псевдоніми знаходяться в методі attributeLabels();
- описати зв'язки.

Примітка: методи відповідають за зв'язки, називати строго в стилі прикладу і у відповідності до самого зв'язку, наприклад – `getCountryRequirements ()` – зв'язок багато до багатьох між країною і вимогами.

2.9 Створення контролера

Контролер створювати в папці `controllers`, назвати в стилі `CountryController.php`.

Контролер відловлює і обробляє запити від клієнта до сервера, рендерить сторінки, обробляє запити на збереження, видалення, зміна тощо:

- підключити `namespace namespace app \ controllers, use Yii, use yii \ web \ Controller` і тп;
- створити клас `CountryController extends \ yii \ web \ Controller` в класі описати методи.

Метод `behaviors` – відповідає за права і повноваження різних груп користувачів по відношенню до контролера.

Методи серії `action / * Ім'я Дія *` – оброблюємо відповідний імені запит браузера.

Іноді бувають допоміжні методи, для отримання об'єктів з БД.

Створення уявлення – папка `view`, а в ній папка з назвою `/ * ім'я-модуля */` наприклад: `country`.

Пошукова модель: `CountySearch.php` містить правила по аналогії з основною, зазвичай генерується і не потребує ручних доробках.

Все вищесказане справедливо для всіх самописних модулів представлених в CRM.

Папка `country` містить наступні файли:

- `_form.php` – універсальна форма для редагування і створення об'єкта;
- `_search.php` – аналог `_form.php`, відповідає за саму можливість пошуку;

- create.php – містить верстку і викликає _form.php;
- update.php – містить верстку і викликає _form.php;
- view.php – містить верстку, показує конкретний об'єкт, викликається з index.php;
- index.php – містить верстку, показує список всіх об'єктів, надає доступ до всіх інших функцій.

У кожного модуля свій не залежний від інших, але схожий по функціоналу набір.

Примітка: 80% функцій нам надає сам фреймворк, а ми збираємо їх як конструктор.

Примітка: методи і поля викликаються не через точку, а через стрілку \$ model → Name.

Примітка: назвати методи можна лише строго по нотації actionIndex, де перше слово – службове, друге – наша назва.

Повна інформація вихідного коду контролерів дивитись у додатку Д.

2.10 Опис бібліотеки плагінів

Dektrium User и RBAC призначений для користувача плагін розроблений для керування користувача, підтримує безліч функцій таких як: створення, блокування, видалення користувача.

Створення груп користувачів, зарахування користувача до групи:

- yii\data\ActiveDataProvider – свого роду помічник-шаблон для спрощення доступу до БД;
- yii\base\Model – доступ до базових методів моделі;
- yii\web\NotFoundHttpException – підключення можливості викликати відповідні виключення;
- yii\filters\VerbFilter – доступ до різноманітних фільтрів, в даному випадку до текстового;
- app\models\Country – доступ до іншої, зовнішньої моделі (наприклад

до країн);

- use yii\widgets\ActiveForm – вбудований віджет який дозволяє легко створювати і оперувати формами;

- use yii\helpers\ArrayHelper – дає доступ до масивів з розширеними можливостями, використовується для представлення даних у випадних списках зокрема.

2.11 Ієрархія даних проєкту

Верхнє меню (смужка) перебуває в папці views \ layouts \ main.php вона реалізована за допомогою вбудованого віджета, який в свою чергу заснований на Bootstrap

Верстка меню менеджера перебуває в папці views \ manager \ index.php.

Верстка іконок з офферу на головному меню і їх подробиць знаходиться в views \ site \ viewoffers.php і в views \ site \ showdetail.php відповідно.

Головний index.php лежить в папці web.

Стилі лежать в папці web \ css.

Підключаються і лежать рідні плагіни в папці vendor.

Підключення бази даних знаходиться в config \ db.php.

Підключення зовнішніх плагінів і модулів, а також налаштування сайту відправки листів знаходиться в config \ web.php.

2.12 Структурна схема програмного продукту

Для зображення структурної схеми програмного продукту будемо використовувати діаграми типу Use case (Варіанти використання)

Use case (Варіанти використання) – це функціональне вимога. Даний вид діаграм в основному використовується для опису функціональних вимог до

системи, для опису предметної області з метою кращого розуміння функціонування системи, (рис. 2.2-2.7) [29].

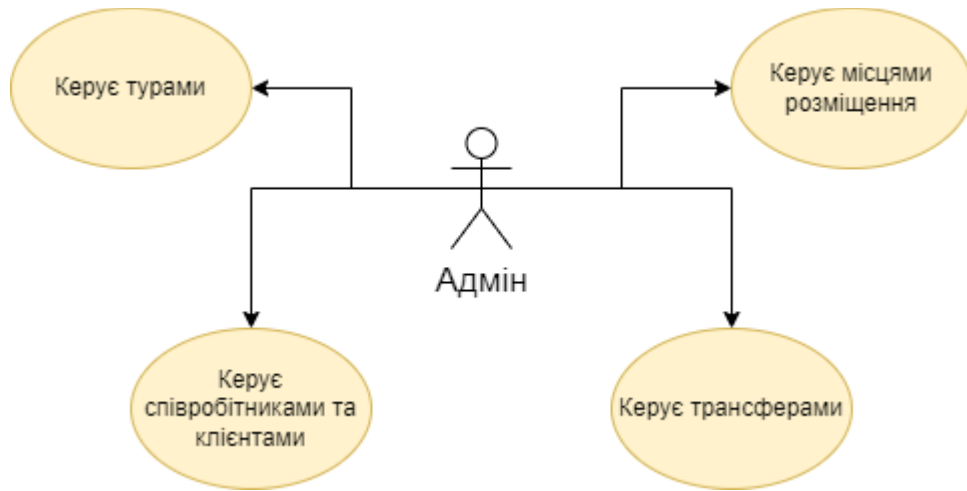


Рисунок 2.2 – Діаграма варіантів використання Адмін [29]

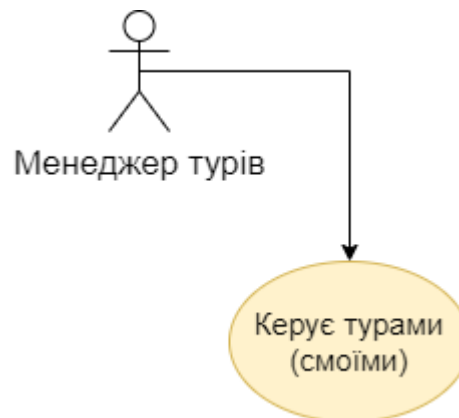


Рисунок 2.3 – Діаграма варіантів використання Менеджер турів [29]

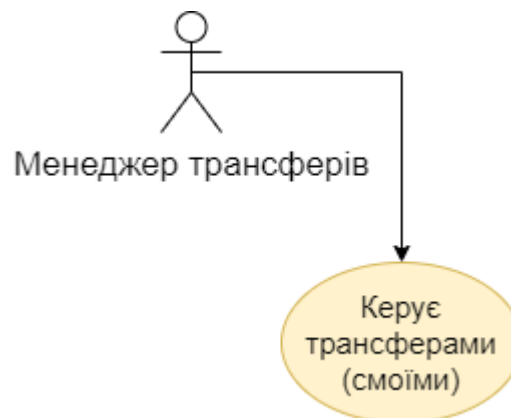


Рисунок 2.4 – Діаграма варіантів використання Менеджер трансферів [29]

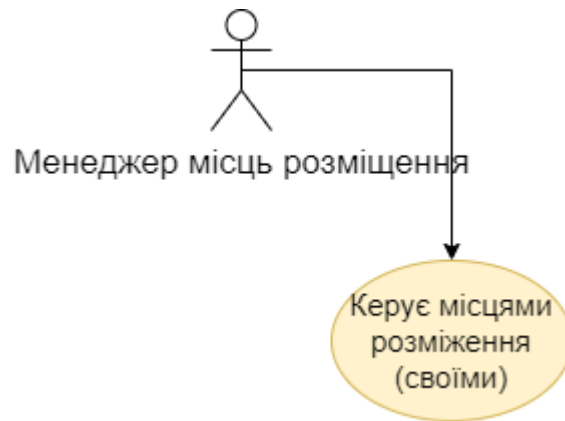


Рисунок 2.5 – Діаграма варіантів використання Менеджер місць розміщення [29]

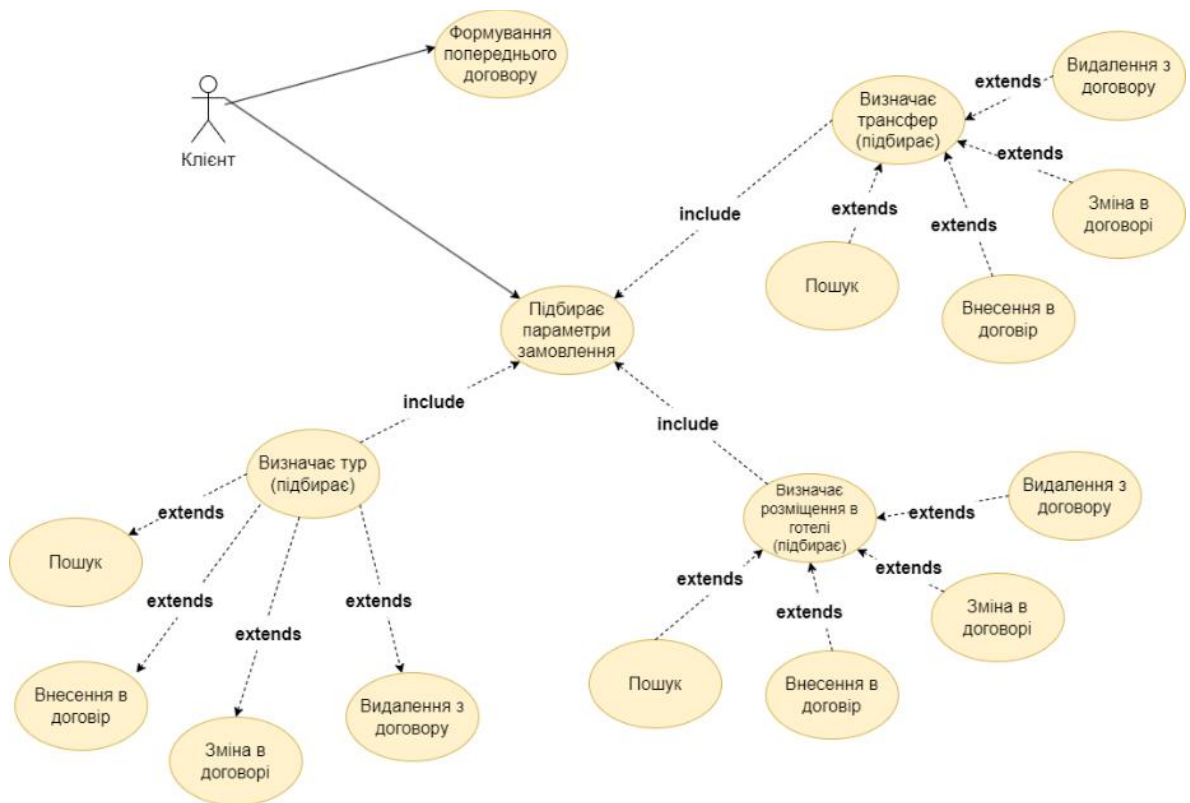


Рисунок 2.6 – Діаграма варіантів використання Клієнт [29]

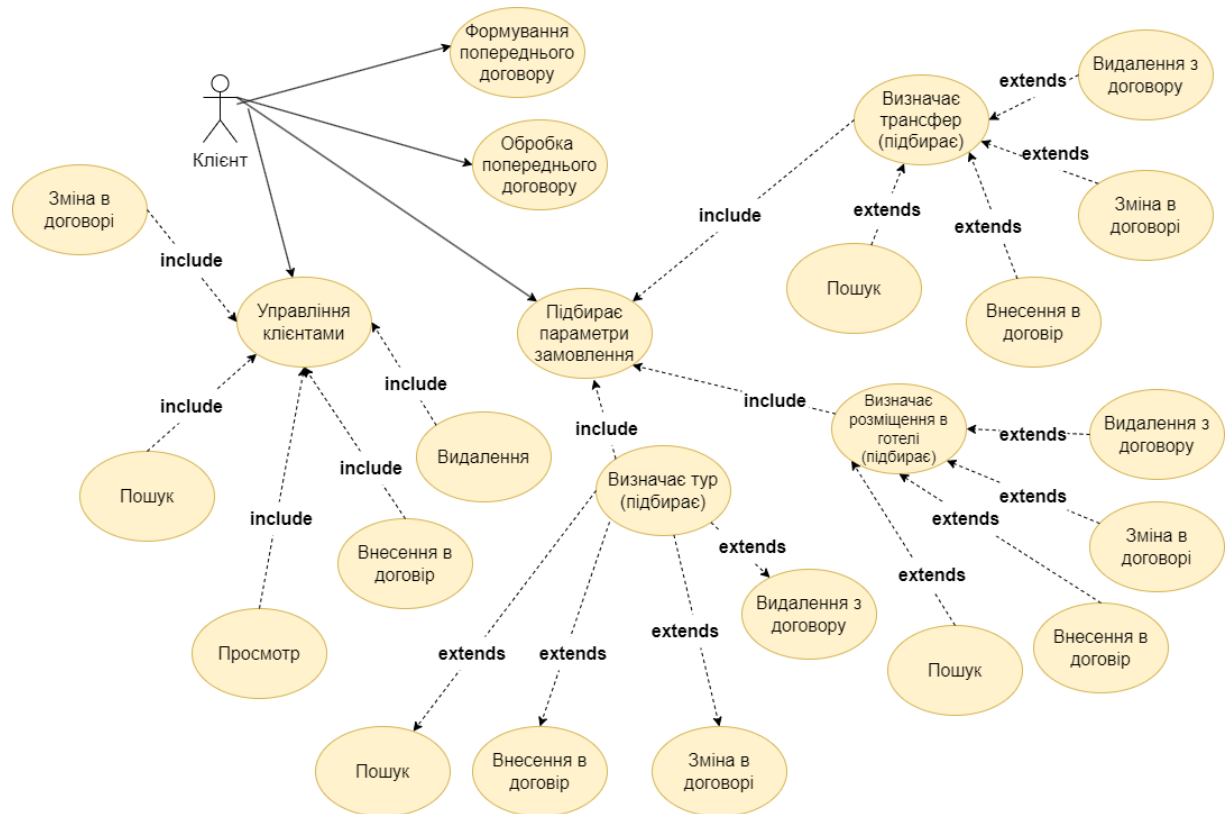


Рисунок 2.7 – Діаграма варіантів використання Менеджер фірми [29]

2.13 Структурна схема програмного продукту

Для кожного сценарію діаграми варіантів використання представлені діаграми послідовностей і кооперацій.

Діаграма послідовності – це діаграма, на якій зображено впорядковане в часі взаємодія об’єктів. На ній зображуються взаємодії об’єкти і послідовність повідомлень, якими вони обмінюються.

Опишімо специфікацію варіантів використання за Коберном дивись у додатку Г.

2.14 Діаграма класів

Діаграма класів – призначені для моделювання структури об'єктно-орієнтованих програм - класів, їх атрибутів і заголовків методів, успадкування, а також зв'язків класів один з одним.

Клас – це основний будівельний блок. Це поняття є і в об’єктно орієнтованих мовах програмування, тобто між класами UML і програмними класами є відповідність, що є основою для автоматичної генерації програмних кодів або для виконання реінжинірингу. Кожен клас має назву, атрибути і операції. Клас на діаграмі показується у вигляді прямокутника, розділеного на 3 області. У верхній міститься назва класу, в середній – опис атрибутів (властивостей), в нижній – назви операцій – послуг, що надаються об’єктами цього класу, (рис. 2.8) [30].

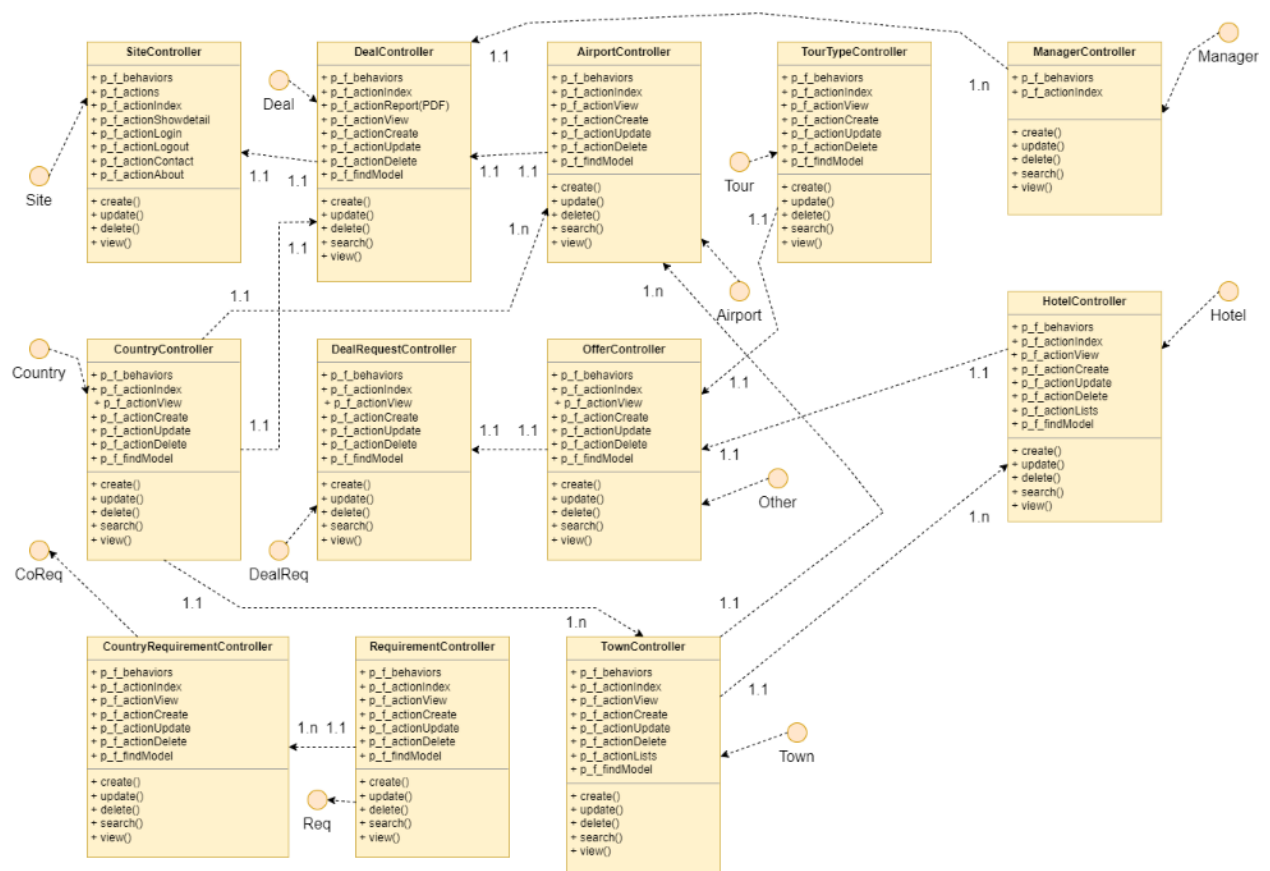


Рисунок 2.8 – Діаграма класів [30]

2.15 Діаграма компонентів

Діаграма компонентів відображає організацію компонентів і залежності між ними. Компонент – це фізичний елемент реалізації (компоненти вихідного коду, бінарного коду, що виконуються компоненти) з чітко визначеним інтерфейсом, призначений для використання в якості змінної частини системи. Кожен компонент являє собою реалізацію деяких класів системи, (рис. 2.9).

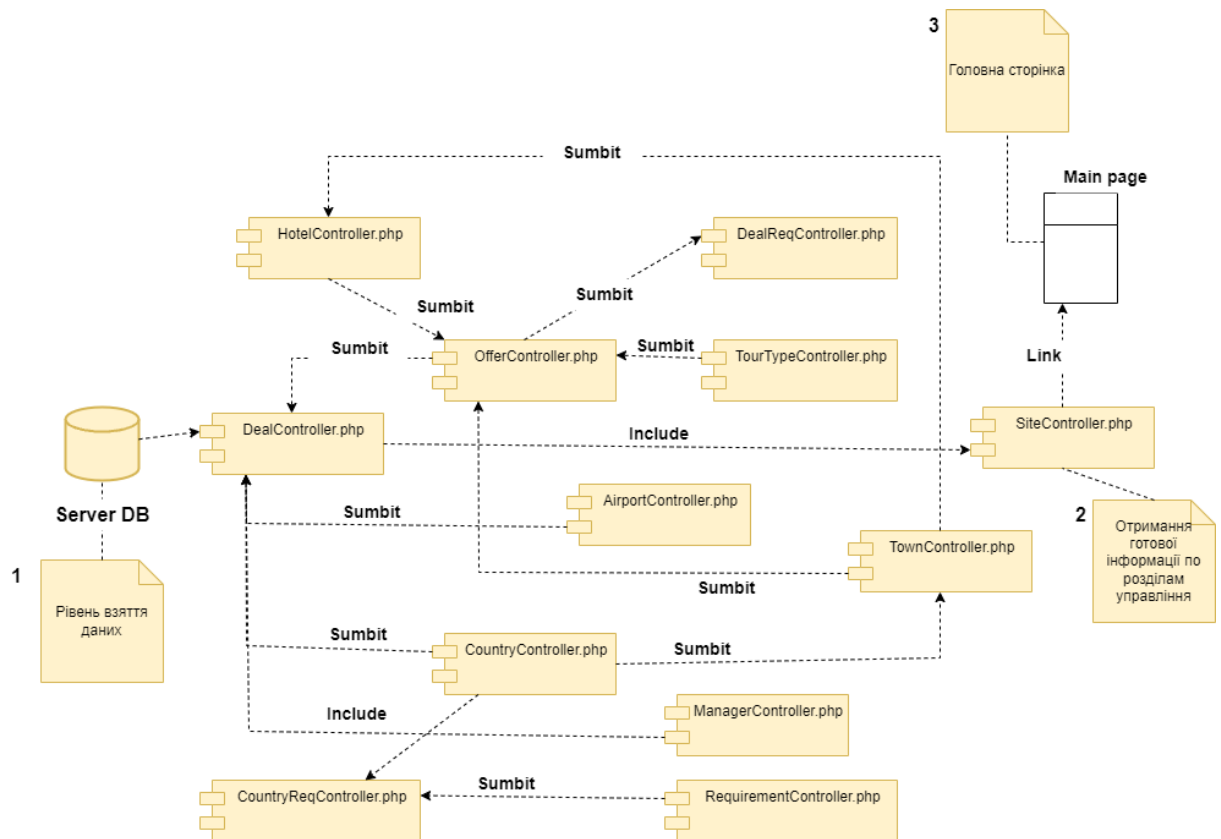


Рисунок 2.9 – Діаграма компонентів [31]

2.16 Діаграма розгортання

Діаграма розгортання відображає фізичні взаємозв'язку між програмними і апаратними компонентами системи, що розробляється. Одним

з істотних понять даного виду діаграм є поняття вузол. Вузол є певний тип обчислювального пристрою, як правило, самостійну частину апаратури, (рис. 2.10).

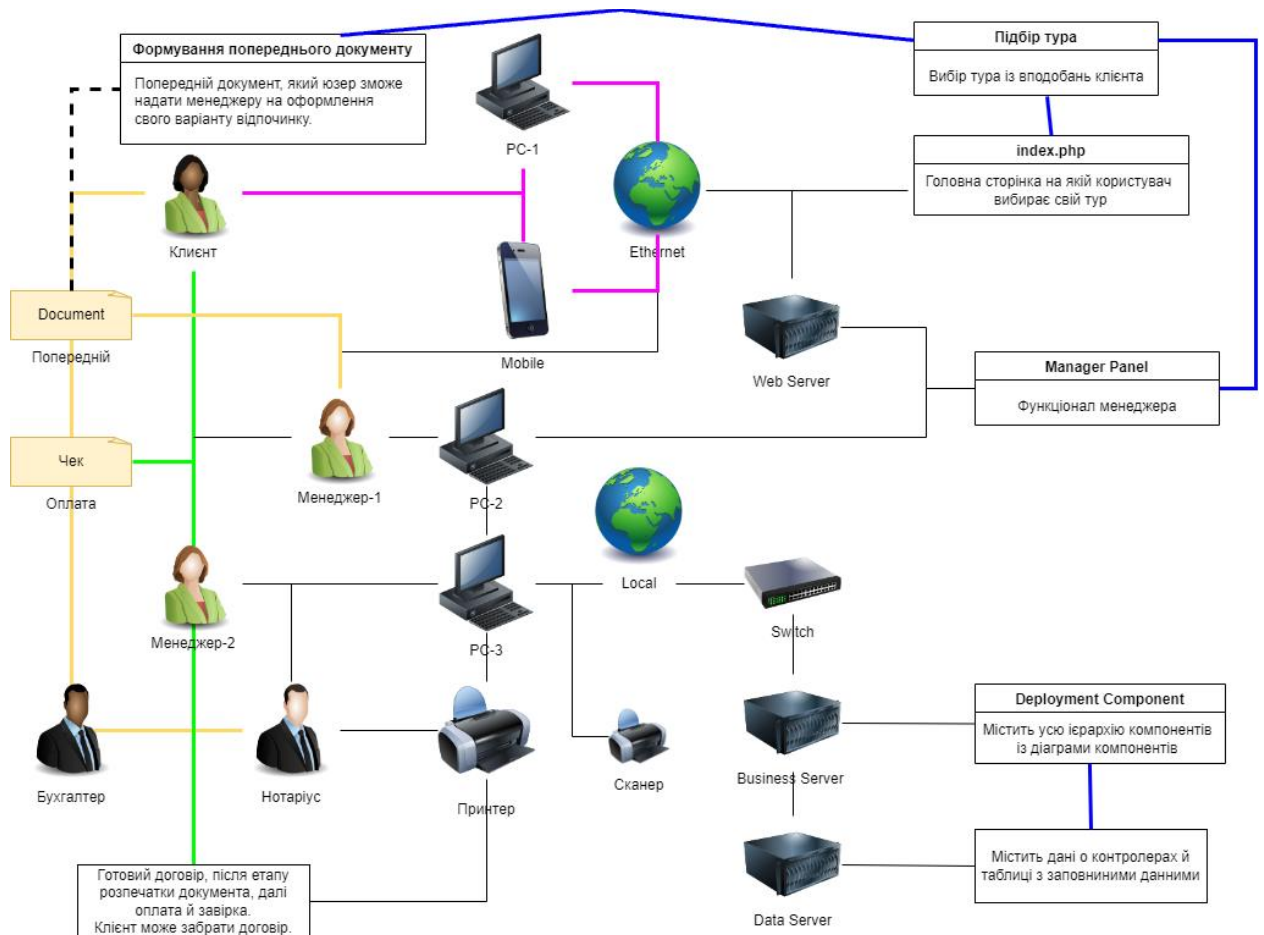


Рисунок 2.10 – Діаграма розгортання [32]

2.2 Висновок до розділу 2

У цьому розділі була можливість зрозуміти важливі аспекти архітектури та розробки програмного продукту "AIC All Tours".

Вибір архітектури є стратегічним кроком, оскільки він впливає на швидкість, надійність, масштабованість і безпеку вашої системи.

У цьому випадку була обрана архітектура клієнт/сервер, оскільки вона дозволяє ефективно розподіляти роботу між клієнтською та серверною

частинами системи.

Використання фреймворку Yii2 та інших інструментів, таких як JavaScript і bootstrap, сприяло швидкій та ефективній розробці програмних продуктів.

Використання Model, Controller, View (підхід MVC) у Yii2 зберігає код упорядкованим і полегшує обслуговування системи.

Створення модулів, включаючи створення таблиць бази даних, моделей і контролерів, вважалося одним із найважливіших кроків розробки.

Було детально показано процес створення моделей і контролерів, з особливим наголосом на важливості підтримки цілісності та узгодженості даних.

Плагіни, такі як Dektrium User і RBAC, тепер дозволяють ефективно керувати користувачами та ролями.

Інші важливі бібліотеки та плагіни, такі як ActiveRecord і ActiveForm, також сприяють розробці програмних продуктів.

Варіанти використання, класи, компоненти та діаграми розгортання забезпечують чіткий огляд архітектури та взаємодії різних компонентів системи. Вони є важливими інструментами для аналізу, розуміння та передачі ключових аспектів проєкту між розробниками та зацікавленими сторонами.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У роботі розглядаються ключові рішення щодо реалізації програмного забезпечення для автоматизації діяльності туристичного агентства. Включаються аспекти вибору програмних інструментів, методів програмування, а також підходів до розробки, які забезпечують ефективне впровадження інноваційних рішень. Робота також зосереджена на аналізі найкращих практик та наявних технічних рішень у галузі, з метою створення оптимальної та ефективної системи.

Для досягнення поставленої мети у роботі необхідно розробити систему, яка інтегрує технічні рішення і методи, спрямовані на поліпшення бізнес-процесів у туристичних агентствах.

3.1 Попередній огляд критеріїв оцінки

Ефективність впровадження програмного продукту тісно пов'язана з автоматизація діяльності, а саме цей зв'язок полягає у вдосконаленні процесів обслуговування клієнтів, збільшенні продуктивності та оптимізації внутрішніх робочих процедур. Автоматизовані системи можуть вести облік бронювань, управління запасами та фінансовими операціями, значно зменшуючи витрати часу та ресурсів. Таким чином, ефективне впровадження таких систем безпосередньо впливає на збільшення конкурентоздатності та прибутковості туристичного агентства. В якості параметрів оцінки функціонування туристичного агентства були виділені:

а) параметр А. «Обсяг продажів (у грн)»,

де A_{old} – обсяг продажів до впровадження програмного продукту;

A_{new} – обсяг продажів після впровадження програмного продукту;

б) Параметр В. «Відсоток успішних бронювань»,

де B_{old} – кількість успішних бронювань до впровадження програмного продукту;

B_{new} – кількість успішних бронювань після впровадження програмного продукту;

с) параметр С. «Відсоток повторних клієнтів»,

де C_{old} – відсоток повторних клієнтів до впровадження програмного продукту.

C_{new} – відсоток повторних клієнтів після впровадження програмного продукту;

д) параметр D. «Рівень задоволеності клієнтів (шкала від 1 до 10)»,

де D_{old} – рівень задоволеності клієнтів до впровадження програмного продукту;

D_{new} – рівень задоволеності клієнтів після впровадження програмного продукту;

е) параметр E. «Ефективність маркетингових кампаній (%)»,

де E_{old} – ефективність маркетингових кампаній до застосування програмного продукту;

E_{new} – ефективність маркетингових кампаній після впровадження програмного продукту;

є) параметр F. Середній час обробки замовлення (у годиннику),

де F_{old} – середній час обробки замовлення до впровадження програмного продукту;

F_{new} – середній час обробки замовлення після впровадження програмного продукту;

ж) параметр G. «Відсоток автоматизації бронювань (%)»,

де G_{old} – відсоток автоматизованих бронювань до впровадження програмного продукту;

G_{new} – відсоток автоматизованих бронювань після впровадження програмного продукту;

з) параметр H. «Інтеграція із зовнішніми сервісами (%)»,

де H_{old} – рівень інтеграції із зовнішніми сервісами до застосування програмного продукту;

H_{new} – рівень інтеграції із зовнішніми сервісами після впровадження програмного продукту;

и) параметр І. «Рівень безпеки даних (шкала від 1 до 10)»,

де I_{old} – рівень безпеки даних до впровадження програмного продукту;

I_{new} – рівень безпеки даних після впровадження програмного продукту;

і) параметр J. «Вартість залучення клієнта (у грошах)»,

де J_{old} – середня вартість залучення клієнта до впровадження програмного продукту;

J_{new} – середня вартість залучення клієнта після впровадження програмного продукту;

Далі додатково до кожного параметра встановлюються вагові коефіцієнти w_i , які в сумі не повинні перевищувати 1.

Для подальшого розрахунку нормалізованих даних визначимо ваги до кожного вищенаведеного параметру:

- параметр А. «Обсяг продажів (у грн)» значення ваги дорівнює 0,06;
- параметр В. «Відсоток успішних бронювань» значення ваги дорівнює 0,07;
- параметр С. «Відсоток повторних клієнтів» значення ваги дорівнює 0,09;
- параметр D. «Рівень задоволеності клієнтів (шкала від 1 до 10)» значення ваги дорівнює 0,09;
- параметр Е. «Ефективність маркетингових кампаній (%)» значення ваги дорівнює 0,06;
- параметр F. Середній час обробки замовлення (у годиннику) значення ваги дорівнює 0,20;
- параметр G. «Відсоток автоматизації бронювань (%)» значення ваги дорівнює 0,15;

- параметр Н. «Інтеграція із зовнішніми сервісами (%)» значення ваги дорівнює 0,13;
- параметр І. «Рівень безпеки даних (шкала від 1 до 10)» значення ваги дорівнює 0,08;
- параметр J. «Вартість залучення клієнта (у грошах)» значення ваги дорівнює 0,07.

Як видно з виділених параметрів оцінки для ефективності процесу автоматизації роботи, а також для функціонування туристичного агентства, найбільш важливими параметрами оцінки є «Параметр F. Середній час обробки замовлення (у годиннику) зі значення ваги 0,20; Параметр G. «Відсоток автоматизації бронювань (%)» значення ваги 0,15; Параметр Н. «Інтеграція із зовнішніми сервісами (%)» значення ваги 0,13.

Маючи визначені параметри та їх ваги застосуємо обрані методи оцінки ефективності для функціонування та автоматизації процесів туристичного агентства.

3.2 Застосування обраних методів

Застосуємо метод нормалізації обраних параметрів, будемо приводити дані до виду min-max нормалізації для перерахунку отриманих значень параметрів в діапазоні від 0 до 1. Для кожного параметра а значення нормалізованого параметра х розрахуємо наступним чином (3.1):

$$a = \frac{a - \min(a)}{\max(a) - \min(a)} \quad (3.1)$$

де а – вихідне значення параметра;

min (a) – мінімальне значення параметра;

max (a) – максимальне значення параметра.

Виконаємо нормалізацію визначених параметрів за методом лінійної комбінації даних:

а) параметр А. «Обсяг продажів (у грн)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Travel Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) A_{H1} = \frac{1\,600\,000-0}{4\,000\,000-0} = 0,40;$$

$$2) A_{H2} = \frac{1\,050\,000-0}{4\,000\,000-0} = 0,26;$$

$$3) A_{H3} = \frac{2\,900\,000-0}{4\,000\,000-0} = 0,72;$$

$$4) A_{H4} = \frac{2\,800\,000-0}{4\,000\,000-0} = 0,70;$$

б) параметр В. «Відсоток успішних бронювань». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer», «RockyRoad Travel _2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) B_{H1} = \frac{40-0}{100-0} = 0,40;$$

$$2) B_{H2} = \frac{25-0}{100-0} = 0,25;$$

$$3) B_{H3} = \frac{72-0}{100-0} = 0,72;$$

$$4) B_{H4} = \frac{70-0}{100-0} = 0,70;$$

в) параметр С. «Відсоток повторних клієнтів». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Travel Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) C_{H1} = \frac{20-0}{100-0} = 0,20;$$

$$2) C_{H2} = \frac{13-0}{100-0} = 0,13;$$

$$3) C_{H3} = \frac{31-0}{100-0} = 0,31;$$

$$4) C_{H4} = \frac{29-0}{100-0} = 0,29;$$

г) параметр D. «Середній рівень задоволеності клієнтів (шкала від 1 до 10)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer H_1», «RockyRoad Travel H_2» «365 Adventures H_3», у порівнянні з нашим розробленим продуктом «AIC Alltours H_4»:

Розраховані значення:

$$1) D_{H1} = \frac{5-0}{10-0} = 0,5;$$

$$2) D_{H2} = \frac{4-0}{10-0} = 0,4;$$

$$3) D_{H3} = \frac{8-0}{10-0} = 0,8;$$

$$4) D_{H4} = \frac{7-0}{10-0} = 0,7;$$

д) параметр E. «Ефективність маркетингових кампаній (%)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer H_1», «RockyRoad Travel H_2» «365 Adventures H_3», у порівнянні з нашим розробленим продуктом «AIC Alltours H_4»:

Розраховані значення:

$$1) E_{H1} = \frac{24-0}{100-0} = 0,24;$$

$$2) E_{H2} = \frac{19-0}{100-0} = 0,19;$$

$$3) E_{H3} = \frac{45-0}{100-0} = 0,45;$$

$$4) E_{H4} = \frac{40-0}{100-0} = 0,40;$$

е) параметр F. Швидкість обробки замовлення (шкала від 1 до 10). Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Travel Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) F_{H1} = \frac{4-0}{10-0} = 0,40;$$

$$2) F_{H2} = \frac{3-0}{10-0} = 0,30;$$

$$3) F_{H3} = \frac{8-0}{10-0} = 0,80;$$

$$4) F_{H4} = \frac{7-0}{10-0} = 0,70;$$

є) параметр G. «Відсоток автоматизації бронювань (%)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Travel Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) G_{H1} = \frac{40-0}{100-0} = 0,40;$$

$$2) G_{H2} = \frac{30-0}{100-0} = 0,30;$$

$$3) G_{H3} = \frac{80-0}{100-0} = 0,80;$$

$$4) G_{H4} = \frac{70-0}{100-0} = 0,70;$$

ж) параметр H. «Інтеграція із зовнішніми сервісами (%)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) H_{H1} = \frac{40-0}{100-0} = 0,40;$$

$$2) H_{H2} = \frac{50-0}{100-0} = 0,50;$$

$$3) H_{H3} = \frac{80-0}{100-0} = 0,80;$$

$$4) H_{H4} = \frac{80-0}{100-0} = 0,80;$$

з) параметр І. «Рівень безпеки даних (шкала від 1 до 10)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) I_{H1} = \frac{7-0}{10-0} = 0,70;$$

$$2) I_{H2} = \frac{5-0}{10-0} = 0,50;$$

$$3) I_{H3} = \frac{8-0}{10-0} = 0,80;$$

$$4) I_{H4} = \frac{8-0}{10-0} = 0,80;$$

и) параметр J. «Рекламна вартість залучення клієнта (у грн)». Виконаємо нормалізацію даних відповідно до програмних продуктів конкурентів «Reliable Trip Designer Н_1», «RockyRoad Н_2» «365 Adventures Н_3», у порівнянні з нашим розробленим продуктом «AIC Alltours Н_4»:

Розраховані значення:

$$1) J_{H1} = \frac{50-0}{200-0} = 0,25;$$

$$2) J_{H2} = \frac{70-0}{200-0} = 0,35;$$

$$3) J_{H3} = \frac{35-0}{200-0} = 0,17;$$

$$4) J_{H4} = \frac{35-0}{200-0} = 0,17.$$

Отримавши значення нормалізованих параметрів виконаємо порівняння ефективності розробленого програмного продукту для функціонування туристичного агентства за продуктам 3 конкурентів та індивідуально розробленого нашого.

Після нормалізації параметрів можна застосувати вагові коефіцієнти для врахування їх відносної важливості. Вагові коефіцієнти можуть бути

задані на основі експертних оцінок або інших методів оцінки важливості кожного параметра.

Загальна оцінка обчислюється як зважена сума нормалізованих параметрів представлених вище відповідно до формули (3.12):

$$S = w_1 * a_1 + w_2 * a_2 + w_3 * a_3 \dots + w_n * a_n \quad (3.12)$$

де S – загальна оцінка;

$w_1, w_2 \dots w_n$ – вагові коефіцієнти;

$a_1, a_2, \dots a_n$ – нормалізовані значення параметрів.

Розрахуємо оцінку ефективного впровадження програмного засобу за трьома обраними конкурентами програмного продукту та індивідуально розробленого нашого:

$$- S_{H_1} = (0,06*0,4) + (0,07*0,4) + (0,09*0,2) + (0,09*0,5) + (0,06*0,24) + (0,2*0,4) + (0,15*0,15) + (0,13*0,4) + (0,08*0,7) + (0,07*0,25) = 0,36;$$

$$- S_{H_2} = (0,06*0,26) + (0,07*0,25) + (0,09*0,13) + (0,09*0,4) + (0,06*0,19) + (0,2*0,3) + (0,15*0,3) + (0,13*0,5) + (0,08*0,5) + (0,07*0,35) = 0,33;$$

$$- S_{H_3} = (0,06*0,72) + (0,07*0,72) + (0,09*0,31) + (0,09*0,8) + (0,06*0,45) + (0,2*0,8) + (0,15*0,8) + (0,13*0,8) + (0,08*0,8) + (0,07*0,17) = 0,68;$$

$$- S_{H_4} = (0,06*0,7) + (0,07*0,7) + (0,09*0,29) + (0,09*0,7) + (0,06*0,4) + (0,2*0,7) + (0,15*0,7) + (0,13*0,8) + (0,08*0,8) + (0,07*0,17) = 0,63.$$

Отже, цей підхід дозволяє врахувати відносну важливість параметрів та надає більш об'єктивну оцінку ефективного впровадження програмного засобу.

В якості визначення критеріїв оцінки ефективного впровадження програмного засобу (S), виділено наступні діапазони:

– діапазон $S = 0.5 - 0.21$, вказує на критичний рівень ефективності розробленого програмного продукту, вимоги та параметри до якого потребують повного перепроектування.

– діапазон $S = 0.22 - 0.32$, вказує на низький рівень ефективності розробленого програмного продукту, який виконує мінімальний набір функцій, та потребує повного перепроектування.

– діапазон $S = 0.33 - 0.54$, вказує на середній рівень ефективності розробленого програмного продукту, який виконує більшість основних функцій, та потребує невеликий об'єм доробки програмного засобу.

– діапазон $S = 0.55 - 0.78$, вказує на достатній рівень ефективності розробленого програмного продукту, який виконує основні функції та має додатковий функціонал різнитися з іншими програмними продуктами.

– діапазон $S = 0.79 - 1$, свідчить про ідеальний рівень ефективності розробленого програмного продукту, який відрізняється від обраного вузького сегменту обраної предметної області, та займає лідируючі позиції серед клієнтів.

Підводячи підсумки використаного методу, було виділено, що розроблений нами програмний продукт підпадає під діапазон $S = 0.55 - 0.78$ з оцінкою 0,63 і наш близький конкурент 0,68 та вказує на достатній рівень ефективності процесу автоматизації на прикладі розробленого програмного продукту, який виконує основні функції та має додатковий функціонал, який відрізняється від інших програмними продуктами. Інші ж конкурентні програмні продукти підпали під $S = 0.33 - 0.54$, з оцінками 0,39 і 0,39 та вказує на середній рівень ефективності розробленого програмного продукту, який виконує більшість основних функцій, та потребує невеликий об'єм доробки програмного засобу.

Додатково до найбільш вагового виділеного параметра F. «Середній час обробки замовлення (у хвиликах)» зі значення ваги 0,20 можна застосувати метод параметричного порівняння експериментальних даних А/В тестування.

В якості загального показника для аналізу даних виступає KPI_{old} та KPI_{new} , які визначають найбільш важливі критерії ефективності впровадженого програмного забезпечення.

В якості загального показника для об'єму вибірки даних виступає N_{old} та N_{new} , які визначають мінімальний та максимальний діапазон аналізованої вибірки даних.

В якості загального показника для визначення кількості даних виступає X_{old} та X_{new} , які визначають кількість успішності відпрацьованих подій.

Також для достовірних розрахунків виділимо параметри для визначення α – рівень значимості (вірогідність помилок першого роду) та β – потужність тесту ($1 - \beta$ – вірогідність помилок другого роду).

Визначивши параметри та критерії оцінки, сформуємо математичну модель оцінки різниці між старим та новим продуктом та сформуємо інтерпретацію формулу наступним чином, див. формулу (3.13):

$$\begin{aligned} H_0 : KPI_{old} &= KPI_{new}, \\ H_1 : KPI_{old} &\neq KPI_{new} \end{aligned} \quad (3.13)$$

де H_0 – нульова гіпотеза (відсутності різниці);

H_1 – альтернативна гіпотеза (з наявною різницею)

Розглянемо формулу (3.14), для розрахунку різниці аналізованих параметрів покращення або погіршення ефективності розроблено програмного продукту.

$$\begin{aligned} KPI_{old} &= \frac{X_{old}}{N_{old}} \\ KPI_{new} &= \frac{X_{new}}{N_{new}} \end{aligned} \quad (3.14)$$

де N_{old} та N_{new} – об'єм вибірки даних;

X_{old} та X_{new} – кількість успішності відпрацьованих подій.

Якщо, результат тесту дозволяє спростувати нульову гіпотезу, це може вказувати на статистичну значимість покращення або погіршення ефективності розроблено програмного продукту.

Використаємо розроблений код на мові програмування PHP для більш швидкого визначення та розрахунку параметричного порівняння експериментальних даних А/В тестування відповідно до обраного параметру розрахунку.

Імпорт бібліотеки для роботи з функціями та статистикою

`include("math_functions.php");`

Визначення значень параметрів

`$N_old = "Зазначене значення параметра N";`

`$X_old = "Зазначене значення параметра X";`

`$KPI_old = $X_old / $N_old;`

`$N_new = "Зазначене значення параметра N";`

`$X_new = "Зазначене значення параметра X";`

`$KPI_new = $X_new / $N_new;`

Рівень значимості та потужності тесту

`$alpha = 0.05;`

`$beta = 0.2;`

Розрахунок стандартної помилки для двох виборок

`$se_old = sqrt($KPI_old * (1 - $KPI_old) / $N_old);`

```

$$\$se\_new = \sqrt{(\$KPI\_new * (1 - \$KPI\_new) / \$N\_new)};$$

```

```
# Розрахунок стандартної помилки різності
```

```

$$\$se\_diff = \sqrt{(\$se\_old^{**2} + \$se\_new^{**2})};$$

```

```
# Розрахунок t-статистики
```

```

$$\$t\_stat = (\$KPI\_new - \$KPI\_old) / \$se\_diff;$$

```

```
# Розрахунок критичного значення
```

```

$$\$t\_crit = \text{abs}(\text{norm\_inv}(\$alpha/2));$$

```

```
# Розрахунок p-значення
```

```

$$\$p\_value = 2 * (1 - \text{norm\_cdf}(\text{abs}(\$t\_stat)));$$

```

```
# Розрахунок потужності тесту
```

```

$$\$power = 1 - \text{norm\_cdf}(\$t\_crit - \text{abs}(\$t\_stat));$$

```

```
# Отримані результати
```

```
echo "t-статистика: " . \$t_stat . "\n";
```

```
echo "p-значення: " . \$p_value . "\n";
```

```
echo "Потужність тесту: " . \$power . "\n";
```

Інтерпретацію отриманих даних можна, розуміти, як:

- якщо $p_value < \alpha$, то присутня статистично значуща різниця між групами;
- t_stat показує напрямок і силу ефекту: позитивне значення свідчить про поліпшення у новій групі, негативне – поліпшення у старій групі;
- $power$ відображає можливість виявлення статистично значущого ефекту за його наявності.

Відповідно до розробленого коду, визначими кроки його виконання:

- визначаємо визначені дані обраного параметра N_old , X_old для розрахунку $KPI_old = X_old / N_old$;
- задаємо рівень значущості (α) та потужність тесту (β) ;
- виконуємо розрахунок стандартних помилок – P ;
- розраховуємо стандартні помилки для старого та нового продукту;
- розрахунок стандартної помилки різниці – вважаємо стандартну помилку різниці між двома продуктами;
- розрахунок t-статистики – обчислюємо t-статистику, яка відбиває, наскільки статистично значущі різницю між двома групами.
- розрахунок критичного значення:
- розраховуємо критичне значення заданого рівня значимості;
- розрахунок р-значення – визначаємо р-значення для двостороннього тесту;
- розрахунок потужності тесту – обчислюємо потужність тесту для заданого рівня значущості та t-статистики.

Висновок результатів:

- виводимо результати, такі як t-статистика, р-значення та потужність тесту.

3.3 Висновок до розділу 3

У даному розділі було розглянуто ключові рішення щодо реалізації компонентів системи автоматизації для туристичного агентства. Вибір мови програмування та технологій був здійснений з урахуванням специфіки туристичної індустрії та конкретних потреб проєкту. Для оцінки ефективності процесів автоматизації на рівні впровадження програмного продукту були визначені ключові параметри, які включають обсяг продажів, відсоток успішних бронювань, повторних клієнтів, рівень задоволеності клієнтів,

ефективність маркетингових компаній, час обробки замовлення, відсоток автоматизації бронювань, інтеграцію з зовнішніми сервісами, рівень безпеки даних та вартість залучення клієнта.

Для кожного параметру було встановлено вагові коефіцієнти, що вказують на їх значимість у загальному оцінюванні ефективності системи. Найбільш значущими параметрами є час обробки замовлення, відсоток автоматизації бронювань та інтеграція з зовнішніми сервісами, що підкреслює фокус на оптимізацію процесів та покращення інтегрованості системи.

Було запропоновано застосувати метод нормалізації для оцінки зазначених параметрів, що дозволить провести кількісний аналіз впливу впровадження програмного продукту на діяльність туристичного агентства. Цей підхід дасть можливість об'єктивно оцінити результативність системи та її складових, що стане основою для подальших стратегічних розв'язань і поліпшень у роботі агентства.

4 РЕАЛІЗАЦІЯ КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Структура та зміст контролерів

Розроблена програмна реалізація складається з проєкту 5 основних папок, 12 класів/контролерів.

Структура файлової системи:

- /web – містить всю клієнтську частину проєкту;
- /css – містить стилі проєкту;
- /views – містить верстку проєкту;
- /vendor – містить рідні плагіни проєкту;
- /config – містить підключені зовнішні плагіни, модулі, базу даних, а також налаштування відправки листів.

Контролери (рис. 4.1):

- AirportController – управління операціями, пов'язаними з аеропортами;
- CountryController – функції, пов'язані з країнами;
- CountryRequirementController – керування вимогами, специфічними для країн;
- DealController – управління функціоналом угод;
- DealRequestController – керування запитами на угоди;
- HotelController – операції, пов'язані з готелями;
- ManagerController – управління операціями на рівні менеджменту;
- OfferController – керування пропозиціями в системі;
- RequirementController – управління вимогами та критеріями;
- SiteController – керування загальними операціями сайту;
- TourTypeController – управління різними типами турів;
- TownController – функції, пов'язані з містами.

Методи контролерів:

- actionIndex – виводить список ресурсів;

- `actionView` – відображає один конкретний ресурс;
- `actionCreate` – створює новий ресурс;
- `actionUpdate` – оновлює існуючий ресурс;
- `actionDelete` – видаляє ресурс.

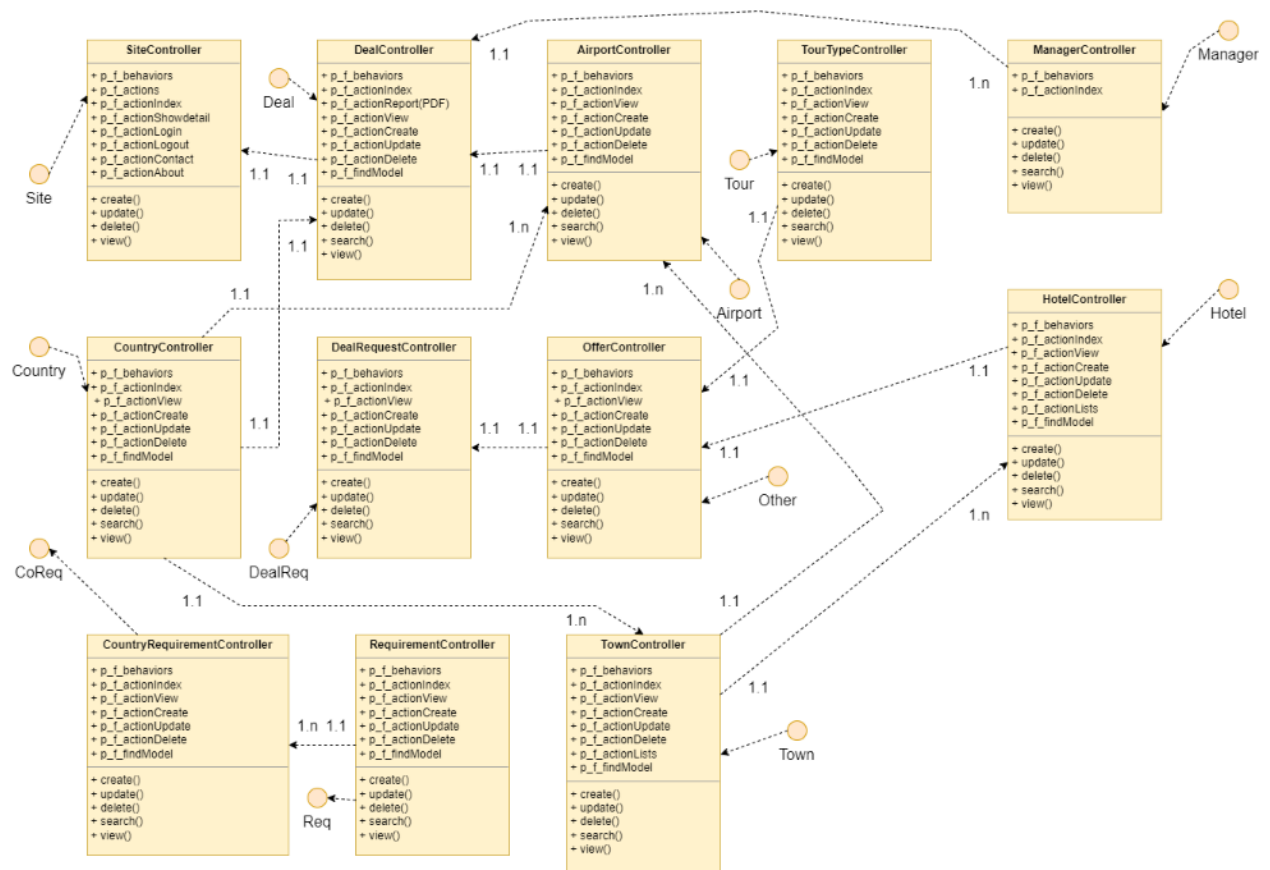


Рисунок 4.1 – Діаграма класів

Додаткові методи, такі як `actionReport` у `DealController` чи `actionLists` у деяких контролерах, є специфічними для конкретних бізнес-вимог цього проекту.

4.2 Реалізовані програмні класи

Класи:

- клас `AuthRule` – управляє правилами аутентифікації;
- клас `AuthItem` – представляє елементи авторизації (наприклад, ролі, дозволи);
 - клас `AuthItemChild` – управління відносинами між різними елементами авторизації;
 - клас `AuthAssignment` – для призначення авторизаційних елементів користувачам;
 - клас `Token` – для роботи з токенами, наприклад, для відновлення пароля або підтвердження електронної адреси;
 - клас `Profile` – для роботи з профілями користувачів;
 - клас `SocialAccount` – для інтеграції з соціальними мережами та обробки інформації про соціальні акаунти;
 - клас `User` – основний клас для управління користувачами системи.

4.3 Реалізація графічного інтерфейсу користувача

При знаходженні на сайті без авторизації Гість бачимо основну сторінку переліком турів, (рис. 4.2).

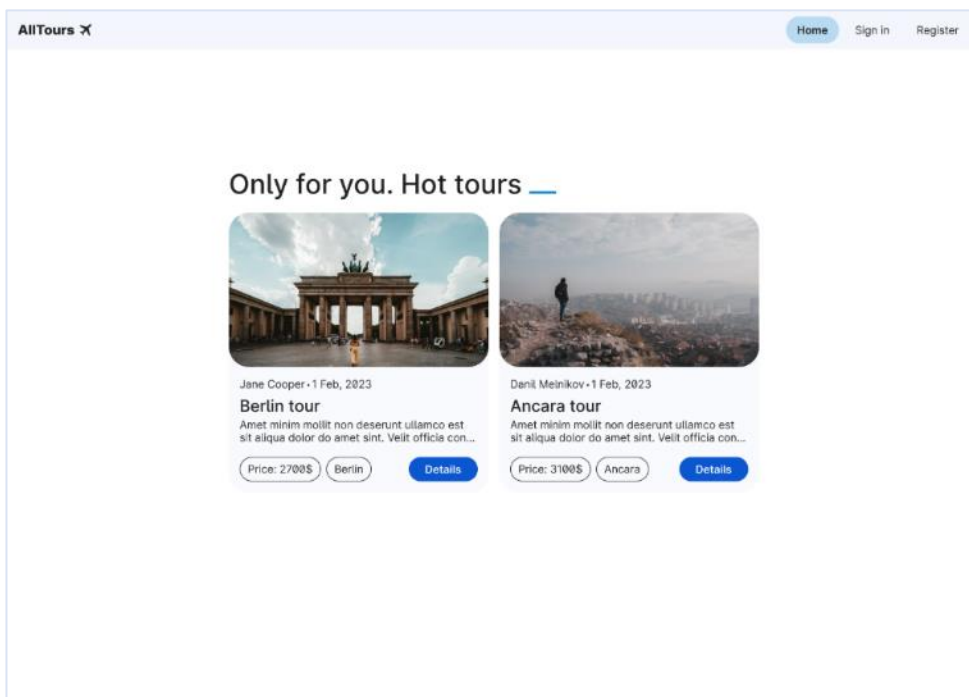


Рисунок 4.2 – Знаходження Гість на сайті

При авторизації за Користувача з'являються додаткова панель, а саме панель персональних запитів клієнта, (рис. 4.3).

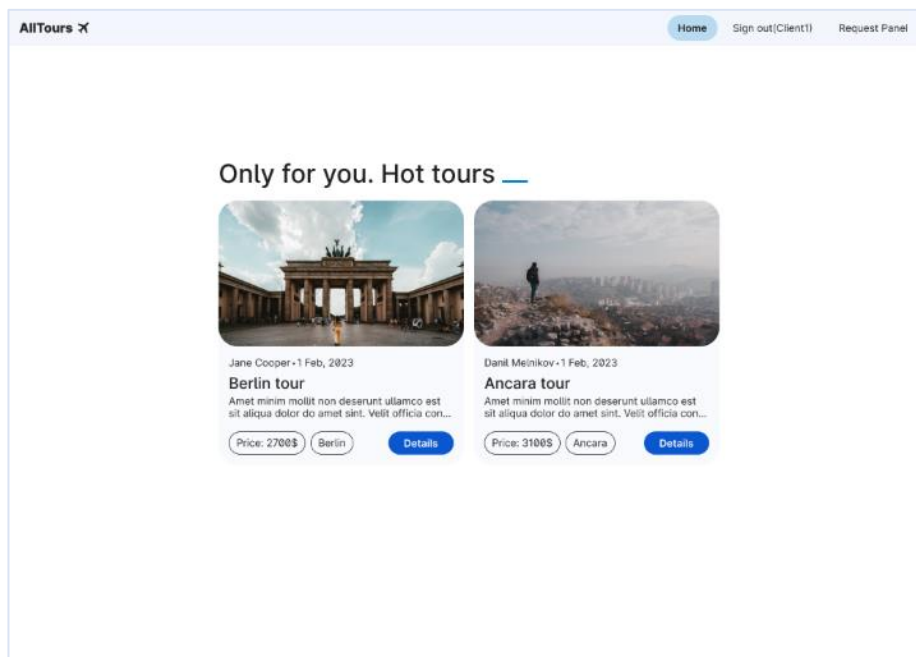


Рисунок 4.3 – Request Panel Користувача

При авторизації за Адміністратор з'являються додаткові панелі, а саме панель адміністратор, панель менеджера і панель запитів, (рис. 7.4).

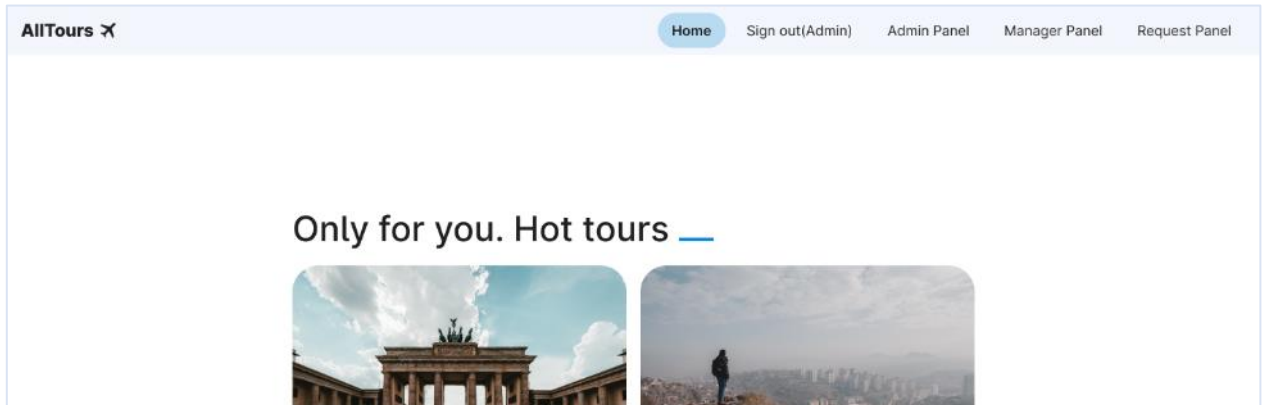


Рисунок 4.4 – Admin Panel, Manager Panel, Request Panel Адміністратора

При авторизації за Менеджера з'являються додаткова панель, а саме панель менеджера для керування турами, (рис. 4.5).

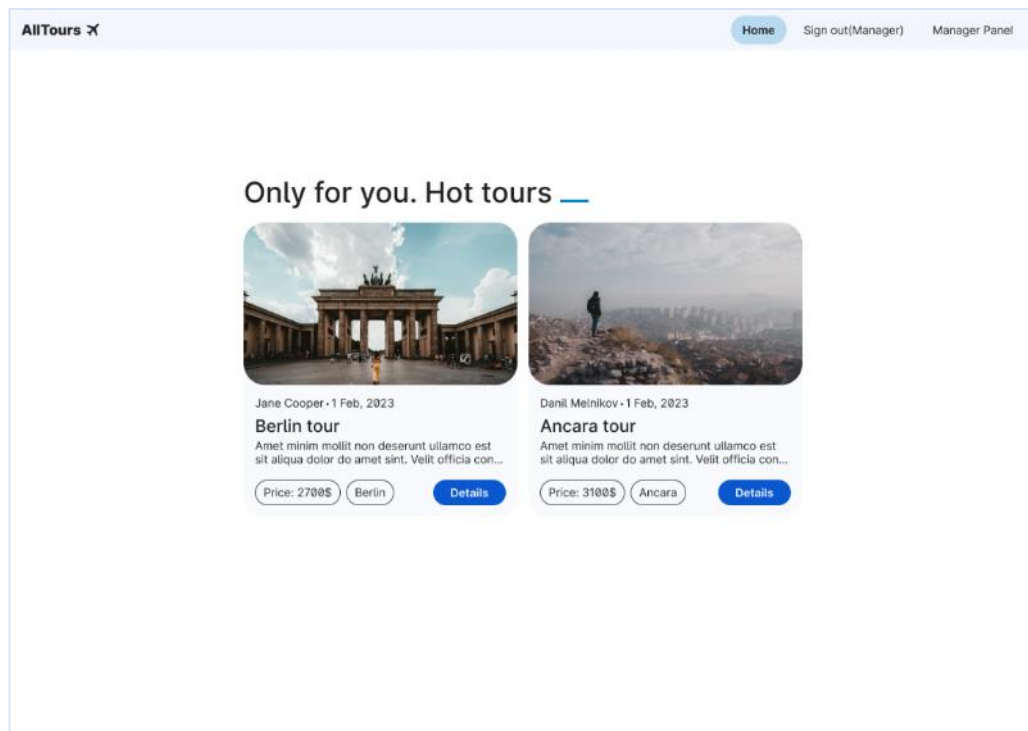


Рисунок 4.5 – Manager Panel Менеджера

При авторизації за Менеджера-поставщика з'являються додаткова панель, а саме панель менеджера-поставщика для заповнення актуальних партнер даних, (рис. 4.6).

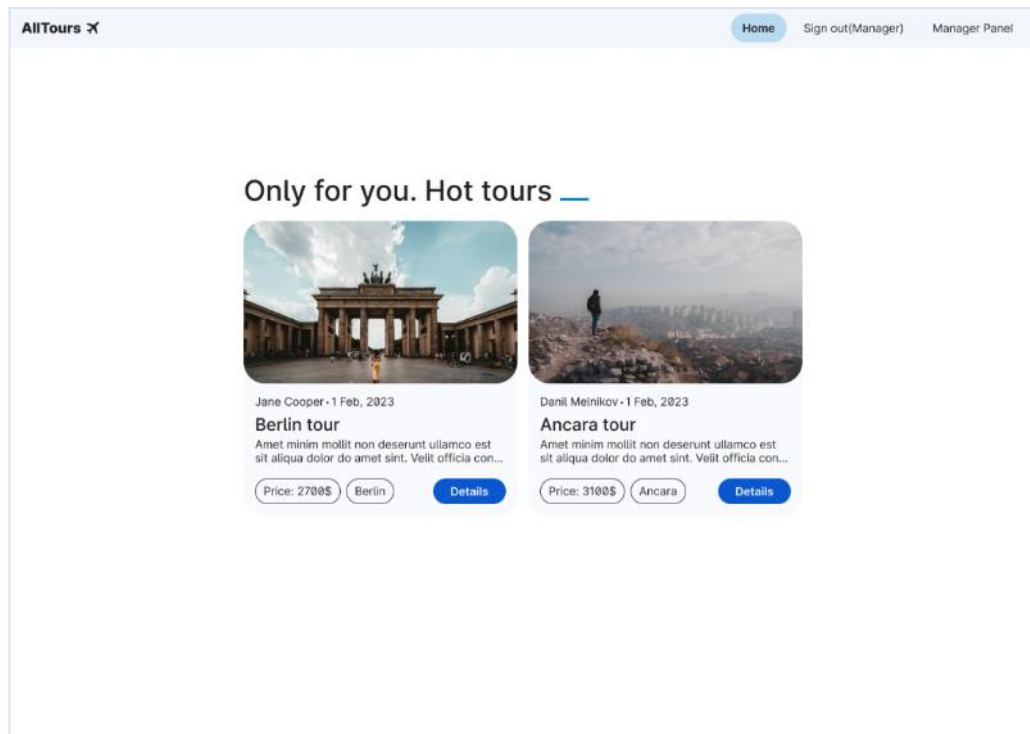


Рисунок 4.6 – Manager Panel Постачальника

4.4 Висновки за розділом 4

У цьому розділі було детально описано структуру та зміст контролерів. Кожений з класів так контролерів відповідає за певну частину функціональності системи. Це забезпечує чітку організацію коду та легкість управління різними аспектами програми.

Ключові компоненти включають контролери для аеропортів, країн, угод, готелів, а також управління користувацькими ролями та авторизацією. Окрема увага була приділена функціональності кожного контролера, з особливим акцентом на такі операції як перегляд, створення, оновлення та

видалення ресурсів. Це забезпечує високий рівень контролю та гнучкості при управлінні даними в системі.

Використання класів для управління правилами аутентифікації, елементами авторизації, взаємодією з соціальними мережами та управлінням профілями користувачів.

Окремо варто відзначити реалізацію графічного інтерфейсу користувача. Інтерфейс адаптований для різних ролей користувачів (наприклад, гості, зареєстровані користувачі, адміністратори, менеджери), що забезпечує інтуїтивно зрозуміле та ефективне взаємодія з системою. Це важливо для забезпечення сприятливого досвіду користувача і надає зручності у використанні системи.

5 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування

Тест-кейс – це професійна документація тестувальника, послідовність дій спрямована на перевірку будь-якого функціоналу, що описує як прийти до фактичного результату [33].

Тест-кейси про перевірку продукту без ознайомлення з усією документацією. Написаний один раз, зручний в підтримці тест-кейс заощадить багато часу і сил тестувальникам.

Тестування за допомогою тест-кейсів та чек-листів є важливим етапом розробки будь-якого програмного забезпечення, оскільки:

- воно дозволяє забезпечити відповідність програмного продукту вимогам специфікації;
- допомагає ідентифікувати помилки та недоліки в коді перед релізом продукту;
- забезпечує більш високу якість програмного продукту і покращує користувацький досвід;
- допомагає уникнути додаткових витрат на підтримку та виправлення помилок після випуску продукту на ринок;
- сприяє покращенню надійності та безпеки програмного забезпечення.

Розглянемо модуль – тестування реєстрації, який зображено у табл. 5.1.

Таблиця 5.1 – Тестування реєстрації

Назва:	Реєстрація	
Функція:	Реєстрація нового користувача	
Дія	Очікуваний результат	Результат теста: ● пройдено
Передумова:		
Відкрийте сайт http://localhost/Travel_CRM/web/	Сайт відкрит и доступний	пройдено
Натисніть на кнопку “Register”	Сторінка "Registration" відкрита и доступна	пройдено
Кроки тестування:		
Заповнити форму реєстрації: "E-mail": test@test.com "Username": Test88 "Password": test99	Дані успішно введені	пройдено
Натисніть кнопку "Sign up"	Сторінка "Confirm email" відкрита	пройдено
Постумова:		
Натисніть по кнопці "Home"	Сторінка "Home" відкрита	пройдено

Як видно з даних табл. 5.1, процес реєстрації користувача пройшов успішно на всіх етапах, включаючи введення даних та підтвердження електронної пошти. Це свідчить про те, що модуль реєстрації функціонує коректно і є надійним.

Розглянемо модуль – авторизація користувача, який зображено у табл. 5.2.

Таблиця 5.2 – Авторизація користувача

Назва:	Логін	
Функція:	Авторизація користувача	
Дія	Очікуваний результат	Результат тесту: ● пройдено
Передумова:		
Відкрийте сайт http://localhost/Travel_CRM/web/	Сайт відкритий и доступний	пройдено
Дія	Очікуваний результат	Результат тесту: ● пройдено
Передумова:		
Натисніть на кнопку "Sign in"	Сторінка "Sign in" відкрита і доступна	пройдено
Кроки тестування:		
Заповните форму авторизації: "Username": Test88 "Password": test99	Дані успішно введені	пройдено
Натисніть кнопку "Sign in"	Сторінка "Home" відкрита	пройдено

Як видно з даних табл. 5.2, процес авторизації користувача також пройшов без помилок, що демонструє правильну роботу системи аутентифікації.

Розглянемо модуль – тестування створення заявки, який зображено у табл. 5.3.

Таблиця 5.3 – Тестування створення заявки

Назва:	Створення заявки	
Функція:	Створення заявки	
Дія	Очікуваний результат	Результат тесту: ● пройдено
Передумова:		
Відкрийте сайт http://localhost/Travel_CRM/web/	Сайт відкритий і доступний	пройдено
Нажміть на кнопку “Details”	Сторінка "Details" відкрита і доступна	пройдено
Нажміть на кнопку “Apply”	Сторінка "Create Deal Request" відкрита і доступна	пройдено
Кроки тестування:		
Заповнити форму заявки: "Name": Melnikov Danylo Andriyovich	Дані успішно введені	пройдено
"Phone": 38012312313 "OfferID": Stambul		
Натисніть кнопку "Create"	Сторінка "/*Name*/" відкрита	пройдено
Постумова:		
Натисніть по кнопці "Home"	Сторінка "Home" відкрита	пройдено

Як видно з даних табл. 5.3, створення заявки на угоду було успішно завершено, і користувач мав можливість повернутися на головну сторінку, що свідчить про ефективність функціоналу створення заяв.

Розглянемо модуль – тестування редагування заяви, який зображено у табл. 5.4.

Таблиця 5.4 – Тестування редагування заяви

Назва:	Редагування заяви		
Функція:	Редагування заяви		
Дія		Очікуваний результат	Результат тесту: ● пройдено
Передумова:			
Відкрийте сайт http://localhost/Travel_CRM/web/		Сайт відкритий і доступний	пройдено
Натисніть на кнопку “Request Panel”		Сторінка "Request Panel" відкрита і доступна	пройдено
Натисніть на кнопку “Update”		Сторінка "Update Deal Request" відкрита і доступна	пройдено
Кроки тестування:			
Заповніть форму заяви: "Name": Ivanov Ivan Petrovich "Phone": 099228322 "OfferID": Stambul		Дані успішно введені	пройдено
Натисніть кнопку "Update"		Сторінка "/*Name*/" відкрита, дані змінились	пройдено
Постумова:			
Натисніть по кнопці "Home"		Сторінка "Home" відкрита	пройдено

У цьому модулі користувач мав можливість редагувати існуючу заявку. Всі кроки пройшли успішно - сайт було доступний, сторінка для редагування була доступна і данні були оновлені після натискання кнопки "Update".

Розглянемо модуль – тестування видалення заяви, який зображено у табл. 5.5.

Таблиця 5.5 – Тестування видалення заяви

Назва:	Видалення заяви		
Функція:	Видалення заяви		
Дія		Очікуваний результат	Результат тесту: ● пройдено
Передумова:			
Відкрийте сайт http://localhost/Travel_CRM/web/		Сайт відкритий і доступний	пройдено
Натисніть на кнопку “Request Panel”		Сторінка "Request Panel" відкрита і доступна	пройдено
Кроки тестування:			
Натисніть на кнопку “Delete”		Запит на підтвердження дії	пройдено
Натисніть кнопку "Ok"		Заява видалена	пройдено
Постумова:			
Натисніть по кнопці "Home"		Сторінка "Home" відкрита	пройдено

В цьому модулі користувач успішно видалив заявку. Сайт було доступний, панель управління запитами була доступна і після підтвердження дії натисканням кнопки "Ok" заява була видалена.

5.2 Чек-ліст функціоналу WEB-додатку

Чек-ліст (Checklist - контрольний список) - список, що містить ряд необхідних перевірок для будь-якої роботи. Відзначаючи пункти списку,

співробітник може дізнатися про стан, коректності виконання цієї роботи [34].

На рисунку 5.1 представлений чек-лист верстки, який включає ряд перевірок, спрямованих на забезпечення якості і точності відображення веб-сторінок у різних браузерях та пристроях.

Розглянемо чек-лист «верстки», який зображено на рис. 5.1.

	IE8	FF	GChrome
Головна сторінка			
Коректе відображення сторінки	Passed	Passed	Passed
Header	Passed	Passed	Passed
Footer	Passed	Passed	Passed
Коректе відображення Fonts	Passed	Passed	Passed
Перевірка граматики контенту	Passed	Passed	Passed
Відображення кольорової гамми, contrast-cheker	Passed	Passed	Passed
Коректне відображення кнопок, СТА	Passed	Passed	Passed
Коректність скрола	Passed	Passed	Passed
Коректна робота реєстрації	Passed	Passed	Passed
Коректна відображення вказівника	Passed	Passed	Passed
Коректна робота функціоналу	Passed	Passed	Passed
Коректне відображення блоків	Passed	Passed	Passed

Рисунок 5.1 – Чек-лист верстки

Як видно з даних рис. 5.1, основні пункти, такі як, к сумісність з основними браузерами, адаптивність дизайну до різних розмірів екрану, коректність відображення шрифтів та кольорової палітри, перевірку на наявність зламаних блоків або неправильно вирівняних елементів пройшли успішно.

Рисунок 5.2 ілюструє чек-лист, призначений для перевірки сумісності веб-додатку з різними пристроями. Це включає тести на працездатність на мобільних телефонах, планшетах, настільних комп'ютерах з різними операційними системами.

Розглянемо чек-лист «використаних пристроїв», який зображено на рис. 5.2.

	iPhone	iPhone	iPhone	iPad	iPad	Samsung	Sony	Sony	HTC windows
OS versions	iOS 5.1.1	iOS 6.0.2	iOS 7.0.4	iOS 4.3.3	iOS 7.0.2	Android	Android	Android	Windows Phone 8.0
Головна сторінка									
Коректне відображення сторінок	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректне відображення Fonts	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Відображення кольорової гамми, contrast-checker	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректне відображення кнопок, CTA	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність скрола	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність тапов	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність свайпів	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність відображення pop-up's	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Швидкодія	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність валідації	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed
Коректність переходів	Skipped	Skipped	Skipped	Skipped	Skipped	Passed	Passed	Passed	Passed

Рисунок 5.2 – Чек-лист пристроїв

Як видно з даних рис. 5.2, основні пункти, такі як, адаптивність дизайну до різних розмірів екрану, коректність відображення шрифтів та кольорової палітри, тапів, свайпів, валідації, перевірку на наявність зламаних блоків або неправильно вирівняних елементів пройшли успішно.

На рисунку 5.3 представлено чек-лист, який зосереджений на функціональному аспекті веб-додатку.

Розглянемо чек-лист «функціоналу», який зображено на рис. 5.3.

Browser	IE8	IE9	IE10	FF	GChrome	Safari	Opera
Реєстрація та профіль							
Реєстрація та профіль	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Редагування профіля	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Функціонал адміністратора							
Робота з замовленнями	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Робота з готелями	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Робота з містами	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Робота з країнами	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Робота з вимогами	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Управління користувачами	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Функціонал користувача							
Створення замовлення	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Перегляд замовлення	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Редагування замовлення	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Видалення замовлення	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped
Робота з замовленнями	Passed	Passed	Passed	Passed	Passed	Skipped	Skipped

Рисунок 5.3 – Чек-лист функціоналу

Як видно з даних рис. 5.3, основні пункти, такі як, перевірка функціонування основних функцій, а також тестування користувацьких сценаріїв для забезпечення безперебійної роботи з точки зору кінцевого користувача.

5.3 Висновок до розділу 5

Чек-лісти є ключовим інструментом для забезпечення високого стандарту якості при розробці веб-додатків. Використання чек-листа верстки допомагає гарантувати, що дизайн є консистентним і правильно виконаним на всіх платформах. Чек-лист пристроїв забезпечує адаптацію додатку до широкого спектра пристроїв і середовищ, тоді як чек-лист функціоналу гарантує, що всі основні функції додатку працюють бездоганно. Разом це сприяє створенню надійного і користувацького веб-досвіду.

6 КЕРІВНИЦТВО ПРОГРАМІСТА

6.1 Призначення й умови використання

Мета проекту – створення інформаційної системи, яка б дозволила автоматизувати робочі процеси туристичного агентства, включаючи управління турами, обслуговування клієнтів та процес бронювання. Специфікація завдань повинна включати розробку користувацького інтерфейсу, серверної частини, інтеграцію з зовнішніми сервісами (наприклад, системами онлайн-бронювання), а також забезпечення безпеки даних.

Для коректної роботи ПП необхідні такі мінімальні системні вимоги:

- операційна система Microsoft Windows 10/8/7 (рекомендується встановити всі останні оновлення);
- мережевий адаптер 100 Мбіт / с;
- жорсткий диск не менше 500;
- процесор з мінімальною тактовою частотою 2,0 ГГц. Рекомендується використовувати як мінімум двоядерний процесор;
- оперативна пам'ять від 2 ГБ;

6.2 Архітектура

Система розробляється на основі модульної архітектури, що включає в себе окремі компоненти для управління турами, клієнтами, бронюваннями та аналітикою. Використовується клієнт-серверний підхід з веб-базованим інтерфейсом для зручності користувачів.

Модульна структура:

- проєкт розділений на модулі: Frontend (клієнтська частина), Backend (серверна частина), Database (база даних).

Взаємозв'язки:

– всі модулі з'єднуються через чітко визначені API контракти. Frontend спілкується з Backend через RESTful API, який, у свою чергу, взаємодіє з базою даних для збереження та отримання даних.

Архітектурні рішення:

– використання мікросервісної архітектури для забезпечення гнучкості та масштабованості системи. Контейнеризація за допомогою Docker для легкості розгортання та управління середовищами. Використання CI/CD пайплайнів для автоматизації тестування та розгортання.

6.3 Структура коду

Опис основних контролерів та методів:

- DealController – керування пропозиціями агентства;
- SiteController – загальне керування сайтом;
- RequirementController – управління вимогами та критеріями;

Методи контролерів:

- actionIndex – виводить список ресурсів;
- actionView – відображає один конкретний ресурс;
- actionCreate – створює новий ресурс;
- actionUpdate – оновлює існуючий ресурс;
- actionDelete – видаляє ресурс.

Загальні принципи організації коду:

- використані встановлені паттерни проєктування, де це доречно, для покращення читабельності коду та його можливостей для масштабування: MVC (Model-View-Controller) для Backend і компонентний підхід для Frontend.;
- правила іменування: camelCase для змінних та функцій, PascalCase для класів.

- чистота коду: весь код написаний чітко та зрозуміло, з дотриманням принципів чистого кодування. Це означає, що кожен клас, функція чи модуль має виконувати одну конкретну задачу і бути легким для розуміння та підтримки.

- файли та каталоги організовані логічно. Компоненти Frontend можуть бути розділені за функціональним призначенням, а код Backend - за моделями, контролерами та сервісами.

- обробка помилок: код вміщує ефективну обробку помилок, що забезпечить легке виявлення та усунення проблем. Використовуйте промиси або `async/await` для управління асинхронними операціями.

- тестування: розробка повинна супроводжуватися написанням модульних та інтеграційних тестів. Це допомагає забезпечити надійність коду та спрощує процес рефакторингу.

- версійний контроль: Використовуйте системи версійного контролю, такі як Git, для ведення історії змін, сприяючи колаборативному розробленню та ефективному управлінню кодом.

6.4 Використані технології

Технології, використані в документі:

- PHP;
- JavaScript;
- MySQL;
- фреймворк Yii2;
- методологія розробки Transfer;
- патерн Model-View-Controller (MVC).

Ці технології використовуються для розробки програмної системи для туристичного агентства. Система автоматизує та оптимізує ключові бізнес-процеси, такі як бронювання турів, управління відносинами з клієнтами та

відстеження запасів. Система розроблена з використанням фреймворку Yii2 та методології розробки Transfer. Патерн MVC використовується для розділення презентації, логіки та даних системи.

Система призначена для зручного користування та легкої взаємодії. Вона надає різноманітні функції, що дозволяють користувачам бронювати тури, управляти відносинами з клієнтами та відстежувати запаси. Система також є безпечною та надійною. Вона регулярно резервується та захищена від несанкціонованого доступу.

Система була розроблена з використанням різноманітних методів тестування. Проведено модульні, інтеграційні та системні тести, щоб переконатися, що система працює так, як очікується.

6.5 Інструкція зі збирання та встановлення

Команди для запуску сервера:

- *php yii serve.*

Ця команда ініціює вбудований веб-сервер PHP і автоматично налаштовує його для роботи з додатком.

Команди для входу в локал:

- *php yii serve --hostname=localhost --port=8080.*

Бібліотеки, які пов'язують код з базою:

- **Yii ORM** – вбудований ORM-інструмент Yii, який дозволяє взаємодіяти з базою даних за допомогою об'єктно-орієнтованого підходу;

- **Yii Query Builder** – використовується для складання SQL-запитів в об'єктно-орієнтованому стилі.

Опис основних блоків коду:

- а) приклад методу для створення готелю:

```
// HotelController.php

public function createAction()
{
    $model = new Hotel();

    if (Yii::$app->user->isGuest) {
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}
```

б) Приклад використання RESTful:

```
// OfferController.php

class OfferController extends Controller
{
    // ...

    public function actionIndex()
    {
        $searchModel = new OfferSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
}
```

6.6 Документація з API

Документація з API доступна за шляхом /api. Для взаємодії з API використовуйте стандартні HTTP-запити, такі як GET, POST, PUT та DELETE.

6.7 Ієрархія папок та шляхи до основних файлів

Папки проєкту:

– /client – містить всю клієнтську частину проєкту;

- /server – містить серверний код, включаючи API ендпойнти та логіку бізнес-процесів;
- /public – використовується для статичних файлів, таких як зображення, CSS та JavaScript для клієнтської частини;
- /models – містить схеми бази даних;
- /config – містить конфігураційні файли для налаштування фреймворку та інших компонентів.

7 КЕРІВНИЦТВО КОРИСТУВАЧА

7.1 Загальна інформація

Посібник користувача є документом, який призначений для користувача розробленої програми. Його мета – забезпечити користувачами зрозумілі та чіпкі інструкції щодо встановлення, налаштування та використання програми. Воно відіграє ключову роль у забезпеченні ефективної взаємодії користувачів із програмою, підвищенні їх задоволеності та зниженні рівня можливих помилок.

Мета посібника користувача:

- надання інформації. Посібник повинен надавати докладні відомості про функціональність програми, включаючи основні можливості, операції та параметри;

- навчання. Він повинен служити засобом навчання користувачам, дозволяючи їм швидко освоїти програму та використовувати її найефективнішим чином;

- вирішення проблем. Посібник повинен містити секції з розв’язаннями частих проблем, а також інструкції з усунення несправностей;

- поліпшення досвіду користувача. Посібник допомагає створити позитивний досвід користувача, надаючи інформацію, необхідну для впевненої та успішної взаємодії з програмою.

Пункти, які зазвичай включаються до посібника користувача:

- введення – короткий опис програми та її цільової аудиторії;

- встановлення – детальні інструкції щодо встановлення програми, включаючи системні вимоги;

- налаштування – пояснення щодо налаштування параметрів програми відповідно до потреб користувача;

- використання — опис основних функцій програми та кроків для їх використання;
- довідник з інтерфейсу — пояснення до елементів інтерфейсу програми;
- приклади використання — конкретні сценарії використання програми для кращого розуміння її функціональності;
- вирішення проблем. Секція з найчастішими запитаннями (FAQ) та інструкціями щодо усунення несправностей;
- технічна підтримка — інформація про те, як отримати додаткову допомогу та підтримку;
- оновлення та нововведення — інформація про майбутні оновлення, нові функції та зміни;
- завершення. Подякуйте користувачу за використання програми та надайте контактну інформацію для зворотного зв'язку.

Створення якісного посібника користувача допомагає покращити сприйняття програми, зменшити навантаження на технічну підтримку та забезпечити задоволення користувачів.

7.2 Область застосування

Сайт туристичного агенства (далі — Сайт) призначений для офіційної публікації такої інформації:

- публікації готових туристичних рішень;
- публікації компаній займаються трансферними послугами;
- публікації компаній займаються готельними послугами;
- інформацію зворотного зв'язку з агентством для подальшої співпраці з постачальниками турів, трансферу, готелів;
- інформацію про місцезнаходження наданих готелів;

- інформації про результати успішності укладення угод;
- статистична інформація про діяльність агентства;
- інформації про кадрове забезпечення.

7.3 Рівень підготовки користувача

Користувач Сайту повинен володіти такою класифікацією.

Призначені для користувача навички в роботі з ПК.

Призначені для користувача навички роботи мінімум з web-браузером Microsoft Internet Explorer.

Знання предметної області і знайомство з Керівництвом користувача.

7.4 Призначення та умови застосування

Види діяльності, функції, для автоматизації яких призначений Сайт.

Сайт автоматизує функції розміщення інформації про готові туристичні рішення, так само формування попереднього документа за коштами якого клієнт може безтурботно надати документ менеджеру для швидко його оформлення.

7.5 Вимоги до програмного забезпечення

Для коректної роботи сайту слід використовувати браузери, що забезпечують повну сумісність зі стандартами: HTML 5.0, CSS 3.0, JavaScript, PHP 5.3; Рекомендоване програмне забезпечення див. у табл. 7.1.

Таблиця 7.1 – Порівняльна характеристика обраних середовищ розробки.

Вид ПЗ	Програмний продукт	Версія
ОС	Microsoft Windows	Windows 7/Windows 10, 11, Unix
Web-браузер	Microsoft Edge, Google Chrome, Mozilla Firefox, Opera	Мінімальна версія

Розглянемо наступний пункт підготовки до роботи з програмним продуктом.

7.6 Підготовка до роботи

Запуск системи.

Доступ до Сайту здійснюється інтерактивно через мережу Інтернет за допомогою звичайного web-браузера. Адреса сайту в мережі Інтернет: <http://alltour.com>. Для коректної роботи клієнтської частини повинен використовуватися браузер Microsoft Edge, Google Chrome, Opera, Mozilla Firefox мінімальної версії.

При вході на Сайт користувач потрапляє на Головну сторінку в розділ «Home».

На Сайті розрізняються наступні групи користувачів:

Анонімний користувач (не увійшли на Сайті, далі Гість).

Користувач - приватна особа, авторизований на сайті (далі Користувач), що має можливість редагування інформації на сайті.

Адміністратор сайту (далі Адміністратор) відповідає за групи користувачів і за розмежування їх прав

Менеджер сайту (далі Менеджер) відповідає за зміст туру, формування попереднього туру документа, формування туру з переваг клієнта в розділі заявок.

Постачальник турів, трансферів, готелів (далі Постачальник) відповідає за актуальною наданою інформації з такими ж правами як і менеджер.

В даному документі розглянуті функції Сайту, доступні користувачам перших 5-х груп.

7.7 Перевірка працездатності системи

Якщо при спробі доступу до Сайту з клієнтського робочого місця за допомогою веб-браузера не виникає повідомлень про помилки, то система працює нормально.

7.8 Склад та структура сайту

Головна сторінка сайту

Головна сторінка сайту візуально розділена на дві області:

У верхній частині екрану розташований заголовок, в якому всі існуючі відображення об'єктів панелі використання (зліва направо):

Авторизація, Додому, Реєстрація, Менеджер панель, Адмін панель, панель заявок.

Група посилань Розділів є меню (упорядкований список посилань), призначене для навігації по сайту, (рис. 7.1).

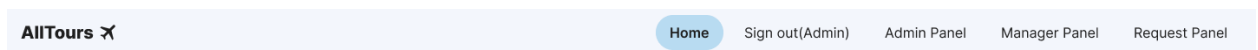


Рисунок 7.1 – Група посилань Розділи меню

Більш докладно використання кожного пункту меню розглядається в відповідних розділах керівництва.

При знаходженні на сайті без авторизації Гість бачимо основну сторінку переліком турів, (рис. 7.2).

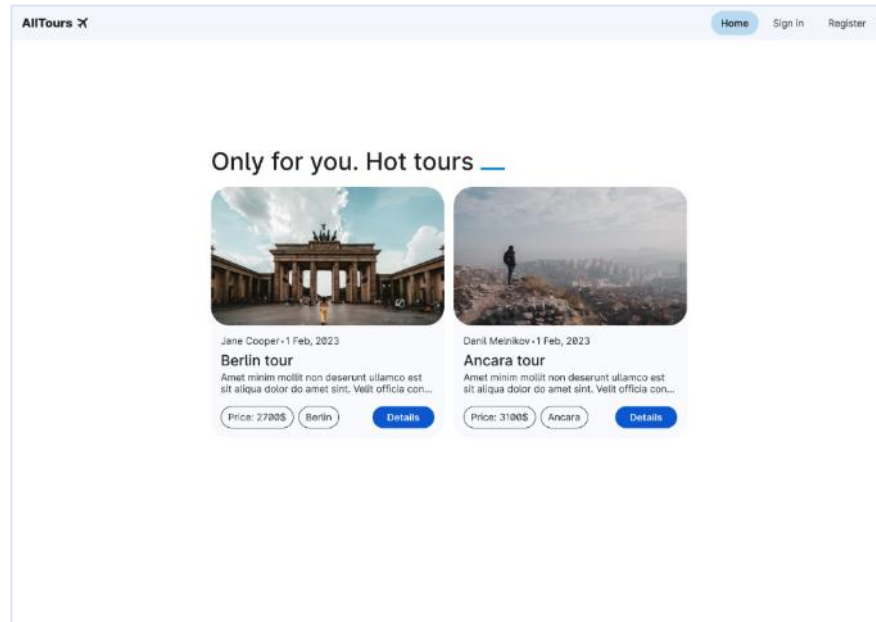


Рисунок 7.2 – Знаходження Гість на сайті

При авторизації за Користувача з'являються додаткова панель, а саме панель персональних запитів клієнта, (рис. 7.3).

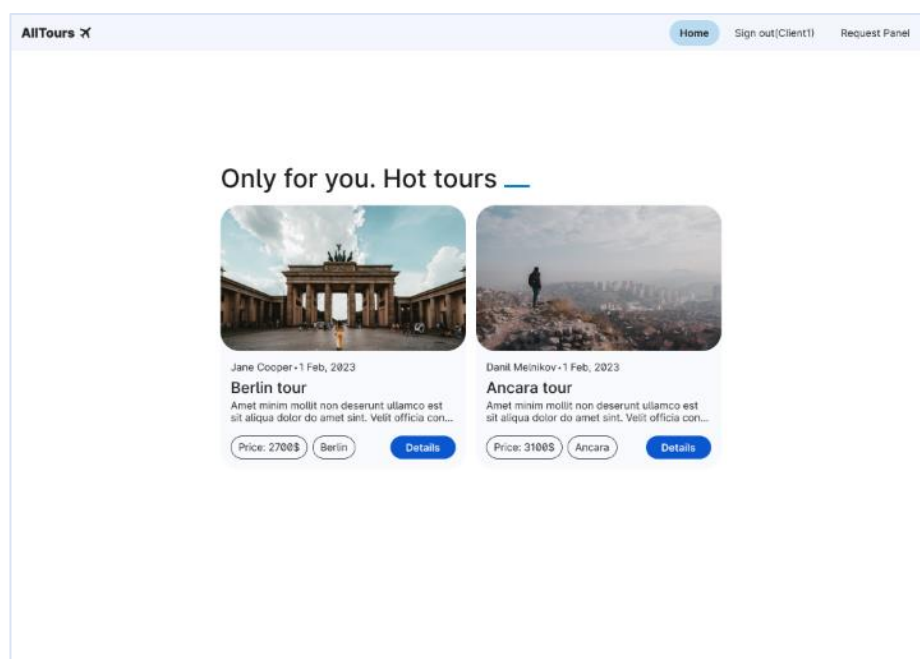


Рисунок 7.3 – Request Panel Користувача

При авторизації за Адміністратор з'являються додаткові панелі, а саме панель адміністратор, панель менеджера і панель запитів, (рис. 7.4).

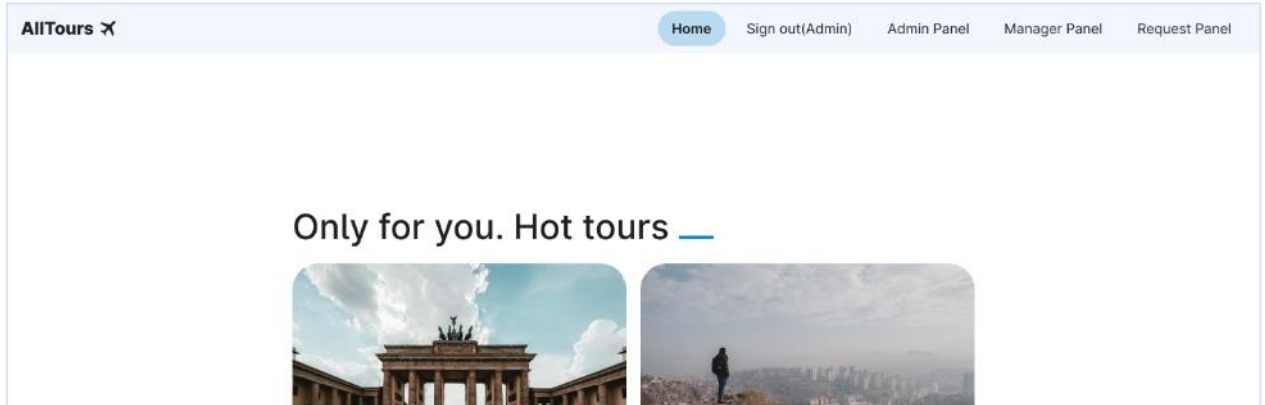


Рисунок 7.4 – Admin Panel, Manager Panel, Request Panel Адміністратора

При авторизації за Менеджера з'являються додаткова панель, а саме панель менеджера для керування турами, (рис. 7.5).

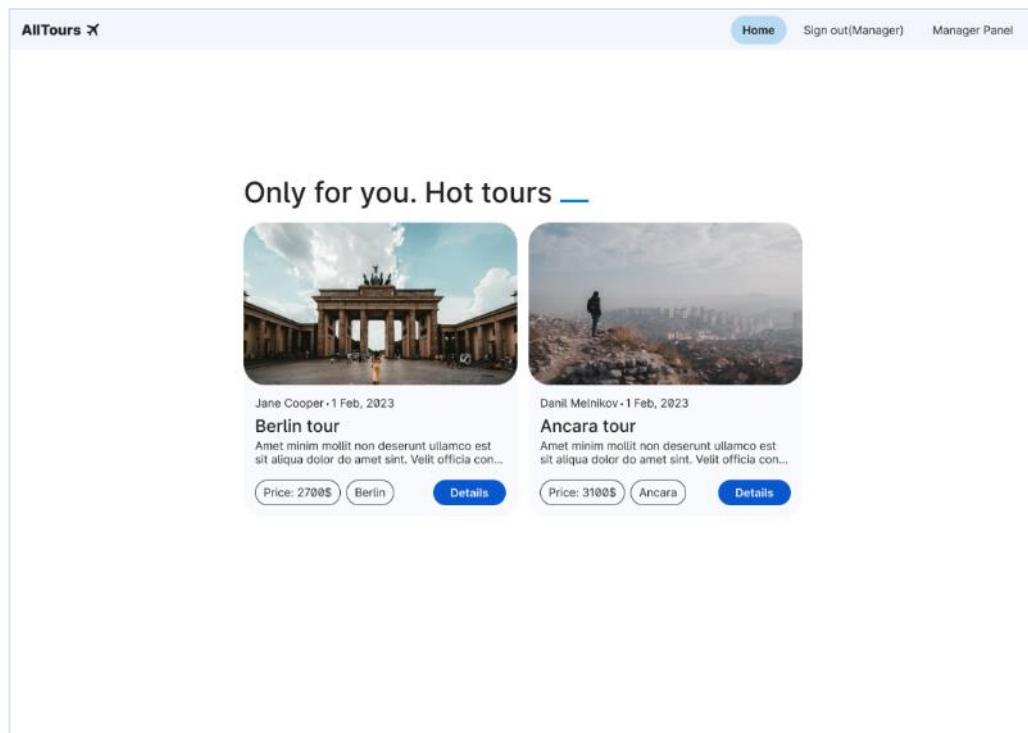


Рисунок 7.5 – Manager Panel Менеджера

При авторизації за Менеджера-поставщика з'являються додаткова панель, а саме панель менеджера-поставщика для заповнення актуальних партнер даних, (рис. 7.6).

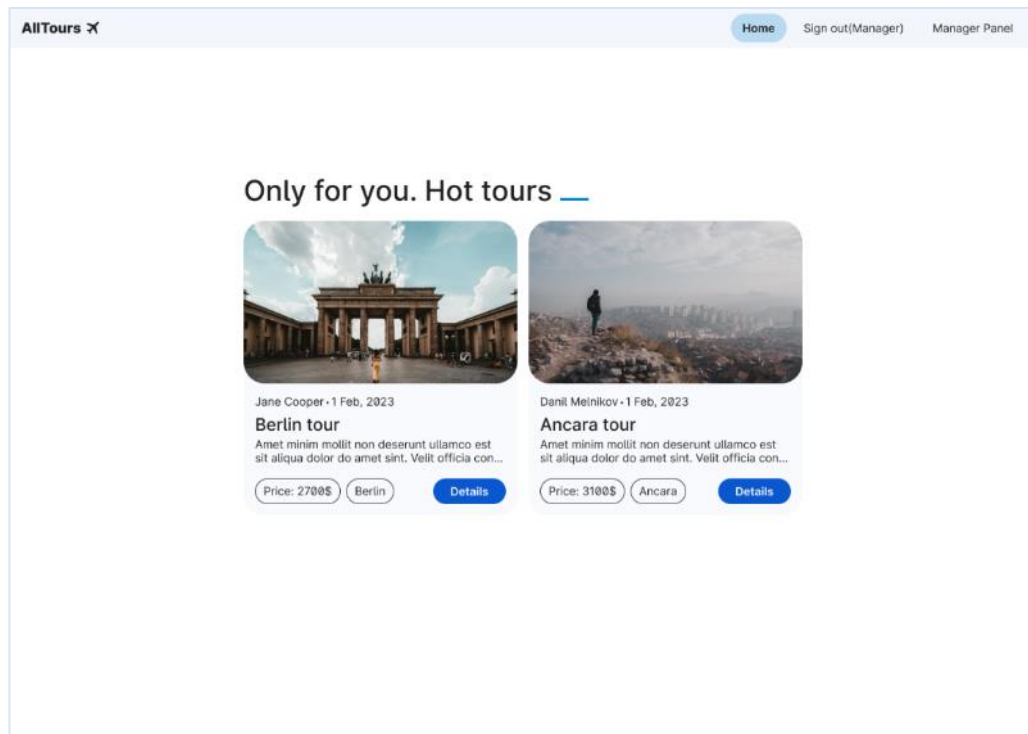


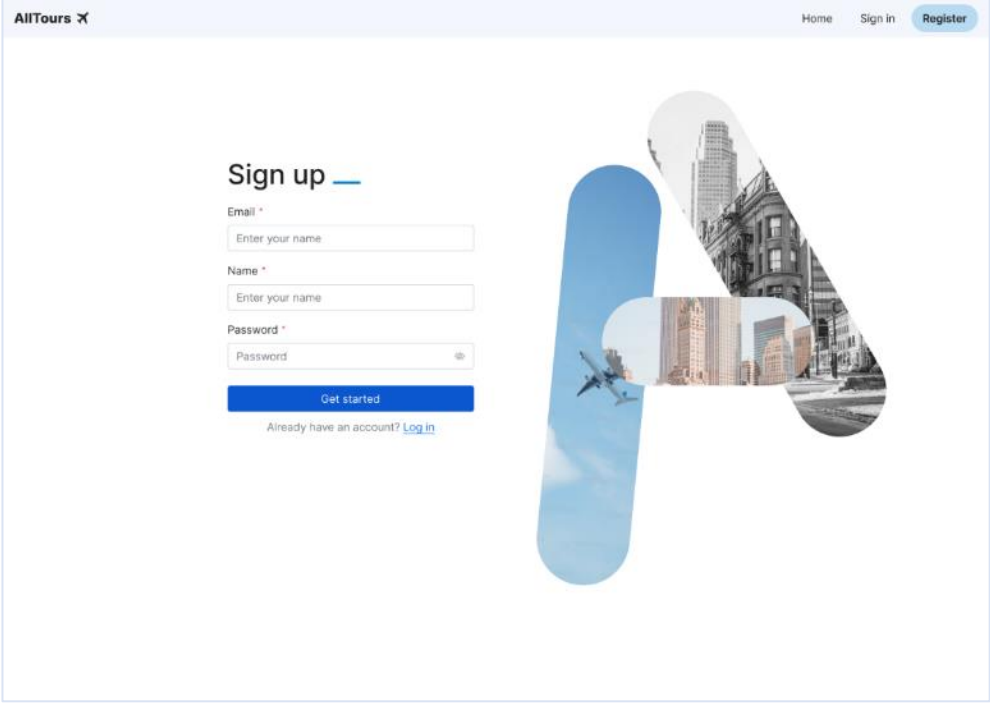
Рисунок 7.6 – Manager Panel Постачальника

7.9 Реєстрація на сайті

Не рекомендується використовувати прості паролі, наприклад, дати народження, імена і т.д. Рекомендується, щоб в паролі присутні цифри, рядкові і заголовні букви латинського алфавіту і символи з набору! @ # \$ % ^ & * () _ + = - \ | / > ? < . , . Довжина пароля повинна бути не менше 8 символів.

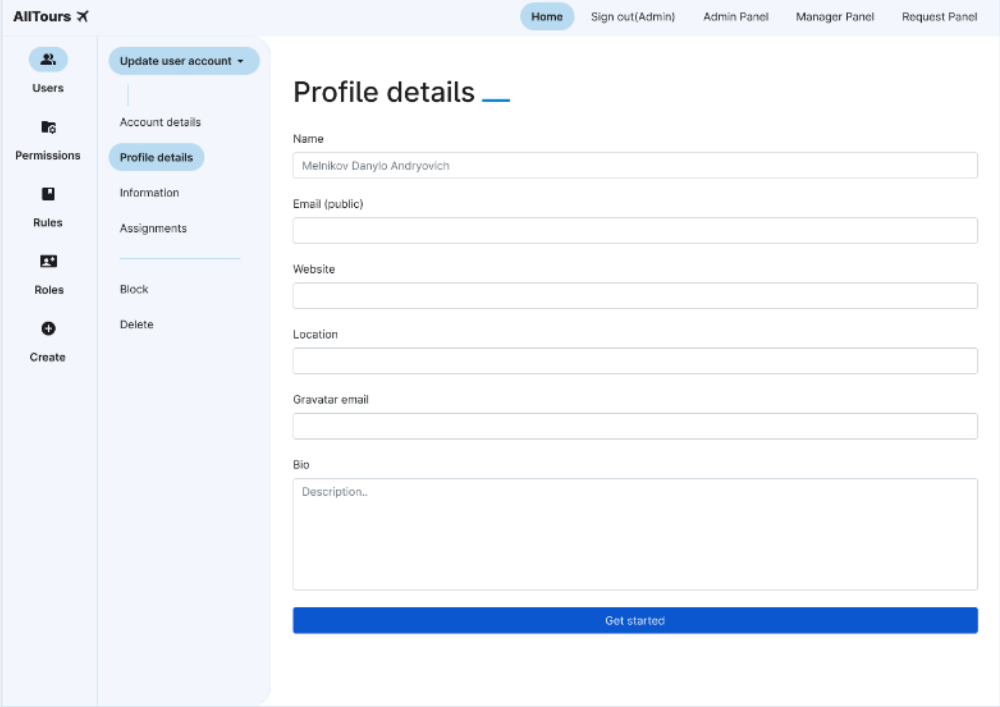
Користувач повинен заповнити форму для реєстрації (розташовану на головній сторінці сайту) з перенаправленням на розділ Реєстрація. Форма включає в себе наступні дані: логін, пароль, email.

Користувачі реєструються на Сайті при відповідному виборі розділу реєстрації, (рис. 7.7-7.8).



The screenshot shows the 'Sign up' page of the AllTours website. The header includes the 'AllTours' logo and navigation links for 'Home', 'Sign in', and 'Register'. The main content area features a 'Sign up' heading followed by three input fields: 'Email *', 'Name *', and 'Password *'. Each field has a placeholder text 'Enter your name' or 'Password'. Below these fields is a blue 'Get started' button. A link for 'Already have an account? Log in' is positioned below the button. To the right of the form is a large, stylized letter 'A' graphic. The left vertical stroke of the 'A' is a solid blue bar, while the right vertical stroke and the horizontal crossbar are filled with a collage of travel-related images, including a city skyline, a street scene, and an airplane in flight.

Рисунок 7.7 – Реєстрація на Сайті



The screenshot displays the 'Profile details' page for a user on the AllTours website. The header shows the 'AllTours' logo and navigation links: 'Home', 'Sign out(Admin)', 'Admin Panel', 'Manager Panel', and 'Request Panel'. On the left, a sidebar contains a menu with icons and labels: 'Users', 'Permissions', 'Rules', 'Roles', and 'Create'. The main content area is titled 'Profile details' and contains several input fields: 'Name' (pre-filled with 'Meinikov Danylo Andriyovich'), 'Email (public)', 'Website', 'Location', 'Gravatar email', and 'Bio' (with a placeholder 'Description...'). A blue 'Get started' button is located at the bottom of the form. The sidebar also includes a section titled 'Update user account' with a dropdown arrow, and a list of options: 'Account details', 'Profile details' (which is highlighted), 'Information', 'Assignments', 'Block', and 'Delete'.

Рисунок 7.8 – Заповнення профілю

7.10 Авторизація та розділи сайту

Авторизація користувача за логіном і паролем.

Діалоговий блок для вказівки параметрів входу в систему, (рис. 7.9).

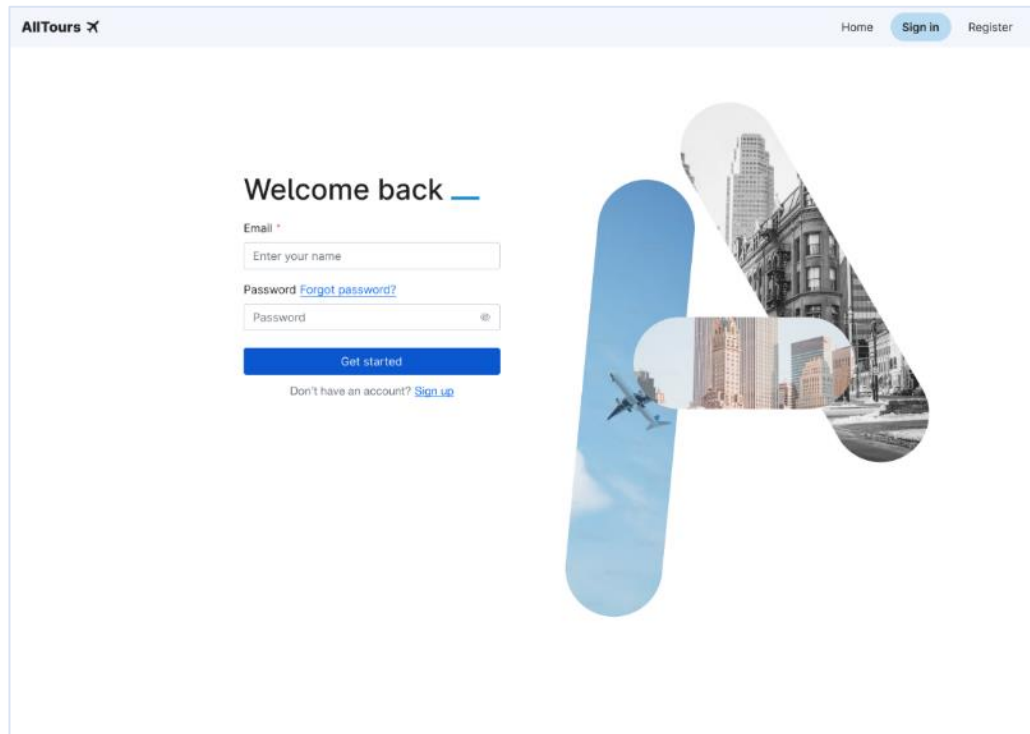


Рисунок 7.9 – Авторизація на сайті

Для авторизації досить ввести реєстраційні дані: логін і пароль, після чого натиснути на кнопку Увійти. При введенні пароля вводяться символи відображаються у вигляді «зірочок».

При відповідних настройках браузера значення параметрів, зазначених в даному діалоговому вікні, можуть запам'ятовуватися на клієнтській машині, і використовуватися при наступному вході в систему.

Розділ Авторизація використовується для входу на Сайт зареєстрованих користувачів.

Функції Сайту, доступні користувачам уповноваженими правами адміністратора.

Після того, як користувач авторизований на Сайті, він отримує відповідні права доступу прикріплені до його групі для виконання операцій на Сайті.

У разі якщо користувач забув пароль, він повинен скористатися посиланням «Forgot password?».

Система пропонує вказати email адреса, і в разі вірного email вишле новий автоматично згенерований пароль користувачу на вказаний при реєстрації основну адресу електронної пошти.

Після того, як прийде повідомлення електронної пошти з новим паролем, авторизація виконується звичайним чином, (рис. 7.10).

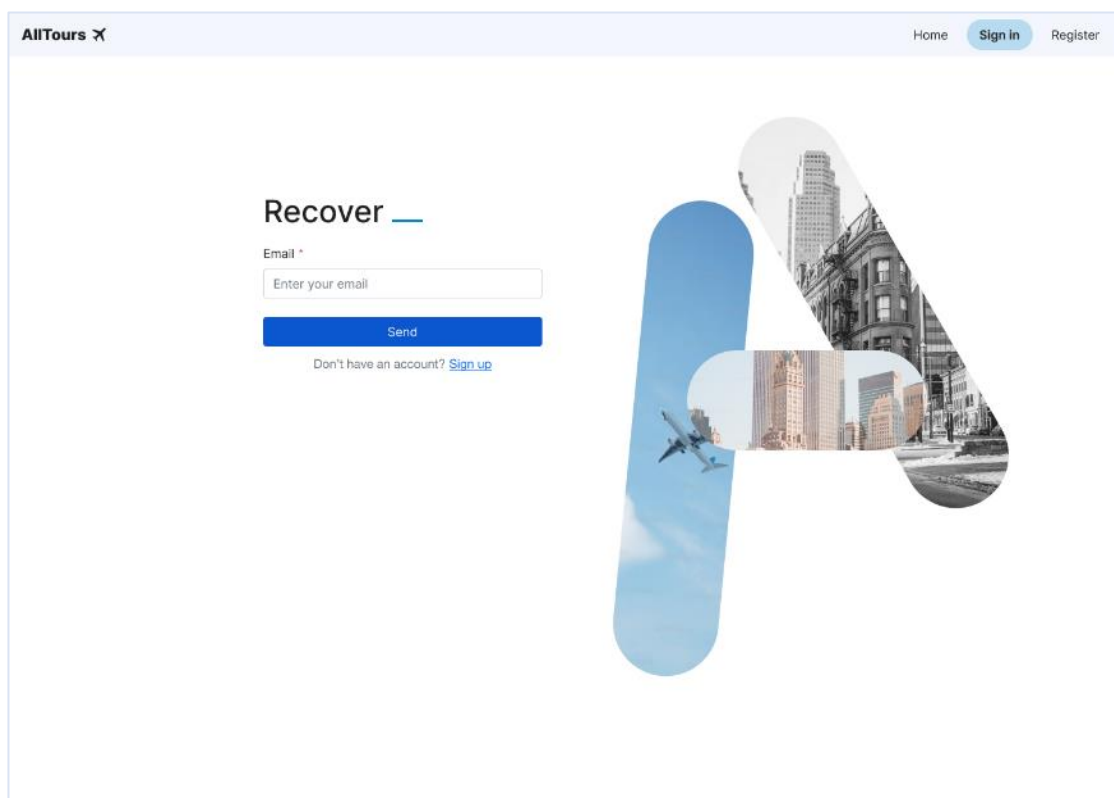


Рисунок 7.10 – Система забули пароль

7.11 Функціонал користувача

Центральна частина екрану містить основний зміст сторінки, на головній сторінці представлені короткі описи турів, (рис. 7.11-7.14).

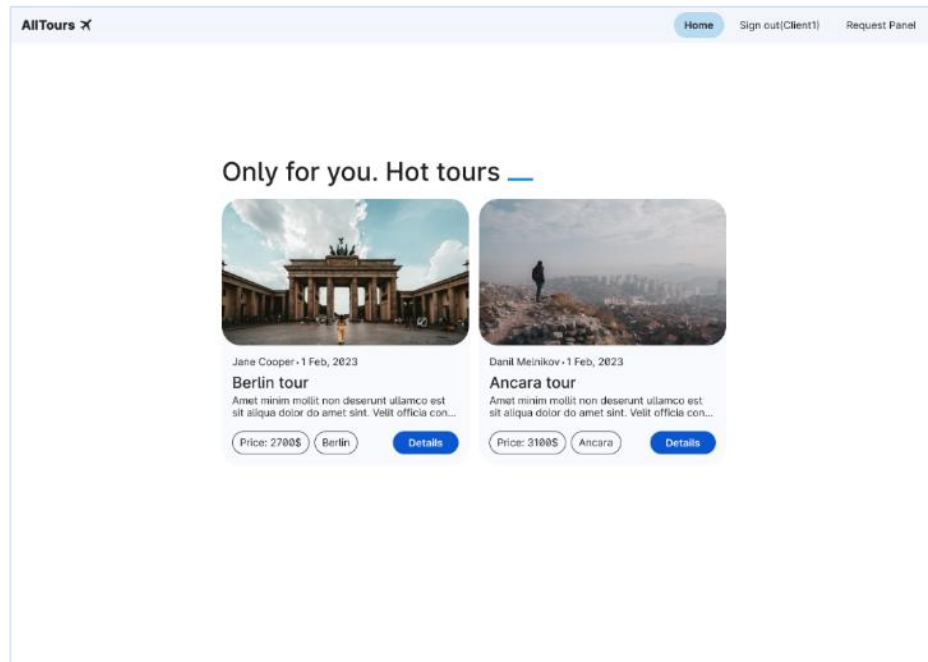


Рисунок 7.11 – Головна сторінка Користувача

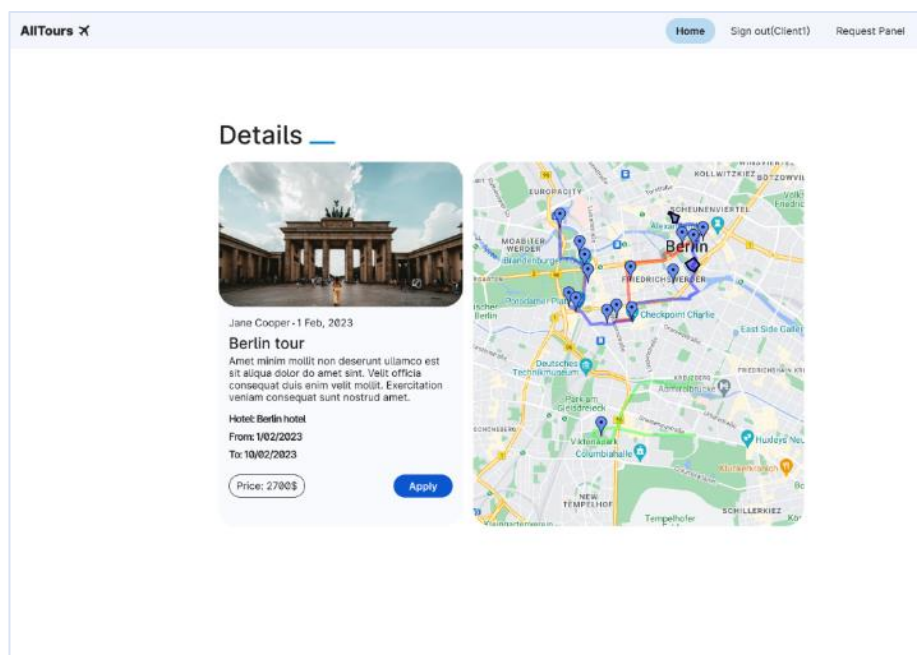


Рисунок 7.12 – Сторінка вибору тура з інформацією та геолокацією

AllTours

[Home](#) [Sign out\(Client1\)](#) [Request Panel](#)

Create Deal Request

Name

Meinikov Danylo Andriyovich

Phone

+38012312323

Offer ID

HOT! TOUR BERLIN

Create

Рисунок 7.13 – Сторінка для створення запиту туру

AllTours

[Home](#) [Sign out\(Client1\)](#) [Request Panel](#)

Melnikov Danylo Andriyovich

ID	6
Name	Meinikov Danylo Andriyovich
Phone	+38012312323
Offer ID	4
Created	2023-07-19 18:45:30
User ID	4

Update

Delete

Рисунок 7.14 – Сторінка номера заяви Клієнта

7.12 Функціонал адміністратора

Адміністратору надається можливість створення нового користувача, зміна дійсного електронного адреси, додавання ролей і правил, зміна імені облікового запису, блокування облікового запису, (рис. 7.15).

The screenshot displays the 'AllTours' Admin Dashboard. On the left, a sidebar menu includes 'Users', 'Permissions', 'Rules', and 'Roles', with a 'Create' button at the bottom. The main content area is titled 'Dashboard' and features a table of users. The table has columns for Username, Email, IP, Registration time, Confirmation, and Block status. The 'Block status' column is highlighted in red for all users, indicating they are blocked. Each row also includes icons for editing and deleting the user.

Username	Email	IP	Registration time	Confirmation	Block status
Admin	melnikov@gmail...	::1	2023-7-19 18:	Confirmed	Block
Client1	melnikov@gmail...	::1	2023-7-19 19:	Confirmed	Block
Client	melnikov@gmail...	::1	2023-7-19 19:	Confirmed	Block
Manager_P	melnikov@gmail...	::1	2023-7-19 20:	Confirmed	Block

Рисунок 7.15 – Функціонал панелі Адміністратора

7.13 Створення акаунта користувача

Адміністратор створює та заповнює акаунт користувача, (рис. 7.16 – 7.18).

The screenshot shows the 'Create' page for creating a new user account. The top navigation bar includes 'Home', 'Sign out(Admin)', 'Admin Panel', 'Manager Panel', and 'Request Panel'. The left sidebar has a 'Users' section with a 'Create' button. The main content area is titled 'Create' and includes a note: 'Credentials will be sent to the user by email. A password will be generated automatically if not provided.' The form contains fields for 'Email' (pre-filled with 'asd@gmail.com'), 'Name' (pre-filled with 'Danylo Melnikov'), and 'Password' (masked with dots). A blue 'Save' button is at the bottom.

Рисунок 7.16 – Створення акаунта користувача

The screenshot shows the 'Profile details' page for editing a user's profile. The top navigation bar is the same as in the previous image. The left sidebar has a 'Users' section with a 'Profile details' button. The main content area is titled 'Profile details' and includes fields for 'Name' (pre-filled with 'Danylo Melnikov'), 'Email (public)', 'Website', 'Location', and 'Gravatar email'. There is also a 'Bio' section with a text area containing placeholder text: 'Tomorrow never comes until it's too late -You could be sitting taking lunch -The news will hit you like a punch -It's only Tuesday -It's only Tuesday - You never thought we'd go to war -After all the things we saw -It's April Fools' day'. A blue 'Get started' button is at the bottom.

Рисунок 7.17 – Заповнення персональної інформації користувача

AllTours

HomeSign out(Admin)Admin PanelManager PanelRequest Panel

Users

New user

New role

Permissions

New permission

New rule

Rules

Roles

Create

Dashboard

Username	Email	IP	Registration time	Confirmation	Block status	
Admin	mehnikov@gmail...	1	2023-7-19 18:0	Confirmed	Block	
Client1	mehnikov@gmail...	1	2023-7-19 19:0	Confirmed	Block	
Client1	mehnikov@gmail...	1	2023-7-19 19:0	Confirmed	Block	
Client2	mehnikov@gmail...	1	2023-7-19 20:0	Confirmed	Block	
Manager	mehnikov@gmail.c...	1	2023-7-19 20:0	Confirmed	Block	
Manager1	mehnikov@gmail.com	1	2023-7-19 21:0	Confirmed	Block	
Test	mehnikov@gmail.c...	1	2023-7-19 22:0	Confirmed	Block	
test	msdag@gmail.com	1	2023-7-19 23:0	Confirmed	Block	
Manager_P	sdad@gmail.com	1	2023-7-19 23:0	Confirmed	Block	
Client31	mdafa@gmail.co...	1	2023-7-19 23:0	Confirmed	Block	
Danylo Mehnikov	asdgl@gmail.com	1	2023-8-19 11:0	Confirmed	Block	

Рисунок 7.18 – Перегляд користувачів системи

7.14 Створення ролей

Адміністратор створює ролі для користувачів системи, (рис. 7.19 – 7.20).

AllTours

HomeSign out(Admin)Admin PanelManager PanelRequest Panel

Users

Update role

Permissions

Rules

Roles

Create

Update role

Name

Manager

Description

Tomorrow never comes until it's too late -You could be sitting taking lunch -The news will hit you like a punch -It's only Tuesday -It's only Tuesday - You never thought we'd go to war -After all the things we saw -It's April Fools' day

Rule

Select rule

Children

Save

Рисунок 7.18 – Створення нової ролі користувача

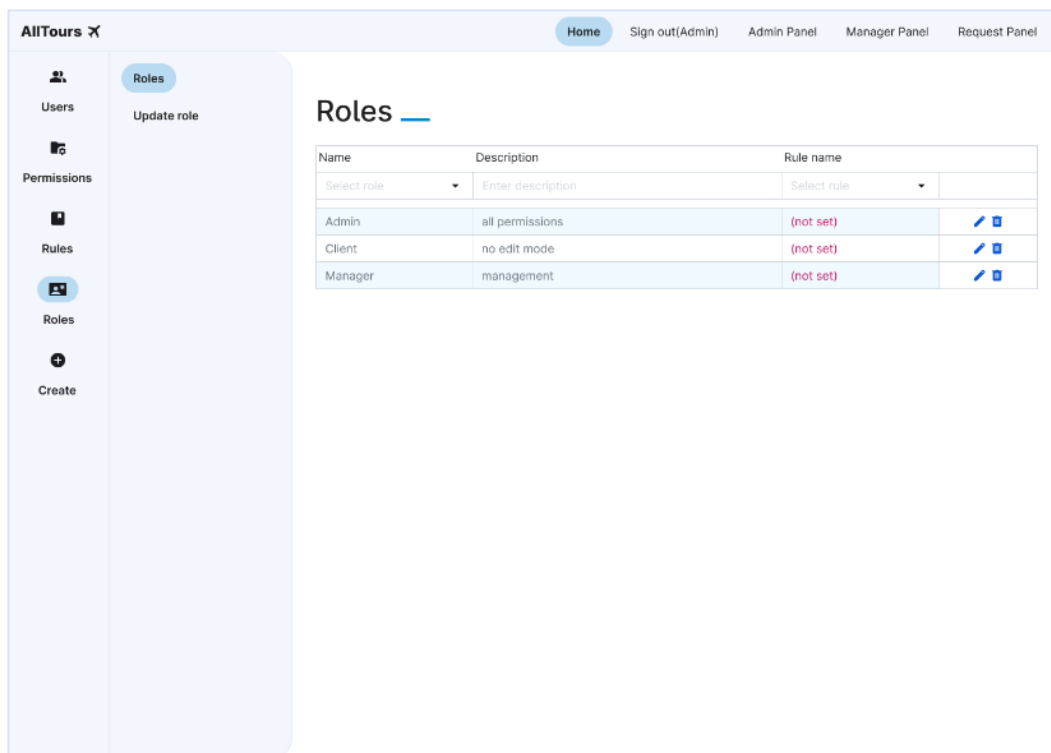


Рисунок 7.20 – Перегляд ролей користувачів

7.15 Створення правил

Адміністратор створює правила для користувачів системи, (рис. 7.21 – 7.22).

AllTours

HomeSign out(Admin)Admin PanelManager PanelRequest Panel

UsersPermissionsRulesRolesCreate

PermissionsCreate new permission

Create new permission

Name

Block to view admin panel

Description

Block to view admin panel

Rule

Select rule

Data

Children

Save

Рисунок 7.21 – Створення правил для користувачів системою

AllTours

HomeSign out(Admin)Admin PanelManager PanelRequest Panel

UsersPermissionsRulesRolesCreate

PermissionsCreate new permission

Permissions

Name	Description	Rule name	
Select permission	Enter description	Select rule	
Block to view admin pa...	all permissions	(not set)	 
Block to view admin pa...	no edit mode	(not set)	 
Block to view admin pa...	management	(not set)	 

Рисунок 7.22 – Перегляд правил користувачів

7.16 Функціонал менеджера

Менеджеру надається можливість додавання турів на головну сторінку з актуальної інформації яку заповнюють постачальники, (рис. 7.23).

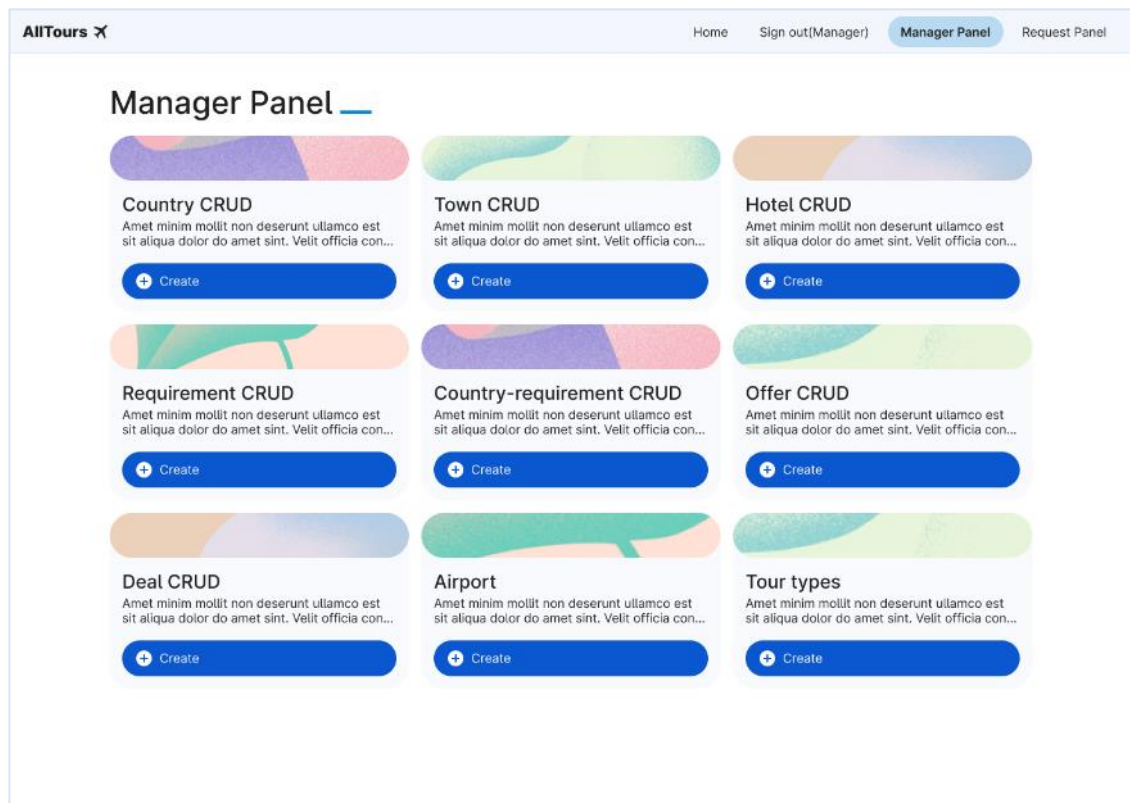


Рисунок 7.23 – Панель управління турами Менеджера

7.17 Створення турів

Менеджер додає нові тури, (рис. 7.24 – 7.26).

AllTours ✕
Home
Sign out(Manager)
Manager Panel
Request Panel

Update Offer —

Name

Photo upload



Description tour

Amet minim mollit non deserunt ullamco est sit aliqua dolor do amet sint. Velit officia consequat duis enim velit mollit. Exercitation veniam consequat sunt nostrud amet...

Рисунок 7.24 – Заповнення туру. Частина 1

Price

2021-10-30 → 2021-11-06

<< <
Feb 2021
> >>

Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Type ID

Update

Рисунок 7.25 – Заповнення туру. Частина 2

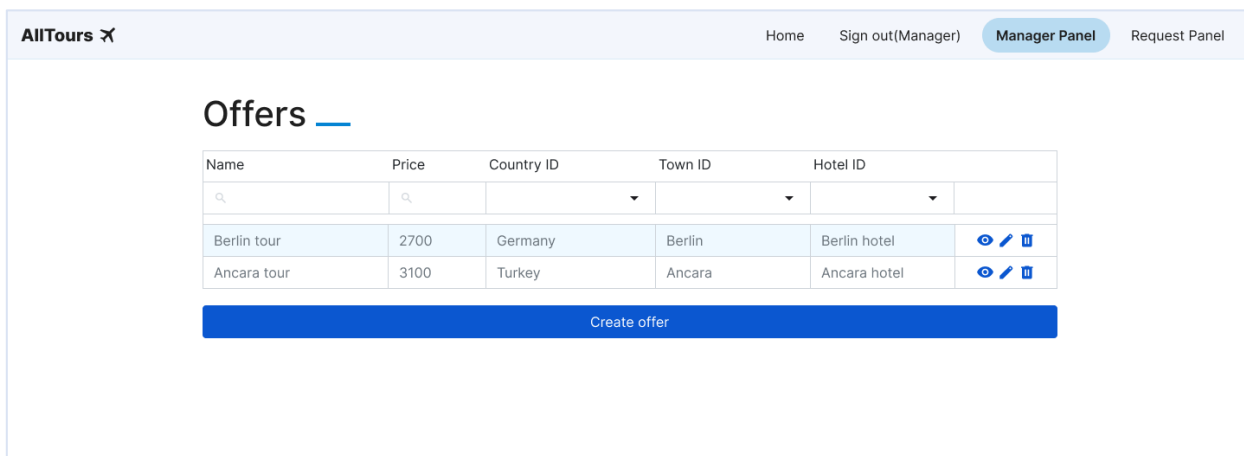


Рисунок 7.26 – Найменування турів

7.18 Попередній документ тура клієнта

Менеджеру надається можливість вибрати тур клієнтові і подальше формування попереднього документа на друк, (рис. 7.27 – 7.31).

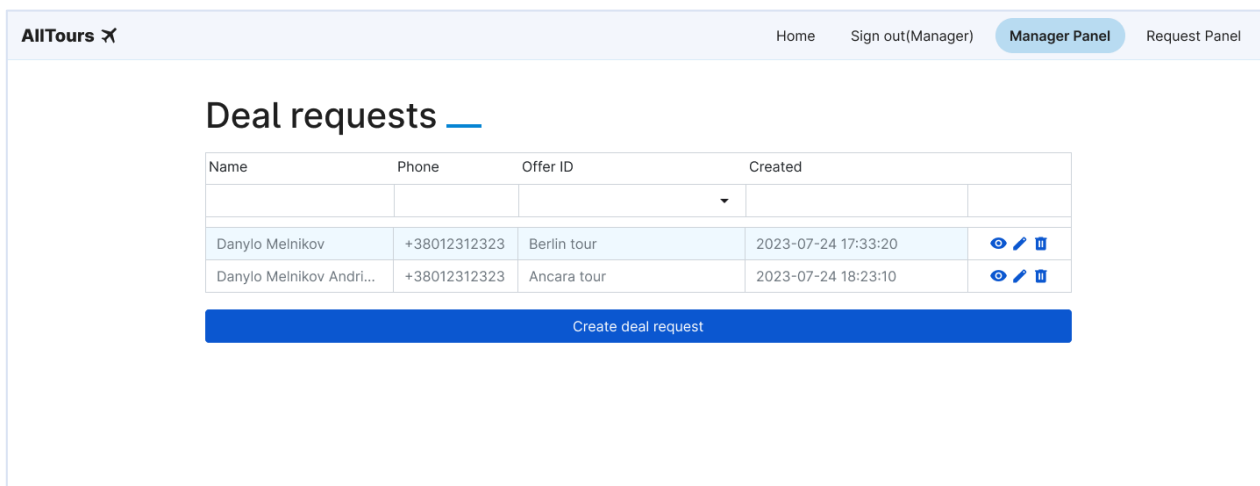


Рисунок 7.27 – Список створених заявок

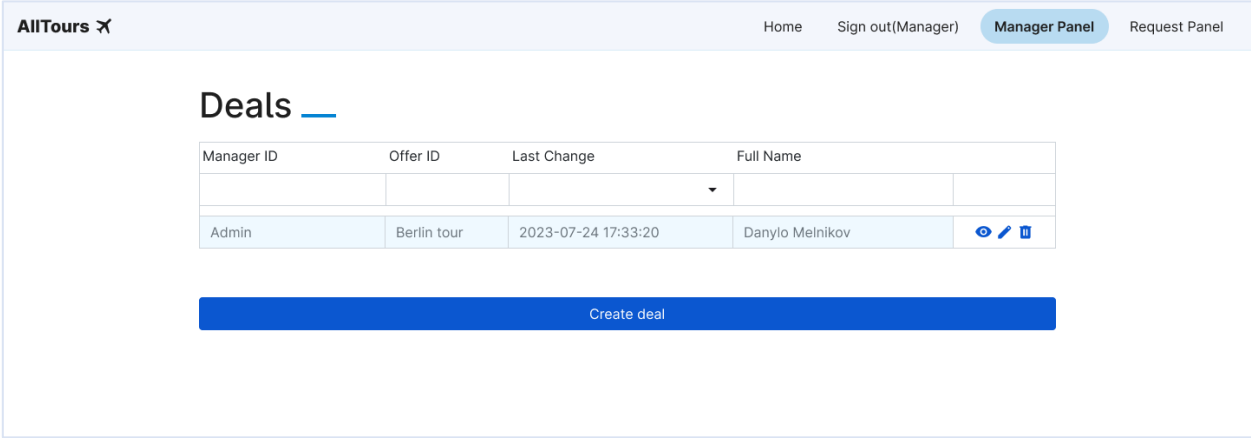


Рисунок 7.28 – Перегляд туру клієнта

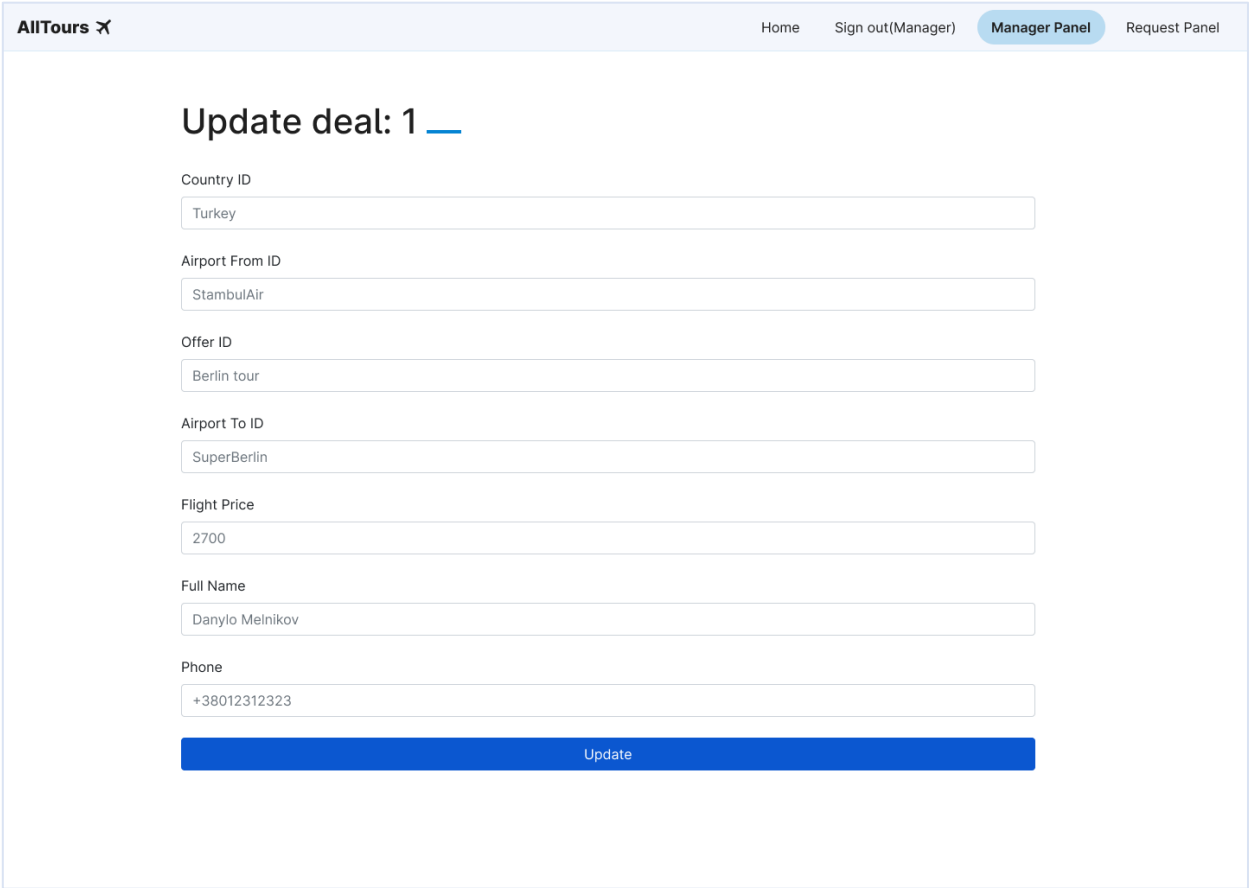


Рисунок 7.29 – Перевірка та редагування заяви

AllTours ✕ Home Sign out(Manager) **Manager Panel** Request Panel

1

Manager ID	Admin
Offer ID	Berlin tour
Created	2023-07-24 17:33:20
User ID	Danylo Melnikov

PDF Update Delete

Рисунок 7.30 – Сторінка формування попереднього документа

Договір про надання туристичних послуг

ПІБ клієнта: Danylo Melnikov
 Телефон: +38012312323
 Послуга: Berlin tour
 Країна призначення: Germany
 Місто призначення: Berlin
 Готель: Berlin hotel
 Вимоги:
 • Visa
 • Vaccination
 Прибуття: 2023/02/01, від'їзд: 2023/02/10
 Виліт з Аеропорту: StambulAir, в аеропорт: SuperBerlin
 Ціна туру: 2700, ціна перельоту: 2700

Підпис клієнта:

Підпис консультанта:

Рисунок 7.31 – Сформований попередній документ тура

7.19 Функціонал менеджера-постачальника

Постачальнику послуг надається можливість додавання, видалення, оновлення Країн, Міст, Готелів, Вимог до країн (наприклад щеплення), аеропорти и типи турів, (рис. 7.32).

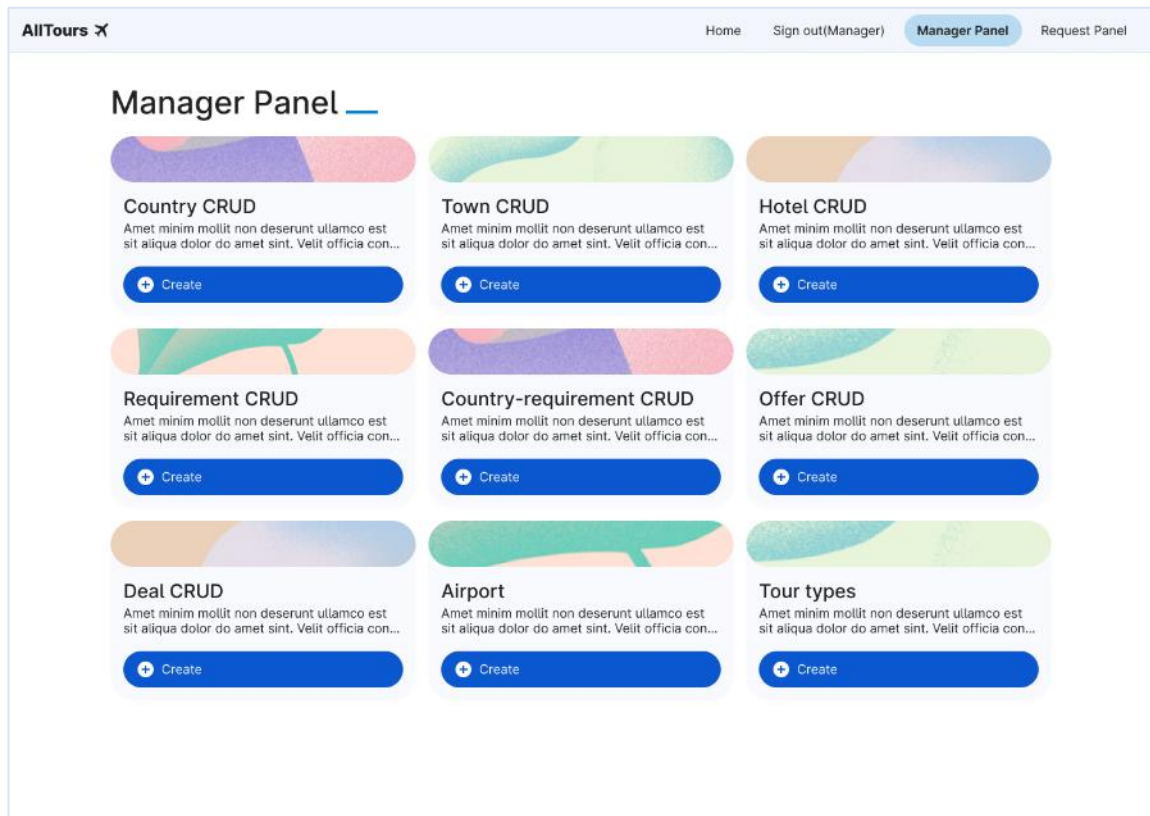


Рисунок 7.32 – Панель Менеджера-постачальника

7.20 Заповнення даних країн

Постачальник послуг наповнює дані країн, (рис. 7.33-7.34).

The screenshot shows the 'Create country' form within the 'Manager Panel'. The form has the following fields and elements:

- Title**: 'Create country' with a blue underline.
- Name**: A text input field containing 'Germany'.
- Description**: A larger text input field also containing 'Germany'.
- Create Button**: A prominent blue button at the bottom labeled 'Create'.

Рисунок 7.33 – Заповнення даних країн

AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Countries

Name	
Turkey	View Edit Delete
Germany	View Edit Delete

[Add](#)

Рисунок 7.34 – Перегляд даних країни

7.21 Заповнення даних міст

Менеджер постачальник заповнює дані міст, (рис. 7.35 – 7.36).

AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Create town

Name
Berlin

Coord Grad One
120

Coord Grad Two
-140

Description
Berlin

Country ID
Germany

[Create](#)

Рисунок 7.35 – Заповнення даних городів

AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Towns

Name	
Ancara	View Edit Delete
Berlin	View Edit Delete

[Add](#)

Рисунок 7.36 – Перегляд даних городів

7.22 Заповнення даних готелів

Менеджер поставщик заповнює дані готелів, (рис. 7.37-7.38).

AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Create hotel

Name
Berlin hotel

Stars
5

Phone
4123123123213

Email
sadasd@gmail.com

Country ID
Germany

Coord Grad One
41.042966

Coord Grad Two
28.980435

[Create](#)

Рисунок 7.37 – Заповнення даних готеля

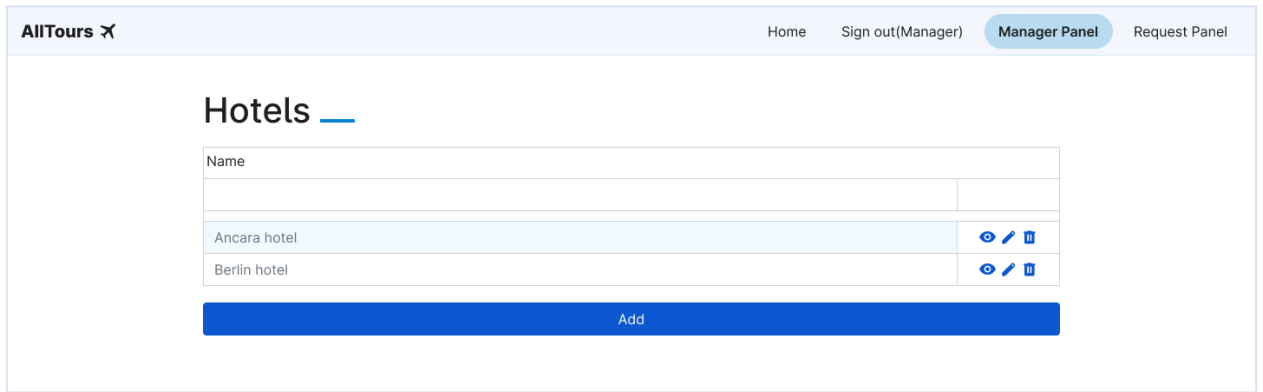


Рисунок 7.38 – Перегляд даних готеля

7.23 Заповнення даних вимог

Менеджер постачальник заповнює дані вимог, (рис. 7.39-7.40).

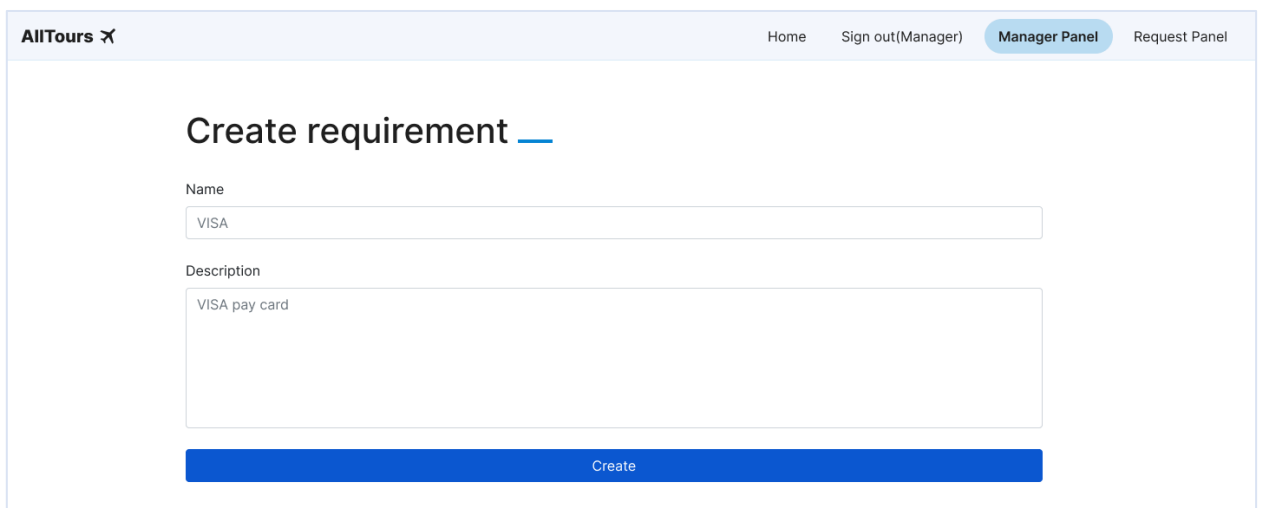


Рисунок 7.39 – Заповнення даних вимог

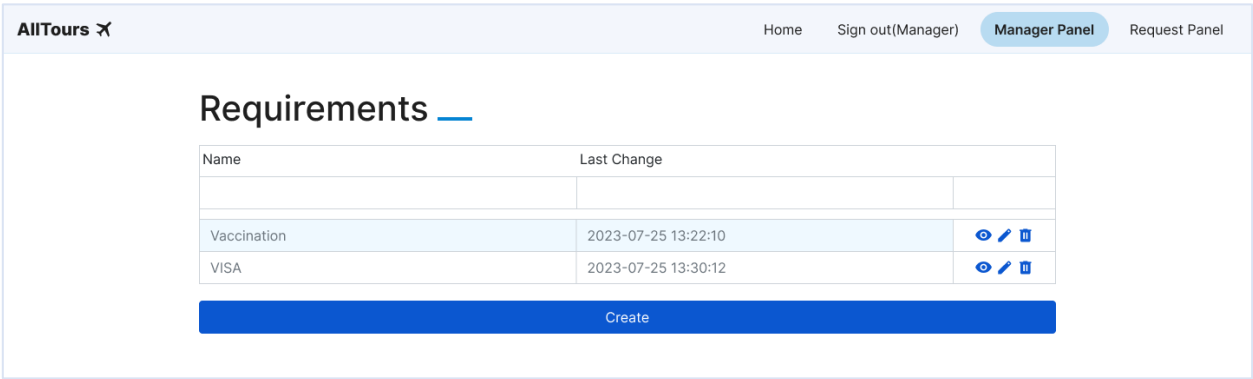


Рисунок 7.40 – Перегляд даних вимог

7.24 Заповнення даних вимог

Менеджер постачальник заповнює дані вимог країн, (рис. 7.41-7.42).

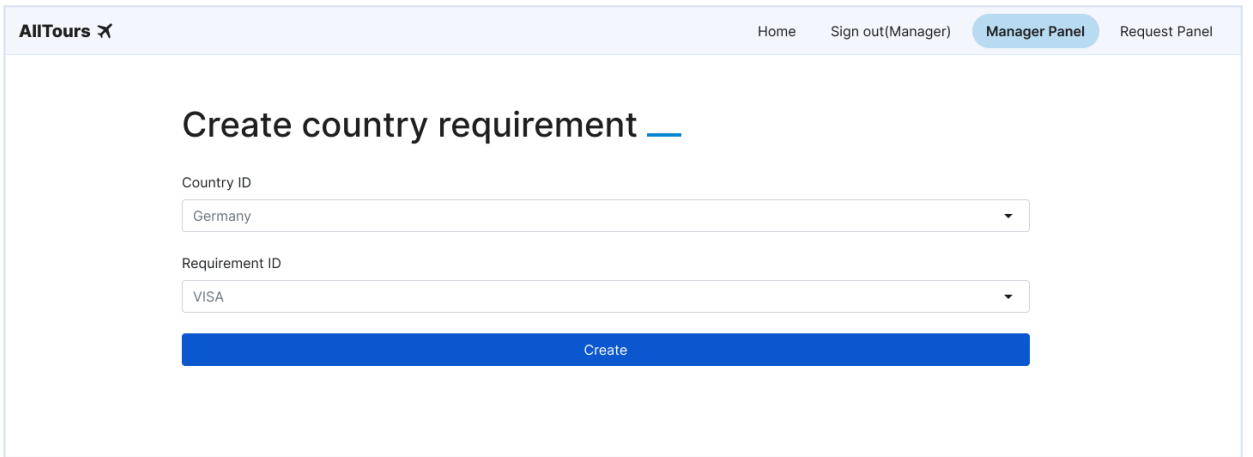


Рисунок 7.41 – Заповнення вимог даних країн

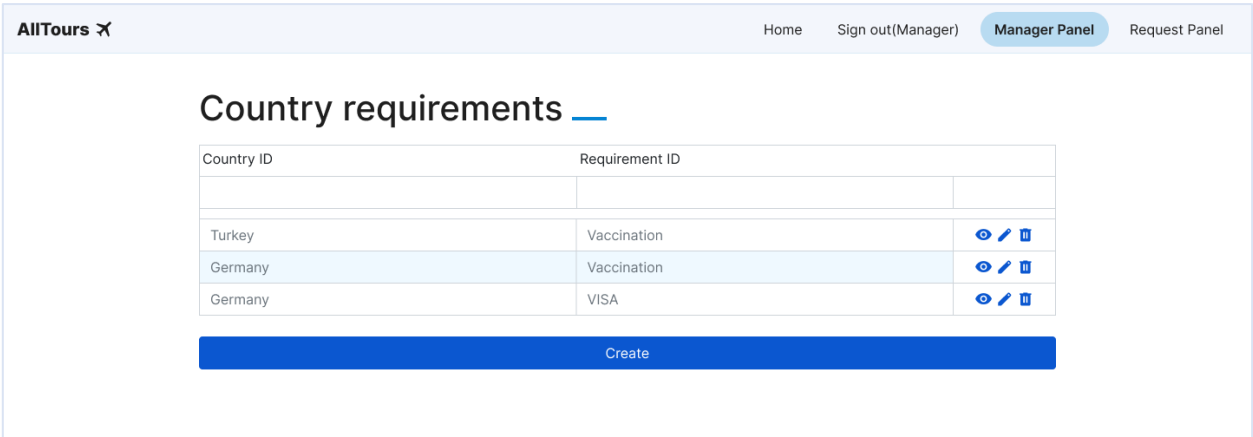
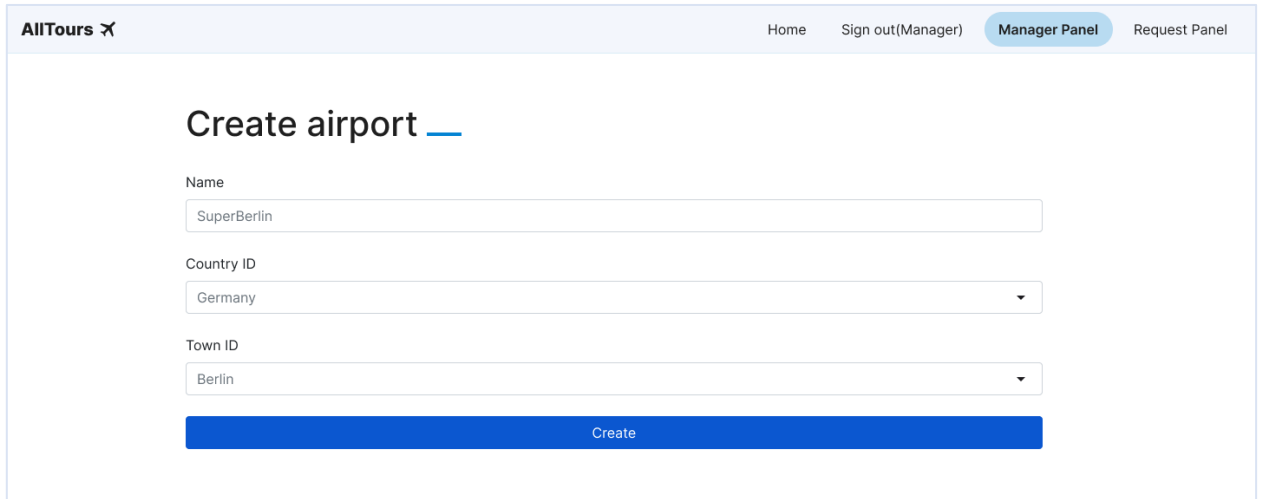


Рисунок 7.42 – Перегляд вимог даних країн

7.25 Заповнення даних аеропортів

Менеджер постачальник заповнює дані аеропортів, (рис. 7.43-7.44).



AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Create airport

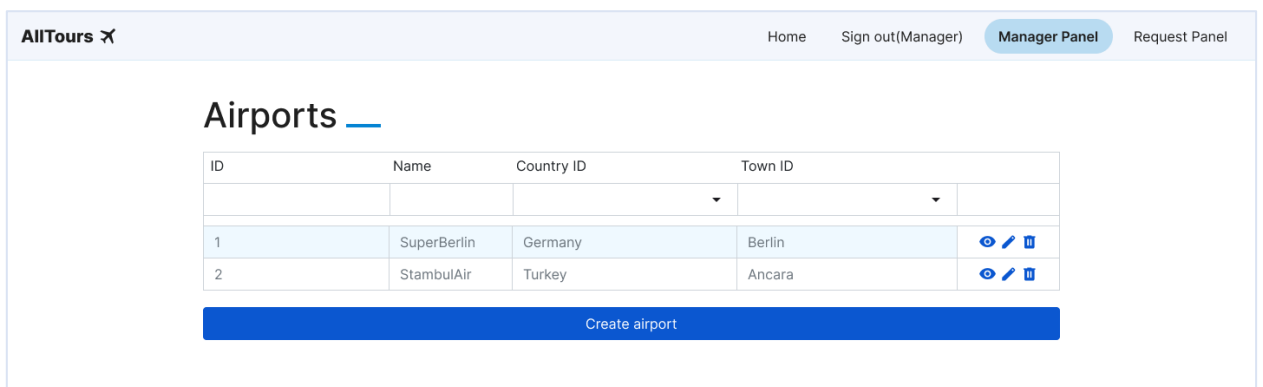
Name
SuperBerlin

Country ID
Germany

Town ID
Berlin

Create

Рисунок 7.43 – Заповнення даних аеропортів



AllTours Home Sign out(Manager) **Manager Panel** Request Panel

Airports

ID	Name	Country ID	Town ID	
1	SuperBerlin	Germany	Berlin	
2	StambulAir	Turkey	Ancara	

Create airport

Рисунок 7.44 – Перелік аеропортів

7.26 Заповнення даних типів турів

Менеджер поставщик заповнює дані типів турів, (рис. 7.45-7.46.)

AllTours

HomeSign out(Manager)Manager PanelRequest Panel

Create tour type

Name

Jumping

Description

Air jumping

Create

Рисунок 7.45 – Заповнення даних турів

AllTours

HomeSign out(Manager)Manager PanelRequest Panel

Tour types

ID	Name	Description	
1	Snowboard	n1 snow tour	  
2	Jumping	Air jumping	  
3	Rest at the h...	Rest at the hotel	  
4	Excursion	Excursion	  

Create

Рисунок 7.46 – Перегляд даних турів

ВИСНОВКИ

В результаті виконання дипломної кваліфікаційної роботи магістра було проведено повний цикл розробки програмного забезпечення, що включає аналіз предметної області, проєктування архітектури системи, реалізацію компонентів системи, а також тестування отриманого продукту.

Аналіз предметної області виявив основні потреби та вимоги до системи з боку туристичних агенцій, що дозволило ефективно спланувати структуру та функціонал майбутнього програмного продукту. Вибір технологій та інструментарію був обґрунтований сучасними тенденціями в області розробки програмного забезпечення та специфікою завдань, що стояли перед системою.

Практична частина проєкту включала розробку бази даних, модулів для обробки даних, користувацького інтерфейсу, а також системи звітності. Реалізована система демонструє високий рівень надійності, зручності у використанні та гнучкості налаштувань під конкретні потреби користувача.

Тестування програмного продукту показало його ефективність та правильність роботи. Експериментальне дослідження підтвердило, що автоматизація дозволяє значно покращити швидкість та якість обслуговування клієнтів туристичної агенції.

Розділи керівництва для програміста та оператора забезпечують необхідну інформацію для ефективного використання системи та її подальшої підтримки. Важливою частиною роботи стала розробка заходів з охорони праці, які забезпечують безпеку використання системи в умовах реальної експлуатації.

Система може бути успішно впроваджена на практиці і принести значну користь для діяльності туристичних агенцій. Рекомендації, надані у висновках, можуть бути використані для подальшого розвитку та масштабування системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Основи туризмознавства. Ринок туристичних послуг. Схема функціонування туристичного бізнесу [Електрон. ресурс]. – Режим доступу: <http://turizmo-znavstvo.blogspot.com/2014/05/6.html>.
2. Організація та планування діяльності туристичних підприємств: теорія та практика. Схема функціонування туристичної агенції [Електрон. ресурс]. – Режим доступу: <https://geography.lnu.edu.ua/wp-content/uploads/2021/02/Planuvannia3-dodane.pdf>.
3. Туроператорська та турагентська діяльність. Складемо схему функціонування туристичного сайту [Електрон. ресурс]. – Режим доступу: <https://kychinskiy.blogspot.com/2017/12/blog-post.html>.
4. Bevington F.R. Data Reduction and Error Analysis for the Physical Sciences Third Edition. / F.R. Bevington, K.D. Robinson. – New York: McGraw-Hill College, 2002. – 336p.
5. The min-max scaling method | Preprocessing data | [Electronic resource]. – Access mode: <https://scikit-learn.org/stable/modules/preprocessing.html>.
6. Tourplan | Solutions | [Electronic resource]. – Access mode: <https://www.tourplan.com/>.
7. Lemax | Solutions | [Electronic resource]. – Access mode: <https://www.lemax.net/>.
8. Dolphin Dynamics [Electronic resource]. – Access mode: <https://www.dolphind.com/>.
9. Python | Documentation | [Electronic resource]. – Access mode: <https://docs.python.org/3/>.
10. PHP | Manual | [Electronic resource]. – Access mode: <https://www.php.net/manual/en/>.
11. JavaScript | Tutorial | [Electronic resource]. – Access mode: <https://javascript.info/>.

12. Java | Documentation | [Electronic resource]. – Access mode: <https://docs.oracle.com/en/java/>.

13. PhpStorm | Documentation | [Electronic resource]. – Access mode: <https://www.jetbrains.com/help/phpstorm/getting-started.html>.

14. Visual Studio Code | FAQ | [Electronic resource]. – Access mode: <https://code.visualstudio.com/docs/supporting/faq> Eclipse | Documentation | [Electronic resource]. – Access mode: <https://docs.python.org/3/>.

15. Eclipse | Documentation | [Electronic resource]. – Access mode: <https://help.eclipse.org/2023-06/index.jsp>.

16. Sublime Text | Documentation | [Electronic resource]. – Access mode: <https://www.sublimetext.com/docs/>.

17. Yii2 | The Definitive Guide to Yii 2.0 | [Electronic resource]. – Access mode: <https://www.yiiframework.com/doc/guide/2.0/en/structure-assets>.

18. Ferraiolo D.F. Role-Based Access Control. / D.F. Ferraiolo, D.R. Kuhn. – Gaithersburg: Artech House, 2007. – 418p.

19. Bootstrap | Documentation | [Electronic resource]. – Access mode: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>.

20. Relational database [Electronic resource]. – Access mode: <https://www.techtarget.com/searchdatamanagement/definition/relational-database>.

21. Laravel [Electronic resource]. – Access mode: <https://laravel.com/docs/10.x>.

22. React [Electronic resource]. – Access mode: <https://react.dev/learn>.

23. Symfony [Electronic resource]. – Access mode: <https://symfony.com/doc/current/index.html>.

24. Christianson, B. Communications and Multimedia Security. / B. Christianson, P. Hu, B. Snook. – Boston: D. Springer, 1995. – 242 p.

25. Архітектура клієнт-сервер. Архітектура та проектування компонентних систем [Електрон. ресурс]. – Режим доступу: <https://victana.lviv.ua/knyhy/konspekty-lektsii/133-kros-platformenne->

prohramuvannia-ta-khmarni-servisy/568-lektsiia-20-arkhitektura-ta-proektuvannia-komponentnykh-system-2016-r.

26. MVC | Docs MVC | [Electronic resource]. – Access mode: <https://jakeydocs.readthedocs.io/en/latest/mvc/>.

27. CRM | Customer relationship management and big data enabled: Personalization & customization of services | [Electronic resource]. – Access mode: <https://www.sciencedirect.com/science/article/pii/S2210832718300735>.

28. CRUD [Electronic resource]. – Access mode: <https://www.phpcrudgenerator.com/documentation/index>.

29. Діаграма варіантів використання Адмін [Електрон. ресурс]. – Режим доступу: <http://www.tsatu.edu.ua/kn/wp-content/uploads/sites/16/laboratorna-robota-5-diahramy-variantiv-vykorystannja.pdf>.

30. Діаграма класів. Створення ієрархії об'єктів ІС в межах заданої предметної галузі. Побудова діаграм класів. [Електрон. ресурс]. – Режим доступу: <https://core.ac.uk/download/pdf/323013267.pdf>.

31. Diagram of components [Electronic resource]. – Access mode: <https://www.omg.org/spec/UML/2.1.2/Superstructure/PDF>.

32. Jacobson, I. Unified Software Development Process, The (Addison-Wesley Object Technology Series) / I. Jacobson, G. Booch, J. Rumbaugh. – Boston: Addison Wesley Professional, 1999. – 496 p.

33. Тест-кейс [Електрон. ресурс]. – Режим доступу: <https://qalight.ua/baza-znaniy/test-case-2/>.

34. Чек-ліст [Електрон. ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/work-with-checklist/>

35. Network Topology [Electronic resource]. – Access mode: <https://www.javatpoint.com/computer-network-topologies>.

36. Топологія мережі. Мережні архітектури. [Електрон. ресурс]. – Режим доступу: <http://nikolay.in.ua/navchaemos/windows-xp/merezhni-tekhnologiji/63-merezhni-arkhitekturi>.

37. Davenport, T. IDEF0 – Davenport T. Process Innovation : reengineering work through information technology / T. Davenport. – Boston: Harvard Business School Press, 1993. – 364 p.

38. BPMN | Specification | [Electronic resource]. – Access mode: <https://www.omg.org/spec/BPMN/2.0/PDF/>.

39. UML | Specification | [Electronic resource]. – Access mode: <https://www.omg.org/spec/UML/2.4/Superstructure/PDF>.

ДОДАТОК А
Технічне завдання

A.1 Загальні вимоги до розробки програмного продукту для ведення інформаційно туристичної діяльності

A.1.1 Загальні відомості

Повне найменування системи та її умовне позначення: "Дослідження та програмна реалізація процесу автоматизації діяльності туристичної агенції".
Коротка характеристика області застосування.

Система призначена для застосування в агентстві з оформлення туристичних квитків.

Порядок оформлення і пред'явлення результатів робіт зі створення системи, з виготовлення та налагодження окремих засобів (технічних, програмних, інформаційних) і програмно-технічних (програмно-методичних) комплексів системи: AIC "All Tours" надаються у вигляді виконуваних модулів по завершенню всього обсягу робіт, технічні засоби тестуються. Оформлення результатів робіт зі створення системи проводиться шляхом моделювання стандартами UML і CASE-засобами AIC "AllTours".

A.1.2 Підстави на розробку

Підставою для розробки технічного завдання є завдання до кваліфікаційної дипломної роботи.

Найменування теми розробки – "Дослідження та програмна реалізація процесу автоматизації діяльності туристичної агенції "

A.1.3 Призначення і цілі створення системи

Функціональне призначення системи: AIC "All Tours" призначена для автоматизації роботи агентства з оформлення туристичних квитків.

Експлуатаційне призначення системи: Система повинна

експлуатуватися співробітниками організації.

Мета створення системи: Система прискорює процес формування попередніх документів на оформлення туристичних квитків, тим самим спрощує роботу агентства.

- система створює тури на головній сторінці;
- система формує попередній документ договору туру;
- система виконує розрахунок ціни туру за кількістю днів і ціни туру;
- система виконує бронювання готелів;
- система виконує пошук геолокації готеля за країною, городом, готелем;
- система використовує автоматичне заповнення даних с персональної інформації при реєстрації;
- система має блок на пошук персонального туру;
- система має адміністративну панель для розмежування прав користувачів;
- система використовує підтримку актуальності даних з інформації менеджерів-постачальників;
- система має панель менеджерів для створення турів;
- система використовує інформацію з вимогами в'їзду в країну.

A.1.4 Вимоги до системи

Перелік підсистем, їх призначення та основні характеристики, вимоги до числа рівнів ієрархії і ступеня централізації системи АІС "All Tours" включає наступні підсистеми:

- оплата заказу;
- оформлення попередньо сформованого документу;
- система реєстрації та авторизації;
- система бронювання;

- система заповнення даних;
- система геолокації;
- система актуальності даних;
- календарна система поїздок;
- персональна система адміністратора;
- персональна система користувача;
- персональна система менеджера;
- персональна система менеджера-постачальника.

Підсистема "Сплата заказу" призначена для реєстрації сплати замовлення туристичного квитка.

Підсистема "Оформлення попередньо сформованого документу" призначена для швидкого формування документа туристичного квитка з побажань клієнта щодо відпочинку.

Підсистема "Система авторизації" призначена для реєстрації клієнта на сайті з заповненням персональної інформації та подальшого сценарія користувача.

Підсистема "Система бронювання" призначена для системи бронювання відпочинку, а саме країна, місто, готель відпочинку, трансфер послуг, страхування, подальші вимоги щодо країни.

Підсистема "Система геолокації" призначена для відображення місцезнаходження тура котрий обрав клієнт.

Підсистема "Система актуальності даних" призначена для підтримки актуальної бази турів за допомогою роле менеджер-поставщик.

Підсистема "Календарна система поїздок" призначена для розрахунки ціни та бронювання готелів з вказаних клієнтом днів відпочинку.

Підсистема "Персональна система адміністратора" призначена для розмежування прав користувачів системою, блокування, видалення, редагування, створення персональних даних, правил користувачів системою.

Підсистема "Персональна система користувача" призначена для зберігання даних клієнта та автозаповнення полів туристичного квитка з персональної інформації клієнта при реєстрації.

Підсистема "Персональна система менеджера" призначена для створення турів які використовують актуальну інформацію менеджерів-постачальників з бази даних.

Підсистема "Персональна система користувача" призначена для зберігання даних клієнта та автозаповнення полів туристичного квитка з персональної інформації клієнта при реєстрації.

Підсистема "Персональна система менеджера-поставщика" призначена для заповнення бази даних актуальною інформацією партнерів менеджерів-постачальників.

А.1.5 Вимоги до способів і засобів зв'язку компонентів системи

Вимоги до способів і засобів зв'язку для інформаційного обміну між компонентами системи:

Інформаційний обмін здійснюється за допомогою локальної мережі.

Вимоги до режимів функціонування системи:

Система повинна підтримувати багато користувачів і автономний режим роботи. Користувачі здійснюють доступ до системи через мережу інтернет або Call-центр.

Вимоги до чисельності та кваліфікації персоналу системи

Вимоги до кваліфікації персоналу, порядку його підготовки і контролю знань і навичок:

Співробітник, що займаються прийомом заявок, повинен володіти навичками роботи з персональним комп'ютером на рівні користувача. Чисельність персоналу може бути різною в залежності від обсягу замовлень.

A.1.6 Вимоги до надійності

Відновлення системи при помилках в роботі апаратних засобів (крім носіїв даних і програм) і помилки, пов'язаних з програмним забезпеченням (ОС і драйвери пристроїв), відновлення працездатності покладається на ОС.

При збої в системі електропостачання апаратної частини, що призводять до перезавантаження ОС, відновлення програми має відбуватися після перезапуску ОС і запуску виконуваного файлу системи.

Також повинна бути забезпечена працездатність системи в цілому в разі виникнення збоїв, аварій і відмов на окремих робочих станціях. Для захисту апаратури від кидків напруги і комутаційних перешкод повинні застосовуватися фільтри, а для можливості збереження даних користувачем при збої в системі електроживлення рекомендується застосовувати джерела безперебійного живлення.

A.1.7 Вимоги до безпеки

Замовником забезпечує відповідність технічних рішень, використаних при модифікації і розробки підсистеми, чинним нормам, правилам техніки безпеки, пожежної безпеки і вибухобезпеки, охорони навколишнього середовища.

Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Умови експлуатації, а також види і періодичність обслуговування технічних засобів підсистеми повинні відповідати вимогам по експлуатації, технічного обслуговування, ремонту і зберігання, викладеним в документації заводу-виготовлювача (виробника) на них.

Вимоги до захисту інформації від несанкціонованого доступу

Для захисту інформації від несанкціонованого доступу Система повинна забезпечувати:

- ідентифікацію та автентифікацію користувача;
- перевірку прав і обмежень доступу користувача на рівні функцій і масивів даних при роботі з системою.

Вимоги щодо збереження інформації при аваріях

Необхідно передбачити можливість резервного копіювання даних системи засобами програмного забезпечення, що поставляється Розробником.

Вимоги по стандартизації та уніфікації

Для даної системи повинна застосовуватися каскадна модель життєвого циклу ПО.

В системі повинні використовуватися класифікатори і єдині класифікатори і словники для різних видів алфавітно-цифрової і текстової інформації.

Інтерфейс системи, файли довідок і будь-яка текстова інформація в програмі повинні бути англійською та російською мовою.

Екранні форми повинні проектуватися з урахуванням вимог уніфікації:

Всі екранні форми користувацького інтерфейсу повинні бути виконані в єдиному графічному дизайні, з однаковим розташуванням основних елементів управління і навігації.

A.1.8 Вимоги до завдань, що виконуються системою

По кожній підсистемі перелік функцій, завдань або їх компонентів (в тому числі забезпечують взаємодію частин системи), що підтримують автоматизації

Інформаційна система повинна забезпечувати виконання таких функцій:

- підсистема сплати заказу;
- підсистема оформлення попередньо сформованого документу;
- підсистема системи авторизації;

- підсистема системи бронювання;
- підсистема системи геолокації;
- підсистема системи актуальності даних;
- підсистема календарної системи поїздок ;
- підсистема персональної система адміністратора;
- підсистема персональної системи менеджера;
- підсистема персональної системи користувача;
- підсистема персональна системи менеджера-постачальника.

A.1.9 Вимога до видів забезпечення

До інформаційного забезпечення системи

До складу, структури та способом організації даних в системі

Дані системи зберігаються на одній локальній машині. На вхід системи подається опис замовлення, на виході повинен бути рахунок і ідентифікаційний номер замовника.

До структури процесу збору, обробки, передачі даних в системі і представлення даних.

Дані вводяться в систему вручну, обробляються і надаються користувачеві в необхідному вигляді (електронний, друкований).

До захисту даних від руйнувань при аваріях і збоях в електроживленні системи

У комплекс технічних засобів повинен входити джерело безперебійного живлення. Робота даного джерела повинна становити не менше півгодини для коректного завершення роботи системи.

До контролю, зберігання, оновлення та відновлення даних

Система повинна підтримувати автоматичне щоденне резервне копіювання.

Вимоги до програмного забезпечення системи

Система повинна працювати в операційних системах Windows XP / Vista / 7/8/10

Вимоги до технічного забезпечення системи

До видів технічних засобів, в тому числі до видів комплексів технічних засобів, програмно-технічних комплексів та інших комплектуючих виробів, допустимих до використання в системі.

У комплекс технічних засобів повинні входити такі елементи:

- робочі станції;
- джерела безперебійного живлення;
- кошти для побудови Локально обчислювальної мережі;
- сервер БД;
- принтер.

Вимоги до сервера:

- пам'ять 8 Гб;
- процесор 2.2 ГГц Intel Xeon 5500 мінімум;
- швидкість диска SATA 8 Гбіт / с;
- мережевий адаптер 10 Гбіт / с;
- операційна система Windows Server 2008-12.

Вимоги до робочої станції:

- процесор 2 ГГц;
- пам'ять 2 Гб;
- жорсткий диск не менше 500;
- операційна система Windows 7-10;
- мережевий адаптер 100 Мбіт / с.

Технічні засоби ІС описані з урахуванням вимог до функціонування прикладного програмного комплексу.

Технічні засоби повинні забезпечити:

- цілодобовий режим роботи комплексу технічних засобів і обладнання;

– гарантоване виконання всього комплексу програмного забезпечення разі збою або виходу з ладу частини обладнання;

– захист даних від несанкціонованого доступу;

– сервера і робочі місця повинні бути об'єднані локальною мережею.

Зобразимо представлену топологію локальної обчислювальної мережі (ЛОМ) для "All Tours", (рис. А.1) [35].

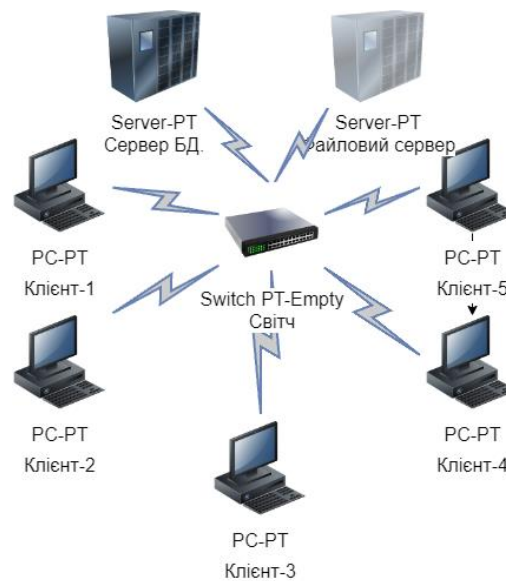


Рисунок А.1 – Візуалізація зв'язку комутатора до сервера БД і файлового сервера з підключенням 5 робочих станцій. Топологія мережі – зірка [36]

У комплекс технічних засобів повинні входити такі елементи:

- робочі станції;
- джерело безперебійного живлення;
- середовище передачі даних між робочими станціями (наприклад, кручена пара UTP 5e);
- принтер;
- технічні засоби купуються Замовником самостійно.

До функціональних, конструктивних і експлуатаційних характеристик засобів технічного забезпечення системи.

Процесор Intel Pentium IV 2 ГГц і вище, оперативна пам'ять не менше 2

Гб, обсяг жорсткого диска не менше 500 Гбайт.

Вимоги до організаційного забезпечення системи

Для організаційного забезпечення призводять вимоги:

До структури і функцій підрозділів, що беруть участь у функціонуванні системи або забезпечують експлуатацію.

Функціонування системи забезпечує інженер-системотехнік (детально по РОЛМ), в експлуатації беруть участь 4 співробітника.

До організації функціонування системи і порядку взаємодії персоналу автоматизованої системи і персоналу об'єкта автоматизації.

Організаційне забезпечення повинно бути достатнім для ефективного виконання персоналом покладених на нього обов'язків при здійсненні функцій системи.

До захисту від помилкових дій персоналу системи.

Захист від помилок персоналу полягає в перевірці заповнення даних в деяких полях, можливості відновлення вихідних даних і скасування останніх змін, розмежування доступу по функціям і повноваженням співробітників.

А.1.10 Характеристика об'єктів автоматизації

Як об'єкт автоматизації виступає процес, пов'язаний з підсистемами системи управління туристичною діяльністю "All Tours". Цей процес включає в себе роботу з наступними підсистемами:

- сплати заказу;
- оформлення попередньо сформованого документу;
- системи реєстрації та авторизації;
- системи бронювання;
- системи заповнення даних;
- системи геолокації;
- системи актуальності даних;

- календарної системи поїздок;
- персональної системи адміністратора;
- персональної системи користувача;
- персональної системи менеджера;
- персональної системи менеджера-постачальника.

A.1.11 Вимоги до документування

Для системи на різних стадіях створення повинні бути випущені такі документи:

- схема організаційної структури;
- схема функціональної структури;
- перелік вхідних сигналів і даних;
- перелік вихідних сигналів (документів);
- пояснювальна записка до технічного проєкту;
- опис автоматизованих функцій;
- опис постановки завдань (комплексу задач);
- опис організації інформаційної бази;
- опис масиву інформації;
- опис програмного забезпечення;
- опис інструментарію програмування
- опис інструментарію з тестування;
- опис бізнес-процесів;
- опис системи проєктування системи;
- керівництво користувача.

A.1.12 Етапи розробки програмного продукту

Розробка повинна бути проведена в 6 стадій:

- розробка технічного завдання;
- розробка документації технічного проєкту;
- робоче проєктування;
- реалізація та тестування;
- документація користувача.

A.1.13 Огляд етапів розробки

На стадії розробки технічного завдання повинен бути виконаний етап розробки, узгодження і затвердження цього технічного завдання.

На стадії розробки проєктної документації повинен бути виконаний етап розробки проєктної документації.

На стадії робочого проєктування повинні бути виконані перераховані нижче етапи робіт: розробка інформаційної системи; підготовка документації інструментарія розробки.

На стадії реалізація та тестування повинні бути виконана підготовка і тестування програми.

На стадії документації користувача повинна виконуватися робота з підготовкою документації користувача для подальшої внесення ясності з використання програми.

A.1.14 Зміст робіт за етапами

На етапі розробки технічного завдання повинні бути виконані перераховані нижче роботи:

- постановка завдання;
- визначення та уточнення вимог до технічних засобів;
- визначення вимог до інформаційної системи;
- визначення стадій, етапів і термінів розробки інформаційної системи і документації на неї;

- обґрунтування об'єктів автоматизації;
- узгодження і затвердження технічного завдання.

На етапі розробки технічної проєктної документації повинні бути виконані перераховані нижче роботи:

- визначення основних бізнес-процесів (у вигляді діаграм IDEF0, BPMN [37]-[37];
- визначення основних варіантів використання Системи для шести категорій користувачів (Гість, Авторизований користувач, Адміністратор, Менеджер, Менеджер-постачальник) у вигляді UML діаграм варіантів використання [39];
- опис концептуального та логічного рівня інформаційної моделі
- проєктування основних компонентів і алгоритмів системи у вигляді відповідних UML діаграм;
- проєктування структури призначеного для користувача інтерфейсу;
- погодження та затвердження проєктної документації.

На етапі реалізації та тестуванні повинна бути виконана робота по розробці інформаційної системи на основі проєктної документації, кодування, тестування і налагодженні.

На етапі документації користувача повинна бути розроблена документація користувача.

A.2 Порядок контролю і приймання системи

A.2.1 Методи випробування системи і її складових частин

Випробування системи проводиться шляхом введення великої кількості даних, введення однакових відомостей, відомостей іншого типу даних, даних більшого діапазону. Проводиться перевірка відповідності екранних форм опису в керівництві для користувача (тестування інтерфейсу).

A.2.2 Загальні вимоги до приймання робіт по стадіям

Здача здійснюється комісією, до складу якої входять представники Замовника (Викладач) і Виконавця. За результатами прийому підписується акт приймальної комісії.

Всі створювані в рамках цієї роботи програмні вироби передаються Замовнику (Викладачеві), як у вигляді готових модулів, так і у вигляді вихідних кодів.

A.2.3 Статус приймальної комісії

Статус приймальної комісії визначається Замовником (Викладач) до проведення випробувань.

A.3 Постановка завдання до роботи

Метою виконання дипломної кваліфікаційної роботи магістра є створення індивідуального програмного продукту для туристичного агентства з урахуванням санкційних обмежень проти країни агресора та його програмних послуг, для подальшого забезпечення надійного та їх ефективного

функціонування згідно методів параметричної лінійної комбінації даних та параметричного порівняння експериментальних даних А/В тестування.

У роботі розглядається методи параметричної лінійної комбінації даних (кроки) та параметричного порівняння експериментальних даних А/В тестування (кроки), які допоможуть виконати розрахунок виділених параметрів, які безпосередньо впливають на ефективність функціонування туристичної агенції та на основні задачі розробки програмного комплексу.

Отже, для вирішення даного завдання необхідно провести параметричний аналіз параметрів: «Обсяг продажів (у грн)»; «Відсоток успішних бронювань»; «Відсоток повторних клієнтів»; «Рівень задоволеності клієнтів (шкала від 1 до 10)»; «Ефективність маркетингових кампаній (%)»; «Середній час обробки замовлення (хвилини);. «Відсоток автоматизації бронювань (%)»; «Інтеграція із зовнішніми сервісами (%)»; «Рівень безпеки даних (шкала від 1 до 10)»; «Вартість залучення клієнта (у грошах)».

Загальна оцінка обчислюється як зважена сума нормалізованих параметрів представлених вище, формула (1.2).

З метою вирішення проблем, що виникають при використанні досліджуваних методів, було прийнято рішення використати метод параметричної лінійної комбінації даних для розрахунку ефективності автоматизації розробленого програмного засобу функціонування туристичного агентства згідно діапазону оцінок і встановлення майбутніх завдань аналізуючи результати отриманих параметрів.

ДОДАТОК Б
Опис фізичної бази

Розглянемо детально таблиці у яких реалізації дані сутностей розробленої бази-даних.

У таблиці «auth_rule» зберігається інформація про права аутентифікації, а саме назва, дані, створення та оновлення даних, див. табл. Б.1.

Таблиця Б.1 – Структура таблиці «auth_rule»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
name	VARCHAR(64)	PK, NN	От name до auth_item (rule_name- (FK))	Назва
data	TEXT	NULL		Дані
created_at	INT(11)	NULL		Створення
updated_at	INT(11)	NULL		Оновлення

У таблиці «auth_item» зберігається інформація про справжні елементи, а саме назва, тип, опис, назва правила, дані, створення оновлення, див. табл. Б.2.

Таблиця Б.2 – Структура таблиці «auth_item»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
name	VARCHAR(64)	PK, NN	Від name до auth_item_child (Parent, Child), auth_assignment (item_name) (FK)	Назва
type	INT(11)	NN		Тип
description	TEXT	NULL		Опис
rule_name	VARCHAR(64)	NULL		Назва правил
data	TEXT	NULL		Дані

Продовження таблиці Б.2.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
created_at	INT(11)	NULL		Створення
updated_at	INT(11)	NULL		Оновлення

У таблиці «auth_item_child» зберігається інформація про аунтефікацію дочірніх елементів, а саме батько и дитина, див. табл. Б.3.

Таблиця Б.3 – Структура таблиці «auth_item_child»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
parent	VARCHAR(64)	PK, NN	Auth_item (name)FK	Батько
child	VARCHAR(64)	PK, NN	Auth_item (name)FK	Дитина

У таблиці «auth_assignment» зберігається інформація про призначення аутифікації, а саме пункт імені, № користувача, створення, див. табл. Б.4.

Таблиця Б.4 – Структура таблиці «auth_assignment»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Item_name	VARCHAR(64)	PK, NN	Auth_item (name)FK	Назва ітему
User_id	VARCHAR(64)	PK, NN		Користувач
Created_at	INT(11)	NULL		Створення

У таблиці «token» зберігається інформація про талон або символи, а саме № Користувача, код, створення, тип., див. табл. Б.5.

Таблиця Б.5 – Структура таблиці «token»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
User_id	INT(11)	NN	От User_id до user (id) FK	№ Користувача
code	VARCHAR(12)	NN		Код
Created_at	INT(11)	NN		Створення
type	SMALLINT(6)	NN		Тип

У таблиці «social_account» зберігається інформація про соціальні акаунти, а саме №, № користувача, постачальника, № клієнта, дані, код, електронна адреса, ім'я користувача, див. табл. Б.6.

Таблиця Б.6 – Структура таблиці «social_account»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN	Від user (id) до social_account (user_id) FK	id
User_id	INT(11)	Null		Id Користувача
provider	INT(11)	NN		Створення
Client_id	VARCHAR(255)	NN		Id Клієнта
data	VARCHAR(255)	Null		Дані
code	TEXT	Null, UNIQUE		Код
Creat_at	INT(11)	Null		Коли створено
email	VARCHAR(255)	Null		Електронна адреса
username	VARCHAR(255)	Null		Назва користувача

У таблиці «profile» зберігається інформація про профіль, а саме № користувача, ім'я, публічне ім'я, аватар, № аватару, місцезнаходження, сайт, біографія, часова зона, див. табл. Б.7.

Таблиця Б.7 – Структура таблиці «profile»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
User_id	INT(11)	PK, NN	От user (id) до profile (user_id) FK	№ Користувача
name	VARCHAR(255)	Null		Ім'я
Public_name	VARCHAR(255)	Null		Публічне ім'я
Gravatar_email	VARCHAR(255)	Null		Аватар електронної адреси
Gravatar_id	VARCHAR(32)	Null		№ Аватара
location	VARCHAR(255)	Null		Розміщення
website	VARCHAR(255)	Null		Веб-сайт
bio	TEXT	Null		Персональна інформація
timezone	VARCHAR(255)	Null		Часова зона

У таблиці «user» зберігається інформація про користувача, а саме №, ім'я користувача, електронна адреса, хешування паролю, ключ аутентифікації, підтвердження, блокування, реєстрація, створення, оновлення та флаг, див. табл. Б.8.

Таблиця Б.8 – Структура таблиці «user»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От id до social_account(user-id), profile (user_id) FK	id

Продовження таблиці Б.8.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(255)	NN, UNIQUE		Ім'я користувача
email	VARCHAR(255)	Null, UNIQUE		Електронна адреса
Password_hash	VARCHAR(60)	NN		Хешування паролю
Auth_key	VARCHAR(32)	NN		Ключ аутентифікації
Confirmed_at	INT(11)	Null		Дозвіл
Blocked_at	VARCHAR(255)	Null		Блокування
Registration_ip	VARCHAR(45)	Null		Ір реєстрація
Created_at	INT(11)	NN		Створення
Updated_at	INT(11)	NN		Оновлення
flags	INT(11)	NN		Прапор

У таблиці «requirement» зберігається вимоги, а саме №, ім'я, опис, останні зміни, див. табл. Б.9.

Таблиця Б.9 – Структура таблиці «requirement»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN	От id до country_requirement(requirement_id)FK	id
name	VARCHAR(255)	NN		Ім'я
description	TEXT	NN		Опис
Last_change	IMESTAMP	NN		Фінальні зміни

У таблиці «country_requirement» зберігається інформація про вимоги до країни, а саме № країни, № вимоги, див. табл. Б.10.

Таблиця Б.10 – Структура таблиці «country_requirement»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Country_id	INT(11)	PK, NN	От country (id) до country_requirement(country_id)FK	Id Країни
Requirement_id	INT(11)	PK, NN	От requirement(id)	Id Вимог

У таблиці «country» зберігається інформація про країни, а саме №, ім'я, опис, див. табл. Б.11.

Таблиця Б.11 – Структура таблиці «country»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN	От id до country-requirement(country_id), offer(country_id), town(country_id), airport(country_id), deal(country_id)FK	id
name	VARCHAR(255)	NN		Ім'я
description	TEXT	NN		Опис

У таблиці «offer» зберігається інформація про пропозиції, а саме №, ім'я, опис, ціна, № країни, № міста, № готелю, останні зміни, продолжительность от, продолжительность до, малое описание, № типу, див. табл. Б.12

Таблица Б.12 – Структура таблиці «offer»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От id до deal_request(offer_id) , deal(offer_id)	id
name	VARCHAR(255)	NN		Ім'я
description	TEXT	NN		Опис
price	DOUBLE	NN		Ціна
Country_id	INT(11)	NN	От country(id)	Id Країни
Town_id	INT(11)	NN	От town(id)	Id Міста
Hotel_id	INT(11)	NN	От hotel(id)	Id Готеля
Last_change	TIMESTAMP	NN		Фінальний вибір
Duration_ from	DATA	NN		Тривалість от
Duration_to	DATA	NN		Тривалість до
Small_description	TEXT	NN		Короткий опис
Type_id	INT(11)	NN	От tour_type(id)	Id Тип

У таблиці «hotel» зберігається інформація про готелі, а саме №, назва, кількість зірок, телефон, електронна адреса, № міста, координати готеля перший варіант, координати готеля других варіант, див. табл. Б.13

Таблиця Б.13 – Структура таблиці «hotel»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От id до offer(hotel_id)	id
name	VARCHAR(255)	NN		Ім'я
stars	DOUBLE	NN		Зірки
phone	INT(11)	NN		Телефон
email	VARCHAR(255)	NN		Електронна адреса
Town_id	INT(11)	NN	От town(id)	Id Городу
Coord_grand_one	DOUBLE	NN		Координати готелю варіант 1
Coord_grand_two	DOUBLE	Null		Координати готелю варіант 2

У таблиці «town» зберігається інформація про міста, а саме №, назва, координати перший варіант, координати других варіант, опис, № країни, див. табл. Б.14.

Таблиця Б.14 – Структура таблиці «town»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От id до hotel(town_id), offer(rown_id), airport(town_id)FK	id
name	VARCHAR(255)	NN		Ім'я
Coord_grand_one	DOUBLE	NN		Координати готелю варіант 1

Продовження таблиці Б.14.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Coord_grand_two	DOUBLE	NN		Координати готелю варіант 2
description	TEXT	NN		Опис
Country_id	INT(11)	NN	От country(id)	Id Країна

У таблиці «tour_type» зберігається інформація тип туру, а саме №, ім'я, опис, див. табл. Б.15.

Таблиця Б.15 – Структура таблиці «tour_type»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От id до offer(type_id)	id
name	VARCHAR(255)	NN		Ім'я
description	TEXT	NN		Опис

У таблиці «deal_request» зберігається інформація про запит угоди, а саме №, ім'я, телефон, № пропозиції, створення, № користувача, див. табл. Б.16.

Таблиця Б.16 – Структура таблиці «deal_request»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	От offer(id) до deal_request(offer_id)	id
name	TEXT	NN		Ім'я
phone	INT(11)	NN		Телефон
Offer_id	INT(11)	NN	От offer(id)	Id Пропозиція

Продовження таблиці Б.16.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Created	TIMESTAMP	NN		Створення
User_id	INT(11)	NN		Id Користувача

У таблиці «deal» зберігається інформація про угоду, а саме №, № менеджера, № пропозиції, останні зміни, повне ім'я, телефон, з якого аеропорту політ, до якого аеропорту переліт, див. табл. Б.17.

Таблиця Б.17 – Структура таблиці «deal»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment		id
Manager_id	INT(11)	NN	От user(id)	Id Менеджера
Offer_id	INT(11)	NN	От offer(id)	Id Пропозиції
Last_change	TIMESTAMP	NN	От country(id)	Останні зміни
Full_name	TEXT	NN		Повне Ім'я
Phone	INT(11)	NN		Телефон
Airport_from_id	INT(11)	NN	От airport(id)	Id Аеропорта звідки
Airport_to_id	TIMESTAMP	NN	От airport(id)	Id Аеропорта куди
Country_id	INT(11)	NN	От country(id)	Id Країна
Flight_price	DOUBLE	NN		Ціна перельоту

У таблиці «airport» зберігається інформація аеропорти, а саме №, ім'я, № країни, № міста, див. табл. Б.18.

Таблиця Б.18 – Структура таблиці «airport»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto_increment	До deal(airport_from, airport_to)	id
name	VARCHAR(255)	NN		Ім'я
Country_id	INT(11)	NN		Id Країна
Town_id	INT(11)	NN		Id Місто

У таблиці «pta_column_info» зберігається інформація про конкретну таблицю, а саме №, ім'я бази, ім'я таблиці, ім'я стовпця, міме типи даних, трансформації, трансформаційних опцій, вивід опцій трансформацій, вивід трансформацій, див. табл. Б.19.

Таблиця Б.19 – Структура таблиці «pta_column_info»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(5)	PK, NN,		id
		Auto_increment, UNIQUE		
Db_name	VARCHAR(64)	NN		Ім'я бази
Table_name	VARCHAR(64)	NN		Ім'я таблиці
Column_name	VARCHAR(64)	NN		Ім'я стовпця
comment	VARCHAR(255)	NN		Коментарій
mimetype	VARCHAR(64)	NN		Міме тип даних
transformation	VARCHAR(255)	NN		Перетворення
Transformation_options	VARCHAR(255)	NN		Перетворення налаштувань
Input_transformation	VARCHAR(255)	NN		Вивід перетворень

Продовження таблиці Б.19.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Input_transformation_options	VARCHAR(255)	NN		Вивід перетворення налаштування

У таблиці «pma_table_coords» зберігається інформація про координати конкретно взятої таблиці, а саме ім'я бази, ім'я таблиці, нумерації pdf документа, координата x, координата y, див. табл. Б.20.

Таблиця Б.20 – Структура таблиці «pma_table_coords»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Table_name	VARCHAR(64)	PK, NN		Ім'я таблиці
Pdf_page_number	INT(11)	PK, NN		Нумерація сторінки pdf
X	FLOAT	NN, UNIQUE		Координата по X
Y	FLOAT	NN, UNIQUE		Координата по Y

У таблиці «pma_savedsearches» зберігається інформація про пошукові операції конкретного користувача, а саме №, ім'я користувача ім'я бази, ім'я пошуку, пошук даних, див. табл. Б.21.

Таблиця Б.21 – Структура таблиці «pma_savedsearches»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, UNIQUE. Auto increment		id

Продовження таблиці Б.21.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN, UNIQUE		Ім'я користувача
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Search_name	VARCHAR(64)	NN, UNIQUE		Пошук Ім'я
Search_data	TEXT	NN, UNIQUE		Пошук даних

У таблиці «rma_users» зберігається інформація про зв'язок користувача з групою, а саме №, ім'я користувача, див. табл. 8.22.

Таблиця Б.22 – Структура таблиці «rma_users»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	VARCHAR(64)	PK, NN		id
username	VARCHAR(64)	PK, NN		Ім'я користувача

У таблиці «rma_table_info» зберігається інформація про таблиці, а саме ім'я бази, ім'я таблиці, поле дисплею, див. табл. Б.22.

Таблиця Б.23 – Структура таблиці «rma_table_info»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Table_name	VARCHAR(64)	PK, NN		Ім'я таблиці
Display_field	VARCHAR(64)	NN		Поле дисплея

У таблиці «rma_ralation» зберігається інформація про встановленні зв'язки з існуючими зовнішніми базами даних, а саме мастер бази, мастер таблиці, мастер полю, зовнішня база, зовнішня таблиці, зовнішнє поле, див. табл. Б.24.

Таблиця Б.24 – Структура таблиці «pma_relation»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Master_db	VARCHAR(64)	PK, NN		Мастер бази
Master_table	VARCHAR(64)	PK, NN		Мастер таблиці
Master_field	VARCHAR(64)	PK, NN		Мастер отримання
Foreign_db	VARCHAR(64)			Зовнішня база
Foreign_table	VARCHAR(64)			Зовнішня таблиця
Foreign_field	VARCHAR(64)			Зовнішнє поле

У таблиці «pma_central_columns» зберігається інформація про особові важливі стовпці, а саме ім'я бази, ім'я стовпця, довжина стовпця, нулевой стовпець, зв'єрнення стовпця, стовпець екстра, стовпець за замовчуванням, див. табл. Б.25.

Таблиця Б.25 – Структура таблиці «pma_central_columns»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Col_name	VARCHAR(64)	PK, NN		Ім'я стовпця
Col_type	VARCHAR(64)	NN		Тип стовпця
Col_lenght	VARCHAR(64)	Null		Довжина стовпця
Col_collation	VARCHAR(64)	NN		Коллаж стовпця
Col_isNull	VARCHAR(64)	NN		Нулл стовпець
Col_extra		Null		Екстра стовпець

Продовження таблиці Б.25.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Col_default		Null		Столбец за замовчуванням

У таблиці «rma_recent» зберігається інформація ім'я користувача та ім'я таблиці з якою він взаємодіє, а саме ім'я користувача, таблиця, див. табл. Б.26.

Таблиця Б.26 – Структура таблиці «rma_recent»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN		Ім'я користувача
tables	TEXT	NN		Таблиця

У таблиці «action» зберігається інформація про лог дій, а саме №, ім'я, дату, № користувача, див. табл. Б.27.

Таблиця Б.27 – Структура таблиці «action»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN		id
name	VARCHAR(255)	NN		Ім'я
data	TIMESTAMP	NN		Дані
User_id	INT(11)	NN		Id Користувача

У таблиці «rma_bookmark» зберігається інформація про помітки, а саме №, база, користувач, помітка, запит, див. табл. Б.28.

Таблиця Б.28 – Структура таблиці «pma_bookmark»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto increment		id
dbase	VARCHAR(255)	NN		База
user	VARCHAR(255)	NN		Користувач
label	VARCHAR(255)	NN		Мітка
query	TEXT	NN		Запит

У таблиці «pma_table_uirprefs» зберігається інформація про коментар до конкретної таблиці в базі, а саме ім'я користувача, ім'я бази, ім'я таблиці, настройки, останні оновлення, див. табл. Б.29

Таблиця Б.29 – Структура таблиці «pma_table_uirprefs»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(255)	PK, NN		Ім'я користувача
Db_name	VARCHAR(255)	PK, NN		Ім'я бази
Table_name	VARCHAR(255)	PK, NN		Ім'я таблиці
prefs	VARCHAR(255)	NN		Настройки
Last_update	TEXT	NN		Останнє оновлення

У таблиці «pma_history» зберігається інформація про список виконаних sql запитів, а саме №, ім'я користувача, база, таблиця, часове значення, sql запит, див. табл. Б.30.

Таблиця Б.30 – Структура таблиці «pma_history»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	BIGINT(20)	PK, NN, Auto increment, UNIQUE		Ім'я користувача

Продовження таблиці Б.30.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	NN		Ім'я бази
db	VARCHAR(64)	NN		Ім'я таблиці
table	VARCHAR(64)	NN		Налаштування
timevalue	TIMESTAMP	NN		Останнє оновлення
sqlquery	TEXT	NN		Sql - запит

У таблиці «permission» зберігається інформація про назву повноважень, а саме №, ім'я, див. табл. Б.31.

Таблиця Б.31 – Структура таблиці «permission»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(11)	PK, NN, Auto increment		id
name	VARCHAR(255)	NN		Ім'я

У таблиці «pma_userconfig» зберігається інформація про містить інформацію параметрів користувача та дату кінцевого оновлення, а саме ім'я користувача, временні значення, параметри відомостей, див. табл. Б.32.

Таблиця Б.32 – Структура таблиці «pma_userconfig»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN		Ім'я користувача
timevalue	TIMESTAMP	NN		Тимчасове значення
Config_data	TEXT	NN		Параметри даних

У таблиці «pma_favorite» зберігається інформація про вподобання, а саме ім'я користувача, таблиця, див. табл. Б.33

Таблиця Б.33 – Структура таблиці «pma_favorite»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN		Ім'я користувача
tables	TEXT	NN		Таблиця

У таблиці «pma_pdf_pages» зберігається інформація про пагінацію, а саме ім'я бази, резерв сторінок, інформація о сторінках, див. табл. Б.34.

Таблиця Б.34 – Структура таблиці «pma_pdf_pages»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Page_nr	TEXT	NN		Резерв сторінок
Page_descr				Інформація про сторінки

У таблиці «pma_designer_setting» зберігається інформація про користувача та настройки, а саме ім'я користувача, параметри даних, див. табл. Б.35.

Таблиця Б.35 – Структура таблиці «pma_designer_setting»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN		Ім'я користувача
Setting_data	TEXT	NN		Параметри даних

У таблиці «pma_usergroups» зберігається інформація про групу користувачів та повноваження, а саме група користувача, вкладка, розміщення допущень, див. табл. Б.36.

Таблиця Б.36 – Структура таблиці «pma_usergroups»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
usergroup	VARCHAR(64)	PK, NN		Групи користувачів
tab	VARCHAR(64)	PK, NN		Вкладка
allowed	ENUM (Y, N)	PK, NN		Розміщення допущень

У таблиці «pma_export_templates» зберігається інформація про експорт моделей та конфігурації, а саме №, ім'я користувача, тип експорту, ім'я шаблону, дані шаблону, див. табл. Б.37.

Таблиця Б.37 – Структура таблиці «pma_export_templates»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
id	INT(5)	PK, NN, UNIQUE, Auto increment		id
username	VARCHAR(64)	NN		Ім'я користувача
Export_type	VARCHAR(10)	NN		Експортування типів
Template_name	VARCHAR(64)	NN		Ім'я шаблону
Template_data	TEXT	NN		Шаблон даних

У таблиці «pma_navigationhiding» зберігається інформація про те що закрито від користувача, а саме ім'я користувача, пункт імені, пункт типу, ім'я бази, ім'я таблиці, див. табл. Б.38.

Таблиця Б.38 – Структура таблиці «pma_navigationhiding»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
username	VARCHAR(64)	PK, NN		Ім'я користувача

Продовження таблиці Б.38.

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Item_name	VARCHAR(64)	PK, NN		Пункт імені
Item_type	VARCHAR(64)	PK, NN		Пункт типу
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Table_name	VARCHAR(64)	PK, NN		Ім'я таблиці

У таблиці «pma_tracking» зберігається інформація про зберігання активності бази, а саме ім'я бази, ім'я таблиці, версія, створення даних, оновлення даних, знімок схеми, дані sql, відстеження, активне відстеження, див. табл. Б.39.

Таблиця Б.39 – Структура таблиці «pma_tracking»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
Db_name	VARCHAR(64)	PK, NN		Ім'я бази
Table_name	VARCHAR(64)	PK, NN		Ім'я таблиці
version	INT(10)	PK, NN,		Версія
		UNIQUE		
Date_created	DATETIME	NN		Створення даних
Date_updated	DATETIME	NN		Оновлення даних
Schema_snapshot	TEXT	NN		Знімок схеми
Schema_sql	TEXT	Null		Схема sql
Data sql	LONGTEXT	Null		Дані sql
Tracking	SET(update,delete,create,replace,insert)	Null		Відстеження
Tracking_active	INT(1)	NN, UNIQUE		Активне відстеження

У таблиці «migration» зберігається інформація про лог міграцій, а саме версія, час застосування, див. табл. Б.40.

Таблиця Б.40 – Структура таблиці «migration»

Назва поля	Тип даних	Властивості	Зв'язок	Опис
version	VARCHAR(180)	PK, NN		Версія
Apply_time	INT(11)	NN		Час застосування

У розділі було представлено загальну структуру розробленої бази даних, яка є фундаментальною для забезпечення функціональності та безпеки WEB-додатку. Детальний огляд таблиць, таких як «auth_rule» та «auth_item», показує, як система управління правами користувачів та їх аутентифікацією є впорядкованою та гнучкою. Це дозволяє адміністраторам додатку легко налаштовувати доступ до ресурсів, контролювати повноваження користувачів та виконувати аудит безпеки системи.

Структура таблиць розроблена таким чином, щоб забезпечити необхідний рівень нормалізації з метою оптимізації запитів та зменшення ризику дублювання даних. Зв'язки між таблицями встановлені з урахуванням майбутнього масштабування системи. Такий підхід у проєктуванні бази даних забезпечує стабільну основу для подальшого розвитку та інтеграції з іншими системами та сервісами. Загалом, розроблена база даних є ключовим компонентом для забезпечення надійності, безпеки та ефективності WEB-додатку.

```
phpMyAdmin SQL Dump
-- version 4.6.5.2
-- https://www.phpmyadmin.net/
--
-- Хост: 127.0.0.1
-- Час створення: Жов 16 2023 г., 10:04
-- Версія сервера: 10.1.21-MariaDB
-- Версія PHP: 5.6.30
```

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
```

```
SET time_zone = "+00:00";
```

```
--
```

```
-- База даних: `travel_crm`
```

```
--
```

```
-----
```

```
--
```

```
-- Структура таблиці `action`
```

```
--
```

```
CREATE TABLE `action` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `airport`
```

```
--
```

```
CREATE TABLE `airport` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `country_id` int(11) NOT NULL,
  `town_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `auth_assignment`
```

```
--
```

```
CREATE TABLE `auth_assignment` (
  `item_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `user_id` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-----
```

```
--
```

```
-- Структура таблиці `auth_item`
```

```
--
```

```
CREATE TABLE `auth_item` (
  `name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `type` int(11) NOT NULL,
  `description` text COLLATE utf8_unicode_ci,
```

```

`rule_name` varchar(64) COLLATE utf8_unicode_ci DEFAULT NULL,
`data` text COLLATE utf8_unicode_ci,
`created_at` int(11) DEFAULT NULL,
`updated_at` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```
-----
```

```

--
-- Структура таблиці `auth_item_child`
--

```

```

CREATE TABLE `auth_item_child` (
  `parent` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `child` varchar(64) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```
-----
```

```

--
-- Структура таблиці `auth_rule`
--

```

```

CREATE TABLE `auth_rule` (
  `name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `data` text COLLATE utf8_unicode_ci,
  `created_at` int(11) DEFAULT NULL,
  `updated_at` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```
-----
```

```

--
-- Структура таблиці `country`
--

```

```

CREATE TABLE `country` (
  `id` int(11) NOT NULL,
  `name` varchar(255) CHARACTER SET utf8 NOT NULL,
  `description` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
-----
```

```

--
-- Структура таблиці `country_requirement`
--

```

```

CREATE TABLE `country_requirement` (
  `country_id` int(11) NOT NULL,
  `requirement_id` int(11) NOT NULL

```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `deal`
```

```
--
```

```
CREATE TABLE `deal` (
  `id` int(11) NOT NULL,
  `manager_id` int(11) NOT NULL,
  `offer_id` int(11) NOT NULL,
  `last_change` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `full_name` text NOT NULL,
  `phone` int(11) NOT NULL,
  `airport_from_id` int(11) NOT NULL,
  `airport_to_id` int(11) NOT NULL,
  `country_id` int(11) NOT NULL,
  `flight_price` double NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-----
```

```
--
```

```
-- Структура таблиці `deal_request`
```

```
--
```

```
CREATE TABLE `deal_request` (
  `id` int(11) NOT NULL,
  `name` text CHARACTER SET utf8 NOT NULL,
  `phone` int(11) NOT NULL,
  `offer_id` int(11) NOT NULL,
  `created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `user_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `group`
```

```
--
```

```
CREATE TABLE `group` (
  `id` int(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `base_salary` double NOT NULL,
  `permission_id` int(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
-- Структура таблиці `hotel`
--
```

```
CREATE TABLE `hotel` (
  `id` int(11) NOT NULL,
  `name` varchar(255) CHARACTER SET utf8 NOT NULL,
  `stars` double NOT NULL,
  `phone` int(11) NOT NULL,
  `email` varchar(255) NOT NULL,
  `town_id` int(11) NOT NULL,
  `coord_grad_one` double NOT NULL,
  `coord_grad_two` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
-- Структура таблиці `migration`
--
```

```
CREATE TABLE `migration` (
  `version` varchar(180) NOT NULL,
  `apply_time` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
-- Структура таблиці `offer`
--
```

```
CREATE TABLE `offer` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `description` text NOT NULL,
  `price` double NOT NULL,
  `country_id` int(11) NOT NULL,
  `town_id` int(11) NOT NULL,
  `hotel_id` int(11) NOT NULL,
  `last_change` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `duration_from` date NOT NULL,
  `duration_to` date NOT NULL,
  `small_description` text CHARACTER SET latin1 NOT NULL,
  `type_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-----
```

```
--
```

```
-- Структура таблиці `permission`
```

```
--
```

```
CREATE TABLE `permission` (  
  `id` int(11) NOT NULL,  
  `name` varchar(255) NOT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `profile`
```

```
--
```

```
CREATE TABLE `profile` (  
  `user_id` int(11) NOT NULL,  
  `name` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `public_email` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `gravatar_email` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `gravatar_id` varchar(32) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `location` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `website` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,  
  `bio` text COLLATE utf8_unicode_ci,  
  `timezone` varchar(40) COLLATE utf8_unicode_ci DEFAULT NULL  
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-----
```

```
--
```

```
-- Структура таблиці `requirement`
```

```
--
```

```
CREATE TABLE `requirement` (  
  `id` int(11) NOT NULL,  
  `name` varchar(255) NOT NULL,  
  `description` text NOT NULL,  
  `last_change` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
-----
```

```
--
```

```
-- Структура таблиці `social_account`
```

```
--
```

```
CREATE TABLE `social_account` (  
  `id` int(11) NOT NULL,  
  `user_id` int(11) DEFAULT NULL,  
  `provider` varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  `client_id` varchar(255) COLLATE utf8_unicode_ci NOT NULL,  
  `data` text COLLATE utf8_unicode_ci,
```

```

`code` varchar(32) COLLATE utf8_unicode_ci DEFAULT NULL,
`created_at` int(11) DEFAULT NULL,
`email` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
`username` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```
-----
```

```
--
```

```
-- Структура таблиці `token`
```

```
--
```

```

CREATE TABLE `token` (
  `user_id` int(11) NOT NULL,
  `code` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  `created_at` int(11) NOT NULL,
  `type` smallint(6) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```
-----
```

```
--
```

```
-- Структура таблиці `tour_type`
```

```
--
```

```

CREATE TABLE `tour_type` (
  `id` int(11) NOT NULL,
  `name` varchar(255) CHARACTER SET utf8 NOT NULL,
  `description` text CHARACTER SET utf8 NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
-----
```

```
--
```

```
-- Структура таблиці `town`
```

```
--
```

```

CREATE TABLE `town` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `coord_grad_one` double NOT NULL,
  `coord_grad_two` double NOT NULL,
  `description` text NOT NULL,
  `country_id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

```
-----
```

```
--
```

```
-- Структура таблиці `user`
```

```
--
```

```

CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `username` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password_hash` varchar(60) COLLATE utf8_unicode_ci NOT NULL,
  `auth_key` varchar(32) COLLATE utf8_unicode_ci NOT NULL,
  `confirmed_at` int(11) DEFAULT NULL,
  `unconfirmed_email` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `blocked_at` int(11) DEFAULT NULL,
  `registration_ip` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` int(11) NOT NULL,
  `updated_at` int(11) NOT NULL,
  `flags` int(11) NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

```

```

--
-- Індекси збережених таблиць
--

```

```

--
-- Індекси таблиці `action`
--
ALTER TABLE `action`
  ADD PRIMARY KEY (`id`);

```

```

--
-- Індекси таблиці `airport`
--
ALTER TABLE `airport`
  ADD PRIMARY KEY (`id`),
  ADD KEY `country_id` (`country_id`),
  ADD KEY `town_id` (`town_id`);

```

```

--
-- Індекси таблиці `auth_assignment`
--
ALTER TABLE `auth_assignment`
  ADD PRIMARY KEY (`item_name`,`user_id`);

```

```

--
-- Індекси таблиці `auth_item`
--
ALTER TABLE `auth_item`
  ADD PRIMARY KEY (`name`),
  ADD KEY `rule_name` (`rule_name`),
  ADD KEY `idx-auth_item-type` (`type`);

```

```

--
-- Індекси таблиці `auth_item_child`
--

```

```

ALTER TABLE `auth_item_child`
  ADD PRIMARY KEY (`parent`,`child`),
  ADD KEY `child` (`child`);

--
-- Індекси таблиці `auth_rule`
--
ALTER TABLE `auth_rule`
  ADD PRIMARY KEY (`name`);

--
-- Індекси таблиці `country`
--
ALTER TABLE `country`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `country_requirement`
--
ALTER TABLE `country_requirement`
  ADD PRIMARY KEY (`country_id`,`requirement_id`),
  ADD KEY `requirement_id` (`requirement_id`);

--
-- Індекси таблиці `deal`
--
ALTER TABLE `deal`
  ADD PRIMARY KEY (`id`),
  ADD KEY `offer_id` (`offer_id`),
  ADD KEY `manager_id` (`manager_id`),
  ADD KEY `airport_from_id` (`airport_from_id`),
  ADD KEY `airport_to_id` (`airport_to_id`),
  ADD KEY `country_id` (`country_id`);

--
-- Індекси таблиці `deal_request`
--
ALTER TABLE `deal_request`
  ADD PRIMARY KEY (`id`),
  ADD KEY `deal_request_ibfk_1` (`offer_id`);

--
-- Індекси таблиці `group`
--
ALTER TABLE `group`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `hotel`
--
ALTER TABLE `hotel`

```

```

ADD PRIMARY KEY (`id`),
ADD KEY `town_id` (`town_id`);

--
-- Індекси таблиці `migration`
--
ALTER TABLE `migration`
  ADD PRIMARY KEY (`version`);

--
-- Індекси таблиці `offer`
--
ALTER TABLE `offer`
  ADD PRIMARY KEY (`id`),
  ADD KEY `offer_ibfk_2` (`country_id`),
  ADD KEY `offer_ibfk_3` (`town_id`),
  ADD KEY `offer_ibfk_4` (`hotel_id`),
  ADD KEY `type_id` (`type_id`);

--
-- Індекси таблиці `permission`
--
ALTER TABLE `permission`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `profile`
--
ALTER TABLE `profile`
  ADD PRIMARY KEY (`user_id`);

--
-- Індекси таблиці `requirement`
--
ALTER TABLE `requirement`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `social_account`
--
ALTER TABLE `social_account`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `account_unique` (`provider`,`client_id`),
  ADD UNIQUE KEY `account_unique_code` (`code`),
  ADD KEY `fk_user_account` (`user_id`);

--
-- Індекси таблиці `token`
--
ALTER TABLE `token`
  ADD UNIQUE KEY `token_unique` (`user_id`,`code`,`type`);

```

```

--
-- Індекси таблиці `tour_type`
--
ALTER TABLE `tour_type`
  ADD PRIMARY KEY (`id`);

--
-- Індекси таблиці `town`
--
ALTER TABLE `town`
  ADD PRIMARY KEY (`id`),
  ADD KEY `town_ibfk_1` (`country_id`);

--
-- Індекси таблиці `user`
--
ALTER TABLE `user`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `user_unique_email` (`email`),
  ADD UNIQUE KEY `user_unique_username` (`username`);

--
-- AUTO_INCREMENT для збережених таблиць
--

--
-- AUTO_INCREMENT для таблиці `action`
--
ALTER TABLE `action`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;

--
-- AUTO_INCREMENT для таблиці `airport`
--
ALTER TABLE `airport`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT для таблиці `country`
--
ALTER TABLE `country`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT для таблиці `deal`
--
ALTER TABLE `deal`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

--
-- AUTO_INCREMENT для таблиці `deal_request`
--
ALTER TABLE `deal_request`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

```

```

--
-- AUTO_INCREMENT для таблиці `group`
--
ALTER TABLE `group`
  MODIFY `id` int(255) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблиці `hotel`
--
ALTER TABLE `hotel`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT для таблиці `offer`
--
ALTER TABLE `offer`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=6;
--
-- AUTO_INCREMENT для таблиці `permission`
--
ALTER TABLE `permission`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблиці `requirement`
--
ALTER TABLE `requirement`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT для таблиці `social_account`
--
ALTER TABLE `social_account`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
--
-- AUTO_INCREMENT для таблиці `tour_type`
--
ALTER TABLE `tour_type`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
--
-- AUTO_INCREMENT для таблиці `town`
--
ALTER TABLE `town`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
--
-- AUTO_INCREMENT для таблиці `user`
--
ALTER TABLE `user`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=7;
--
-- Обмеження зовнішнього ключа збережених таблиць
--
--
-- Обмеження зовнішнього ключа таблиці `airport`

```

```

--
ALTER TABLE `airport`
  ADD CONSTRAINT `airport_ibfk_1` FOREIGN KEY (`country_id`) REFERENCES
`country` (`id`),
  ADD CONSTRAINT `airport_ibfk_2` FOREIGN KEY (`town_id`) REFERENCES `town`
(`id`);

--
-- Обмеження зовнішнього ключа таблиці `auth_assignment`
--
ALTER TABLE `auth_assignment`
  ADD CONSTRAINT `auth_assignment_ibfk_1` FOREIGN KEY (`item_name`)
REFERENCES `auth_item` (`name`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `auth_item`
--
ALTER TABLE `auth_item`
  ADD CONSTRAINT `auth_item_ibfk_1` FOREIGN KEY (`rule_name`) REFERENCES
`auth_rule` (`name`) ON DELETE SET NULL ON UPDATE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `auth_item_child`
--
ALTER TABLE `auth_item_child`
  ADD CONSTRAINT `auth_item_child_ibfk_1` FOREIGN KEY (`parent`) REFERENCES
`auth_item` (`name`) ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `auth_item_child_ibfk_2` FOREIGN KEY (`child`) REFERENCES
`auth_item` (`name`) ON DELETE CASCADE ON UPDATE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `country_requirement`
--
ALTER TABLE `country_requirement`
  ADD CONSTRAINT `country_requirement_ibfk_1` FOREIGN KEY (`country_id`)
REFERENCES `country` (`id`),
  ADD CONSTRAINT `country_requirement_ibfk_2` FOREIGN KEY (`requirement_id`)
REFERENCES `requirement` (`id`);

--
-- Обмеження зовнішнього ключа таблиці `deal`
--
ALTER TABLE `deal`
  ADD CONSTRAINT `deal_ibfk_1` FOREIGN KEY (`offer_id`) REFERENCES `offer` (`id`),
  ADD CONSTRAINT `deal_ibfk_2` FOREIGN KEY (`manager_id`) REFERENCES `user`
(`id`),
  ADD CONSTRAINT `deal_ibfk_3` FOREIGN KEY (`airport_from_id`) REFERENCES
`airport` (`id`),
  ADD CONSTRAINT `deal_ibfk_4` FOREIGN KEY (`airport_to_id`) REFERENCES `airport`
(`id`),

```

```

ADD CONSTRAINT `deal_ibfk_5` FOREIGN KEY (`country_id`) REFERENCES `country`
(`id`);

--
-- Обмеження зовнішнього ключа таблиці `deal_request`
--
ALTER TABLE `deal_request`
  ADD CONSTRAINT `deal_request_ibfk_1` FOREIGN KEY (`offer_id`) REFERENCES
`offer` (`id`) ON DELETE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `hotel`
--
ALTER TABLE `hotel`
  ADD CONSTRAINT `hotel_ibfk_1` FOREIGN KEY (`town_id`) REFERENCES `town`
(`id`);

--
-- Обмеження зовнішнього ключа таблиці `offer`
--
ALTER TABLE `offer`
  ADD CONSTRAINT `offer_ibfk_2` FOREIGN KEY (`country_id`) REFERENCES `country`
(`id`),
  ADD CONSTRAINT `offer_ibfk_3` FOREIGN KEY (`town_id`) REFERENCES `town`
(`id`),
  ADD CONSTRAINT `offer_ibfk_4` FOREIGN KEY (`hotel_id`) REFERENCES `hotel`
(`id`),
  ADD CONSTRAINT `offer_ibfk_5` FOREIGN KEY (`type_id`) REFERENCES `tour_type`
(`id`);

--
-- Обмеження зовнішнього ключа таблиці `profile`
--
ALTER TABLE `profile`
  ADD CONSTRAINT `fk_user_profile` FOREIGN KEY (`user_id`) REFERENCES `user`
(`id`) ON DELETE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `social_account`
--
ALTER TABLE `social_account`
  ADD CONSTRAINT `fk_user_account` FOREIGN KEY (`user_id`) REFERENCES `user`
(`id`) ON DELETE CASCADE;

--
-- Обмеження зовнішнього ключа таблиці `token`
--
ALTER TABLE `token`
  ADD CONSTRAINT `fk_user_token` FOREIGN KEY (`user_id`) REFERENCES `user` (`id`)
ON DELETE CASCADE;

```

```
--  
-- Обмеження зовнішнього ключа таблиці `town`  
--  
ALTER TABLE `town`  
  ADD CONSTRAINT `town_ibfk_1` FOREIGN KEY (`country_id`) REFERENCES `country`  
  (`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

ДОДАТОК В
Текст програми

AirportController.php

```
<?php
```

```
namespace app\controllers;
```

```
use Yii;
use app\models\Airport;
use app\models\Offer;
use app\models\AirportSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\filters\AccessRule;
```

```
/**
 * AirportController implements the CRUD actions for Airport model.
 */
class AirportController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create', 'index', 'update', 'delete', 'lists', 'showlist'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }

    /**
     * Lists all Airport models.
     * @return mixed
     */
}
```

```

*/
public function actionIndex()
{
    $searchModel = new AirportSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

/**
 * Displays a single Airport model.
 * @param integer $id
 * @return mixed
 */
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new Airport model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new Airport();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing Airport model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

```

```

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing Airport model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the Airport model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $id
 * @return Airport the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */

public function actionLists($id) {
    $countAirports = Airport::find()
        ->where(['country_id' => $id])
        ->count();

    $Airports = Airport::find()
        ->where(['country_id' => $id])
        ->all();

    if($countAirports>0){
        echo "<option>click</option>";
        foreach ($Airports as $Airport) {
            echo "<option value='".$Airport->id.">".$Airport->name."</option>";
        }
    }else{
        echo "<option>-</option>";
    }
}

```

```

public function actionShowlist($id) {
    $offer_model = Offer::findOne($id);

    $countAirports = Airport::find()
        ->where(['country_id' => $offer_model->country_id])
        ->count();

    $Airports = Airport::find()
        ->where(['country_id' => $offer_model->country_id])
        ->all();

    if($countAirports>0){
        echo "<option>click</option>";
        foreach ($Airports as $Airport) {
            echo "<option value='". $Airport->id. "'>". $Airport->name. "</option>";
        }
    }else{
        echo "<option>-</option>";
    }
}

protected function findModel($id)
{
    if (($model = Airport::findOne($id)) !== null) {
        return $model;
    } else {
        throw new NotFoundHttpException("The requested page does not exist.");
    }
}
}
}

-----
CountryController.php
<?php

namespace app\controllers;

use Yii;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\web\MethodNotAllowedHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\filters\AccessRule;

use app\models\Country;
use app\models\CountrySearch;

class CountryController extends \yii\web\Controller
{

```

```

public function behaviors()
{
    return [
        'access' => [
            'class' => AccessControl::className(),
            'ruleConfig' => [
                'class' => AccessRule::className(),
            ],
            'rules' => [
                [
                    'actions' => ['create','index','update','delete'],
                    'allow' => true,
                    'roles' => ['admin'],
                ],
                [
                    'actions' => [ 'view', 'search'],
                    'allow' => true,
                    'roles' => ['?', '@', 'admin'],
                ],
            ],
        ],
    ];
}

public function actionIndex()
{
    $searchModel = new CountrySearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

public function actionView($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

```

```

public function actionCreate()
{
    $model = new Country();

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

public function actionDelete($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

protected function findModel($id)
{
    if (($model = Country::findOne($id)) !== null) {
        return $model;
    }
}

```

```

        } else {
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
}

```

CountryRequirementController.php
<?php

```

namespace app\controllers;

use Yii;
use app\models\CountryRequirement;
use app\models\CountryRequirementSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

use yii\filters\AccessControl;
use yii\filters\AccessRule;

/**
 * CountryRequirementController implements the CRUD actions for CountryRequirement
 model.
 */
class CountryRequirementController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create','index','update','delete','lists'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }
}

```

```

        ],
    ],
];
}

/**
 * Lists all CountryRequirement models.
 * @return mixed
 */
public function actionIndex()
{
    $searchModel = new CountryRequirementSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

/**
 * Displays a single CountryRequirement model.
 * @param integer $country_id
 * @param integer $requirement_id
 * @return mixed
 */
public function actionView($country_id, $requirement_id)
{
    return $this->render('view', [
        'model' => $this->findModel($country_id, $requirement_id),
    ]);
}

/**
 * Creates a new CountryRequirement model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new CountryRequirement();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'country_id' => $model->country_id, 'requirement_id' =>
$model->requirement_id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

```

```

}

/**
 * Updates an existing CountryRequirement model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $country_id
 * @param integer $requirement_id
 * @return mixed
 */
public function actionUpdate($country_id, $requirement_id)
{
    $model = $this->findModel($country_id, $requirement_id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'country_id' => $model->country_id, 'requirement_id' =>
$model->requirement_id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing CountryRequirement model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $country_id
 * @param integer $requirement_id
 * @return mixed
 */
public function actionDelete($country_id, $requirement_id)
{
    $this->findModel($country_id, $requirement_id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the CountryRequirement model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $country_id
 * @param integer $requirement_id
 * @return CountryRequirement the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($country_id, $requirement_id)
{
    if (($model = CountryRequirement::findOne(['country_id' => $country_id, 'requirement_id'
=> $requirement_id])) !== null) {
        return $model;
    } else {

```

```

        throw new NotFoundException('The requested page does not exist.');
```

```

    }
}
}

```

DealController.php

```
<?php
```

```
namespace app\controllers;
```

```
use Yii;
```

```
use app\models\Deal;
```

```
use app\models\DealSearch;
```

```
use yii\helpers\ArrayHelper;
```

```
use app\models\Requirement;
```

```
use app\models\RequirementSearch;
```

```
use app\models\CountryRequirement;
```

```
use app\models\CountryRequirementSearch;
```

```
use yii\web\Controller;
```

```
use yii\web\NotFoundException;
```

```
use yii\filters\VerbFilter;
```

```
use mPDF;
```

```
use yii\filters\AccessControl;
```

```
use yii\filters\AccessRule;
```

```
/**
```

```
 * DealController implements the CRUD actions for Deal model.
```

```
 */
```

```
class DealController extends Controller
```

```
{
```

```
    /**
```

```
     * @inheritdoc
```

```
     */
```

```
    public function behaviors()
```

```
    {
```

```
        return [
```

```
            'access' => [
```

```
                'class' => AccessControl::className(),
```

```
                'ruleConfig' => [
```

```
                    'class' => AccessRule::className(),
```

```
                ],
```

```
            'rules' => [
```

```
                [
```

```
                    'actions' => ['create', 'index', 'update', 'delete', 'report'],
```

```
                    'allow' => true,
```

```
                    'roles' => ['admin'],
```

```

        ],
        [
            'actions' => [ 'view', 'search'],
            'allow' => true,
            'roles' => ['?', '@', 'admin'],
        ],
    ],
];
}

/**
 * Lists all Deal models.
 * @return mixed
 */
public function actionIndex()
{
    $searchModel = new DealSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

public function actionReport( $id ) {
    // get your HTML raw content without any layouts or scripts
    $deal_model = Deal::findOne($id);
    $requirement_array = CountryRequirement::find()->where(['country_id'=>$deal_model-
>offer->country->id])->asArray()->one();

    $requirement_data = Requirement::findOne($id);

    $list = " ";

    $list .= "<ul>";

    foreach($requirement_array as $requirement) {

        $requirement_data = Requirement::findOne($requirement);
        $list .= "<li>".$requirement_data->name."</li>";
    }

    $list .= "</ul>";

    $mpdf = new mPDF();

    $mpdf->WriteHTML("
        .container{
            text-align: center;

```

```

    }

    ",1);
    $mpdf->WriteHTML("
        <div class='container'>
            <h3>ДОГОВІР ПРО НАДАННЯ ТУРИСТИЧНИХ ПОСЛУГ</h3>
        </div>
        <p>ПІБ клієнта: $deal_model->full_name</p>
        <p>Телефон: ".$deal_model->phone."</p>
        <p>Послуга: ".$deal_model->offer->name."</p>
        <p>Країна призначення: ".$deal_model->offer->country->name."</p>
        <p>Місто призначення: ".$deal_model->offer->town->name."</p>
        <p>Готель: ".$deal_model->offer->hotel->name."</p>
        <p>Вимоги: ".$list."</p>
        <p>Прибуття: ".$deal_model->offer->duration_from.", від'їзд: ".$deal_model-
>offer->duration_to."</p>
        <p>Виліт с Аеропорту: ".$deal_model->airportFrom->name.", в аеропорт:
".$deal_model->airportTo->name."</p>
        <p>Ціна туру: ".$deal_model->offer->price." +ціна перельоту: ".$deal_model-
>flight_price."</p>
        <p>Підпис клієнта: _____</p>
        <p>Підпис консультанта: _____ </p>

    ");
    $mpdf->Output();

}

/**
 * Displays a single Deal model.
 * @param integer $id
 * @return mixed
 */
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new Deal model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new Deal();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }
}

```

```

    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing Deal model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing Deal model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the Deal model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $id
 * @return Deal the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    if (($model = Deal::findOne($id)) !== null) {
        return $model;
    } else {

```



```

    }

    /**
     * Lists all DealRequest models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new DealRequestSearch();

        if(!Yii::$app->user->identity->isAdmin){
            $dataProvider = new ActiveDataProvider([
                'query' => DealRequest::find()->where('user_id =' . Yii::$app->user->identity-
>getId()->orderBy('created DESC'),
                'pagination' => [
                    'forcePageParam' => false,
                    'pageSizeParam' => false,
                    'pageSize' => 10
                ]
            ]);
        }else{
            $num = -1;
            $dataProvider = new ActiveDataProvider([
                'query' => DealRequest::find()->where('user_id >' . $num)->orderBy('created DESC'),
                'pagination' => [
                    'forcePageParam' => false,
                    'pageSizeParam' => false,
                    'pageSize' => 10
                ]
            ]);
        }

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single DealRequest model.
     * @param integer $id
     * @return mixed
     */
    public function actionView($id)
    {
        return $this->render('view', [
            'model' => $this->findModel($id),
        ]);
    }

```

```

/**
 * Creates a new DealRequest model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new DealRequest();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing DealRequest model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing DealRequest model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**

```



```

        'allow' => true,
        'roles' => ['?', '@', 'admin'],
    ],
],
];
}

public function actionIndex()
{
    $searchModel = new HotelSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

public function actionView($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

public function actionCreate()
{
    $model = new Hotel();

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

```

```

public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

public function actionDelete($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

public function actionLists($id) {
    $countHotels = Hotel::find()
        ->where(['town_id' => $id])
        ->count();

    $Hotels = Hotel::find()
        ->where(['town_id' => $id])
        ->all();

    if($countHotels>0){
        foreach ($Hotels as $Hotel) {
            echo "<option value='".$Hotel->id."'>".$Hotel->name."</option>";
        }
    }else{
        echo "<option>-</option>";
    }
}

protected function findModel($id)

```

```

    {
        if (($model = Hotel::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundException('The requested page does not exist.');
        }
    }
}

```

 ManagerController.php
 <?php

```

namespace app\controllers;
use yii\filters\AccessControl;
use yii\filters\AccessRule;

```

```

class ManagerController extends \yii\web\Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create', 'index', 'update', 'delete'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }

    public function actionIndex()
    {
        return $this->render('index');
    }
}

```

 OfferController.php

```

<?php

namespace app\controllers;

use Yii;
use app\models\Offer;
use app\models\OfferSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

use yii\filters\AccessControl;
use yii\filters\AccessRule;

/**
 * OfferController implements the CRUD actions for Offer model.
 */
class OfferController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create','index','update','delete','lists'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }

    /**
     * Lists all Offer models.
     * @return mixed
     */
    public function actionIndex()

```

```

{
    $searchModel = new OfferSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

/**
 * Displays a single Offer model.
 * @param integer $id
 * @return mixed
 */
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new Offer model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new Offer();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing Offer model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {

```

```

        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing Offer model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    try {
        $this->findModel($id)->delete();
    }

    catch(\Exception $e){
        throw new \yii\web\HttpException(405, 'Error deleting offer, check dependencies');
    }
    return $this->redirect(['index']);
}

/**
 * Finds the Offer model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $id
 * @return Offer the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    {
        if (($model = Offer::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
}
}

-----
RequirementController.php
<?php

namespace app\controllers;

use Yii;
use app\models\Requirement;
use app\models\RequirementSearch;

```

```

use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

use yii\filters\AccessControl;
use yii\filters\AccessRule;

/**
 * RequirementController implements the CRUD actions for Requirement model.
 */
class RequirementController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create','index','update','delete','lists'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }

    /**
     * Lists all Requirement models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new RequirementSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
}

```

```

    });
}

/**
 * Displays a single Requirement model.
 * @param integer $id
 * @return mixed
 */
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new Requirement model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new Requirement();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing Requirement model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

```

```
/**
 * Deletes an existing Requirement model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

/**
 * Finds the Requirement model based on its primary key value.
 * If the model is not found, a 404 HTTP exception will be thrown.
 * @param integer $id
 * @return Requirement the loaded model
 * @throws NotFoundHttpException if the model cannot be found
 */
protected function findModel($id)
{
    if (($model = Requirement::findOne($id)) !== null) {
        return $model;
    } else {
        throw new NotFoundHttpException("The requested page does not exist.");
    }
}
}
}
-----
SiteController.php
<?php

namespace app\controllers;

use Yii;
use yii\filters\AccessControl;
use yii\web\Controller;
use yii\filters\VerbFilter;
use app\models\LoginForm;
use app\models>ContactForm;
use yii\data\ActiveDataProvider;
use yii\web\NotFoundHttpException;
use app\models\Offer;

class SiteController extends Controller
{
    /**
     * @inheritdoc

```

```

*/
public function behaviors()
{
    return [
        'access' => [
            'class' => AccessControl::className(),
            'only' => ['logout'],
            'rules' => [
                [
                    'actions' => ['logout'],
                    'allow' => true,
                    'roles' => ['@'],
                ],
            ],
        ],
        'verbs' => [
            'class' => VerbFilter::className(),
            'actions' => [
                'logout' => ['post'],
            ],
        ],
    ];
}

/**
 * @inheritdoc
 */
public function actions()
{
    return [
        'error' => [
            'class' => 'yii\web\ErrorAction',
        ],
        'captcha' => [
            'class' => 'yii\captcha\CaptchaAction',
            'fixedVerifyCode' => YII_ENV_TEST ? 'testme' : null,
        ],
    ];
}

/**
 * Displays homepage.
 *
 * @return string
 */
public function actionIndex()
{
    $id = Yii::$app->request->get('id');

    $dataProvider = new ActiveDataProvider([
        'query' => Offer::find()->orderBy('last_change DESC'),
    ]);

```

```

        'pagination' => [
            'forcePageParam' => false,
            'pageSizeParam' => false,
            'pageSize' => 5
        ]
    });

    if (empty($dataProvider)) throw new HttpException(404, 'Not found!');

    return $this->render('index', ['dataProvider' => $dataProvider]);
}

public function actionShowdetail() {

    $id = Yii::$app->request->get('id');
    $model = Offer::findOne($id);
    if (empty($model)) throw new HttpException(404, 'Not found!');

    return $this->render('showdetail', ['model' => $model ]);
}

/**
 * Login action.
 *
 * @return string
 */
public function actionLogin()
{
    if (!Yii::$app->user->isGuest) {
        return $this->goHome();
    }

    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        return $this->goBack();
    }
    return $this->render('login', [
        'model' => $model,
    ]);
}

/**
 * Logout action.
 *
 * @return string
 */
public function actionLogout()
{
    Yii::$app->user->logout();
}

```

```

        return $this->goHome();
    }

    /**
     * Displays contact page.
     *
     * @return string
     */
    public function actionContact()
    {
        $model = new ContactForm();
        if ($model->load(Yii::$app->request->post()) && $model->contact(Yii::$app->params['adminEmail'])) {
            Yii::$app->session->setFlash('contactFormSubmitted');

            return $this->refresh();
        }
        return $this->render('contact', [
            'model' => $model,
        ]);
    }

    /**
     * Displays about page.
     *
     * @return string
     */
    public function actionAbout()
    {
        return $this->render('about');
    }
}

```

```

-----
TourTypeController.php
<?php

```

```

namespace app\controllers;

```

```

use Yii;
use app\models\TourType;
use app\models\TourTypeSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;

```

```

use yii\filters\AccessControl;
use yii\filters\AccessRule;

```

```

/**

```

```

 * TourTypeController implements the CRUD actions for TourType model.

```

```

*/
class TourTypeController extends Controller
{
    /**
     * @inheritdoc
     */
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [
                    [
                        'actions' => ['create', 'index', 'update', 'delete', 'lists', 'showlist'],
                        'allow' => true,
                        'roles' => ['admin'],
                    ],
                    [
                        'actions' => ['view', 'search'],
                        'allow' => true,
                        'roles' => ['?', '@', 'admin'],
                    ],
                ],
            ],
        ];
    }

    /**
     * Lists all TourType models.
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new TourTypeSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Displays a single TourType model.
     * @param integer $id
     * @return mixed
     */
    public function actionView($id)

```

```

{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * Creates a new TourType model.
 * If creation is successful, the browser will be redirected to the 'view' page.
 * @return mixed
 */
public function actionCreate()
{
    $model = new TourType();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

/**
 * Updates an existing TourType model.
 * If update is successful, the browser will be redirected to the 'view' page.
 * @param integer $id
 * @return mixed
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

/**
 * Deletes an existing TourType model.
 * If deletion is successful, the browser will be redirected to the 'index' page.
 * @param integer $id
 * @return mixed
 */
public function actionDelete($id)
{

```

```

        $this->findModel($id)->delete();

        return $this->redirect(['index']);
    }

    /**
     * Finds the TourType model based on its primary key value.
     * If the model is not found, a 404 HTTP exception will be thrown.
     * @param integer $id
     * @return TourType the loaded model
     * @throws NotFoundHttpException if the model cannot be found
     */
    protected function findModel($id)
    {
        if (($model = TourType::findOne($id)) !== null) {
            return $model;
        } else {
            throw new NotFoundHttpException('The requested page does not exist.');
        }
    }
}

```

```

-----
TownController.php
<?php

namespace app\controllers;

use Yii;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\web\MethodNotAllowedHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\filters\AccessRule;

use app\models\Town;
use app\models\TownSearch;

class TownController extends \yii\web\Controller
{
    public function behaviors()
    {
        return [
            'access' => [
                'class' => AccessControl::className(),
                'ruleConfig' => [
                    'class' => AccessRule::className(),
                ],
                'rules' => [

```

```

        [
            'actions' => ['create','index','update','delete','lists'],
            'allow' => true,
            'roles' => ['admin'],
        ],
        [
            'actions' => [ 'view', 'search'],
            'allow' => true,
            'roles' => ['?', '@', 'admin'],
        ],
    ],
],
];
}

public function actionIndex()
{
    $searchModel = new TownSearch();
    $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('index', [
        'searchModel' => $searchModel,
        'dataProvider' => $dataProvider,
    ]);
}

public function actionView($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

public function actionCreate()
{
    $model = new Town();

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {

```

```

        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}

public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('update', [
            'model' => $model,
        ]);
    }
}

public function actionDelete($id)
{
    if( Yii::$app->user->isGuest ){
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}

public function actionLists($id) {
    $countTowns = Town::find()
        ->where(['country_id' => $id])
        ->count();

    $Towns = Town::find()
        ->where(['country_id' => $id])
        ->all();

    if($countTowns>0){
        echo "<option>click</option>";
        foreach ($Towns as $Town) {
            echo "<option value='".$Town->id."'>".$Town->name."</option>";
        }
    }
}

```

```
    }else{
        echo "<option>-</option>";
    }

}

protected function findModel($id)
{
    if (($model = Town::findOne($id)) !== null) {
        return $model;
    } else {
        throw new NotFoundException('The requested page does not exist.');
```

ДОДАТОК Г

Контекстні діаграми, діаграми декомпозиції, дерева вузлів, діаграми послідовності, діаграма BPMN 2.0, діаграми потоків, IDEF3



Рисунок Г.1 – Контекстна діаграма функціонування фірми

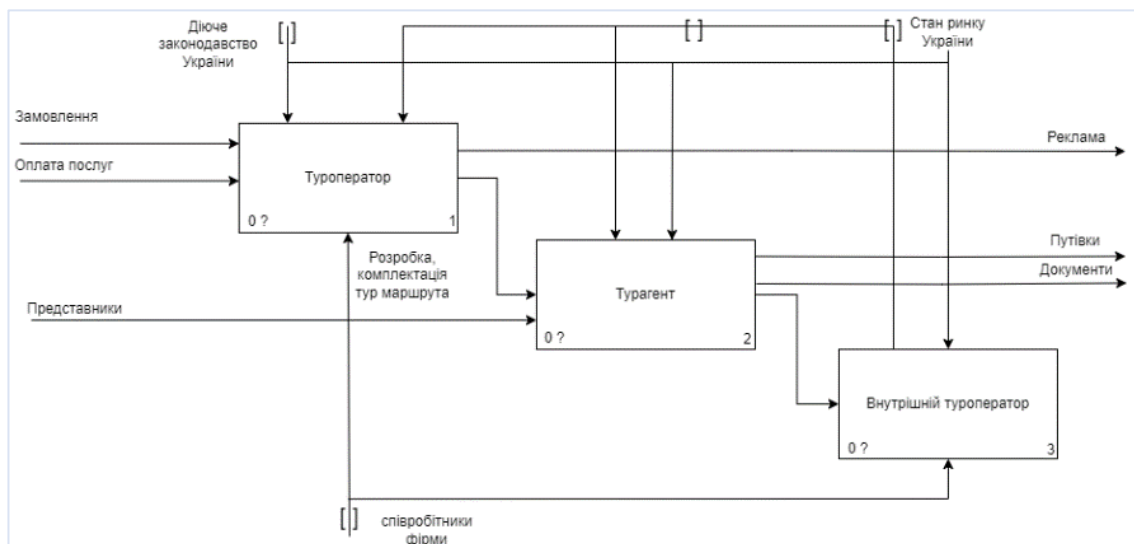


Рисунок Г.2 – Діаграма декомпозиції IDEF0 Автоматизація послуг «Туристичної фірми» Модель 1. Основні відділи фірми і їх робота. Початкова модель (AS-IS)

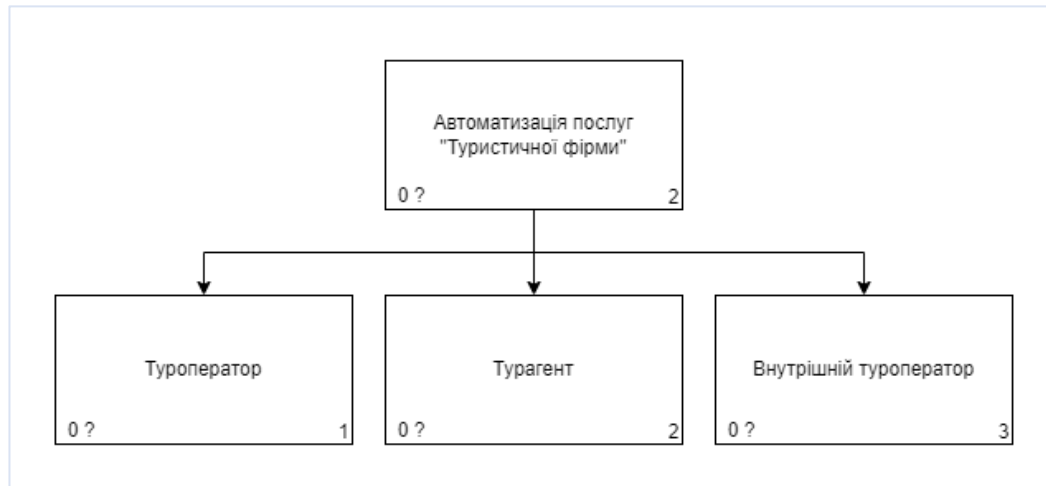


Рисунок Г.3 – Діаграма дерева вузлів Модель 1. Основні відділи фірми і їх робота. Початкова модель (AS-IS)



Рисунок Г.5 – Діаграма дерева вузлів Модель 1. Основні відділи фірми і їх робота. Кінцева модель (ТО-ВЕ)

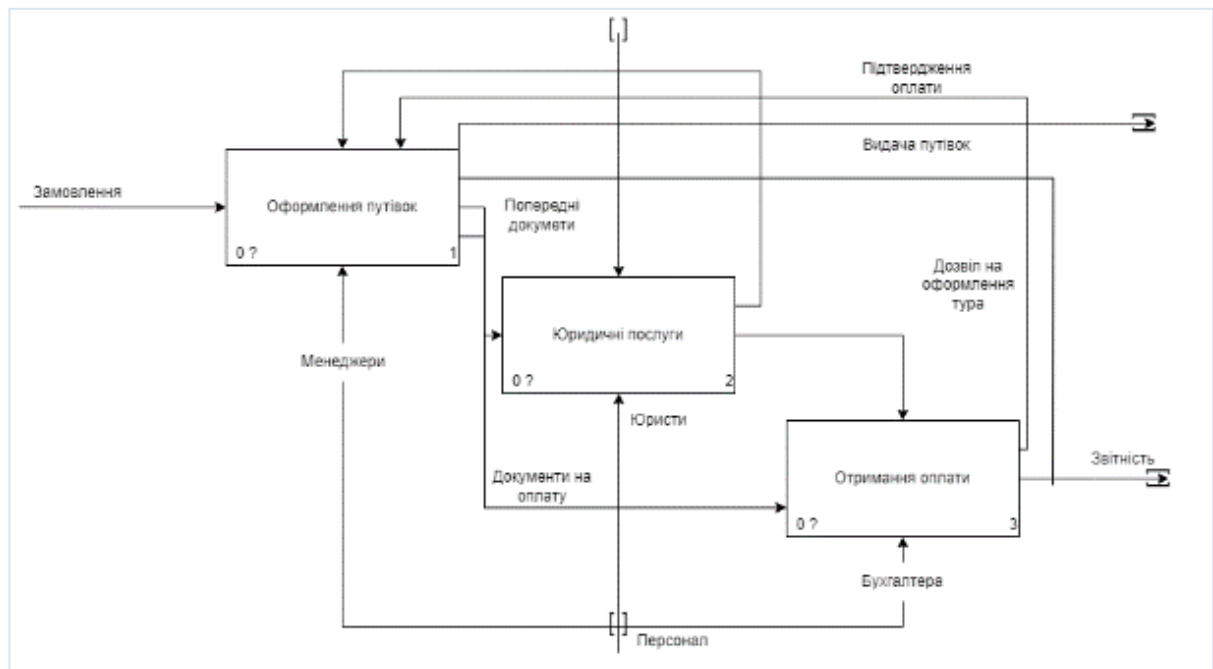


Рисунок Г.6 – Діаграма декомпозиції IDEF0 Автоматизація послуг «Туристичної фірми" Модель 2 (ТО-ВЕ)

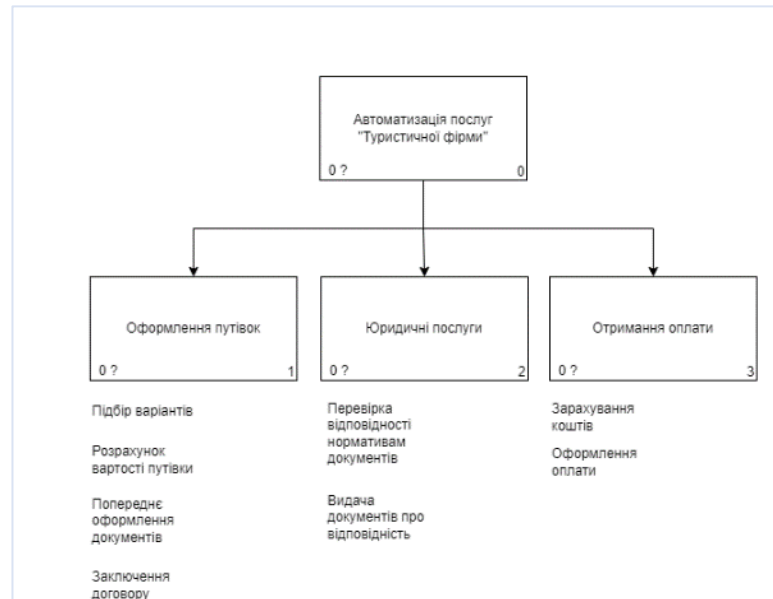


Рисунок Г.6 – Діаграма дерева вузлів Модель 2 (AS-IS)



Рисунок Г.7 – Діаграма декомпозиції IDEF0 Автоматизація послуг «Туристичної фірми" Модель 2. Отримання оплати за послуги (ТО-ВЕ)

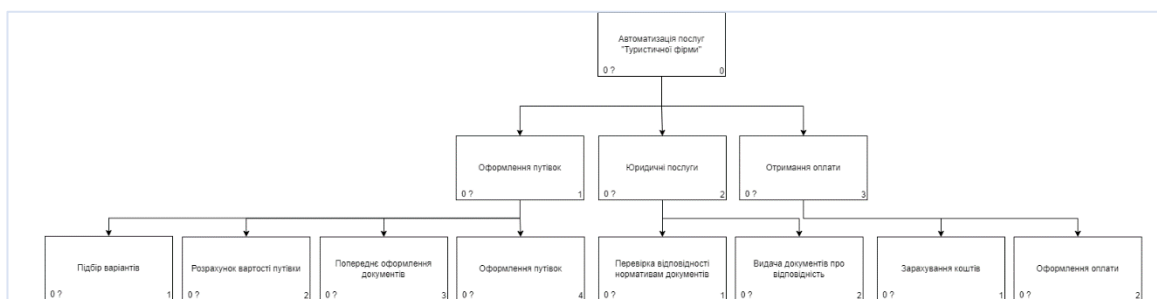


Рисунок Г.8 – Діаграма дерева вузлів Модель 2. Основні відділи фірми і їх робота. Кінцева модель (ТО-ВЕ)

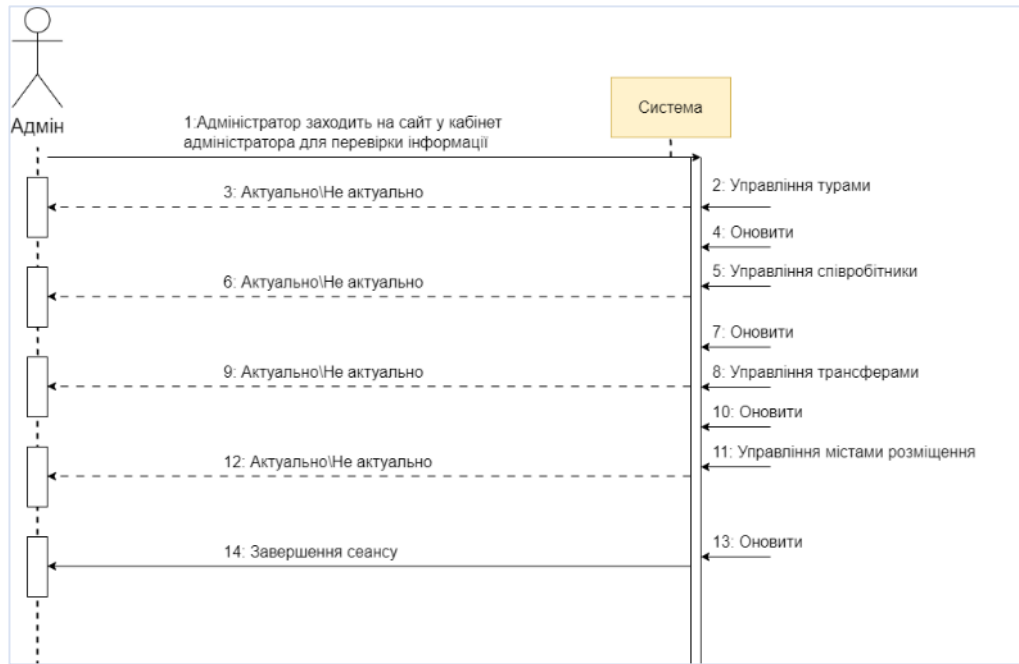
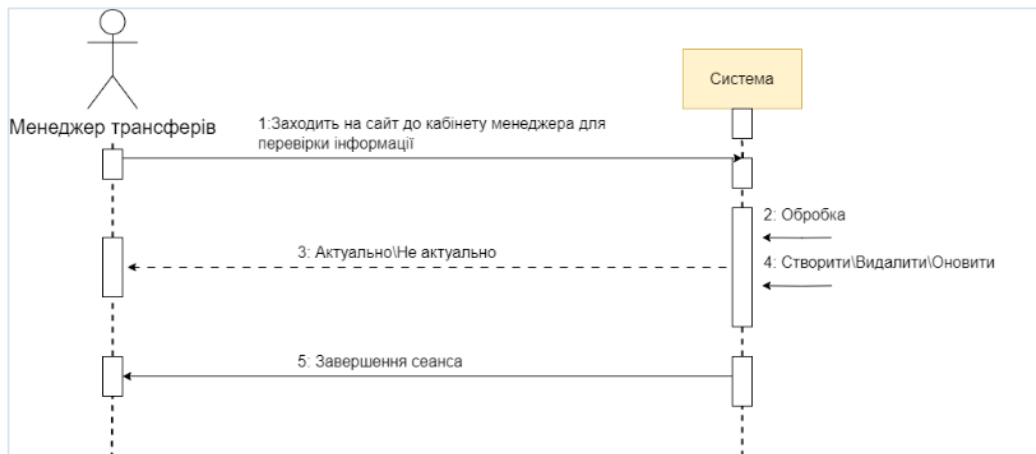


Рисунок Г.10 – Діаграма послідовності Адмін



Рисунок Г.11 – Діаграма послідовності Менеджер турів



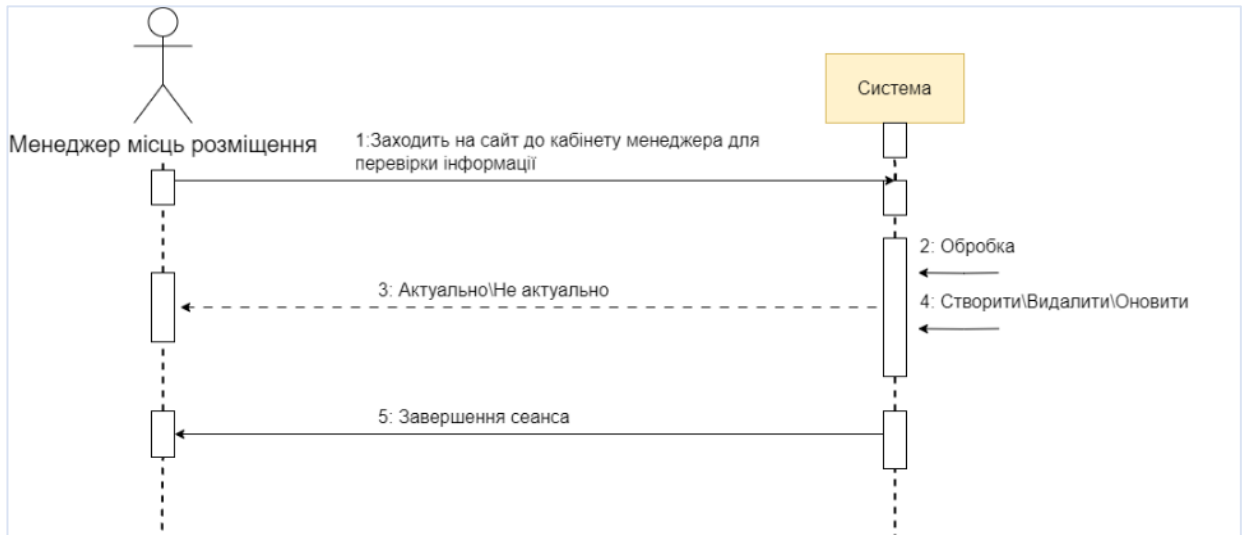


Рисунок Г.12 – Діаграма послідовності Менеджер розміщення готелів

Таблиця «Клієнт»

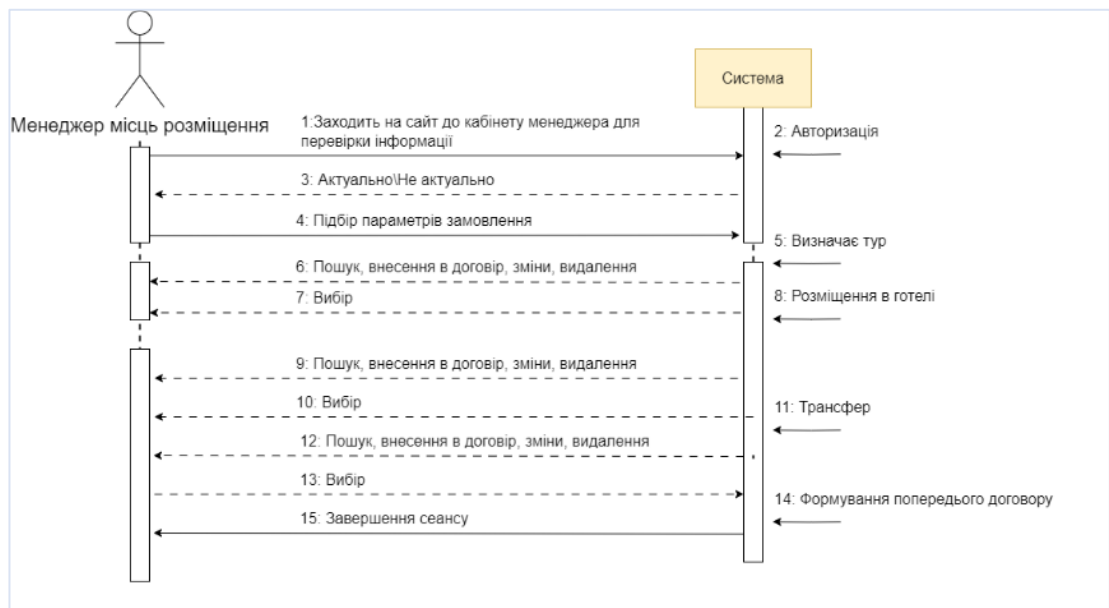


Рисунок Г.13 – Діаграма послідовності Клієнт

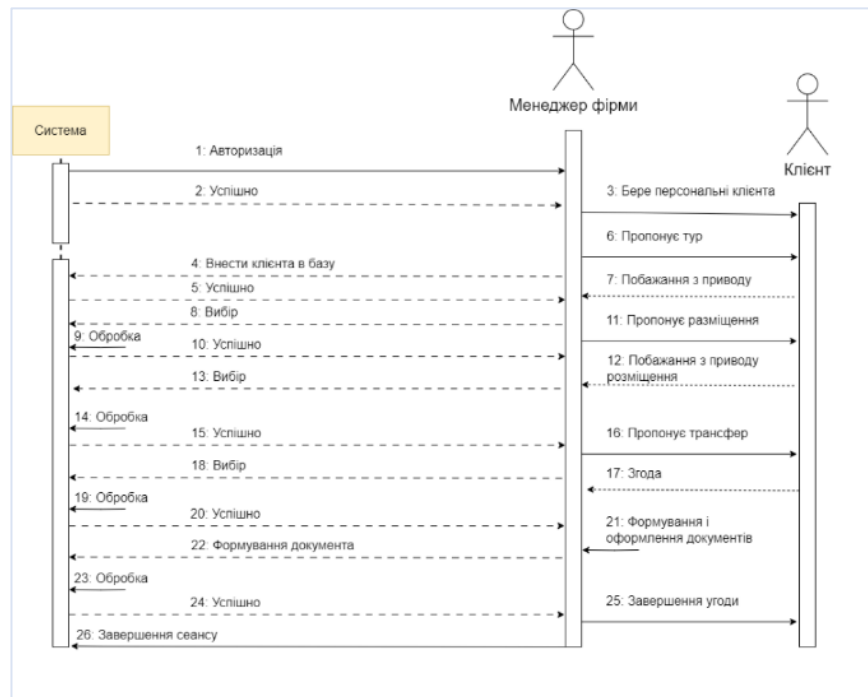


Рисунок Г.14 – Діаграма послідовності Менеджер та новий Клієнт

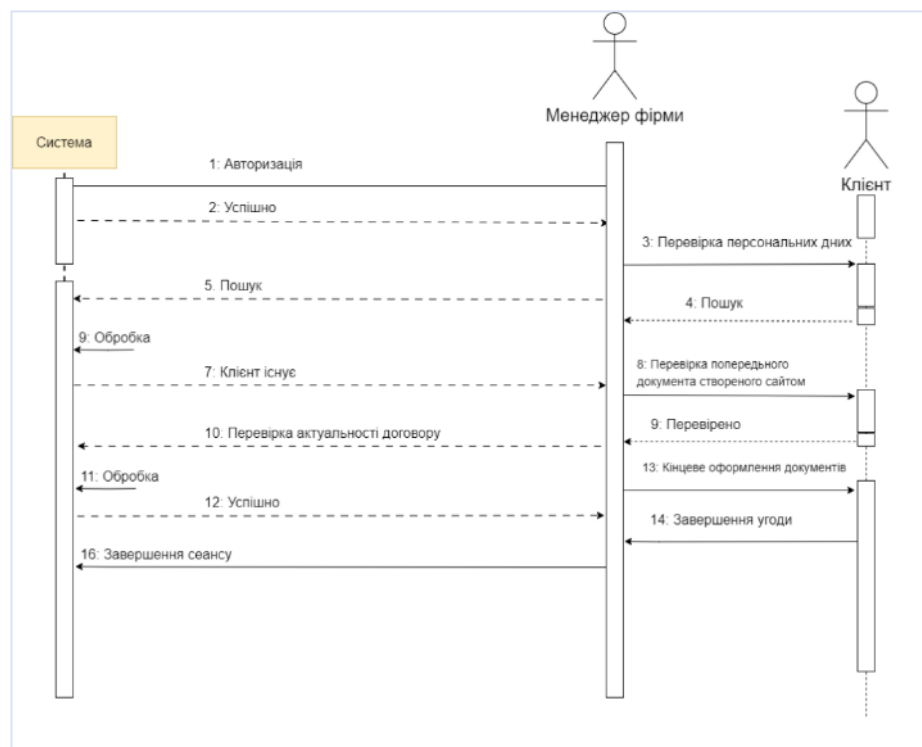


Рисунок Г.15 – Діаграма послідовності Менеджер та Клієнт котрий вже звертався

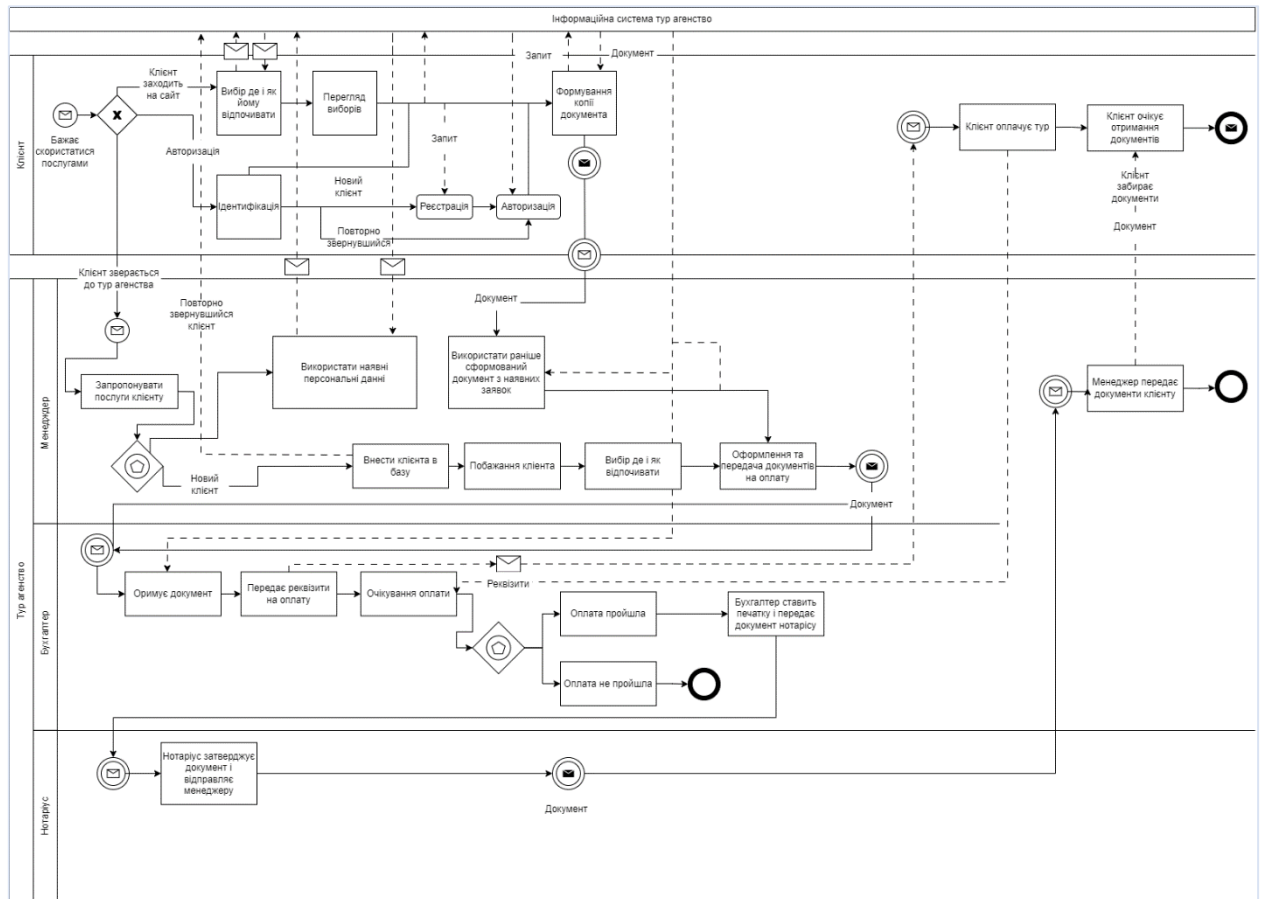


Рисунок Г.16 – Діаграма BPMN 2.0

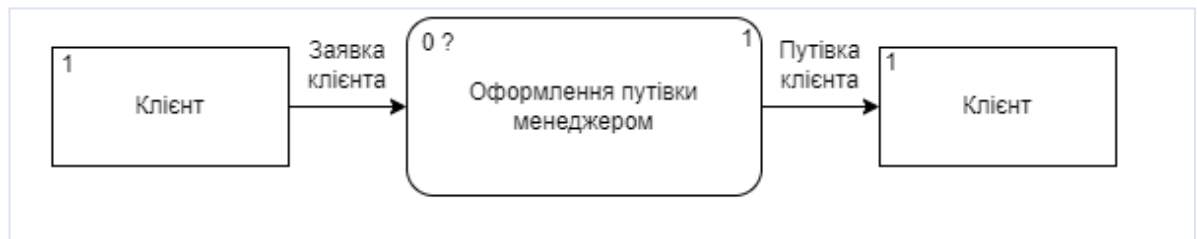


Рисунок Г.17 – Діаграма потоків даних. Під процес продажу путівки

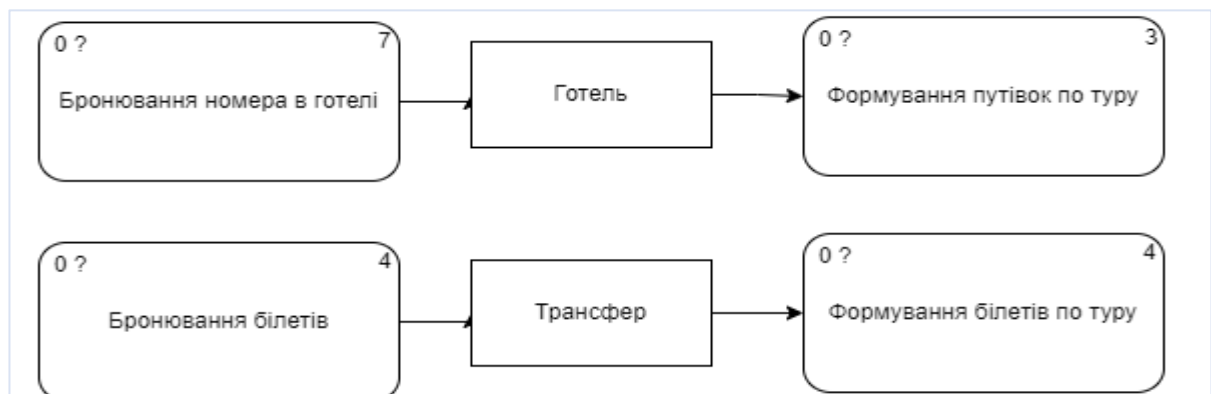


Рисунок Г.18 – Діаграма потоків даних. Під процес формування туру

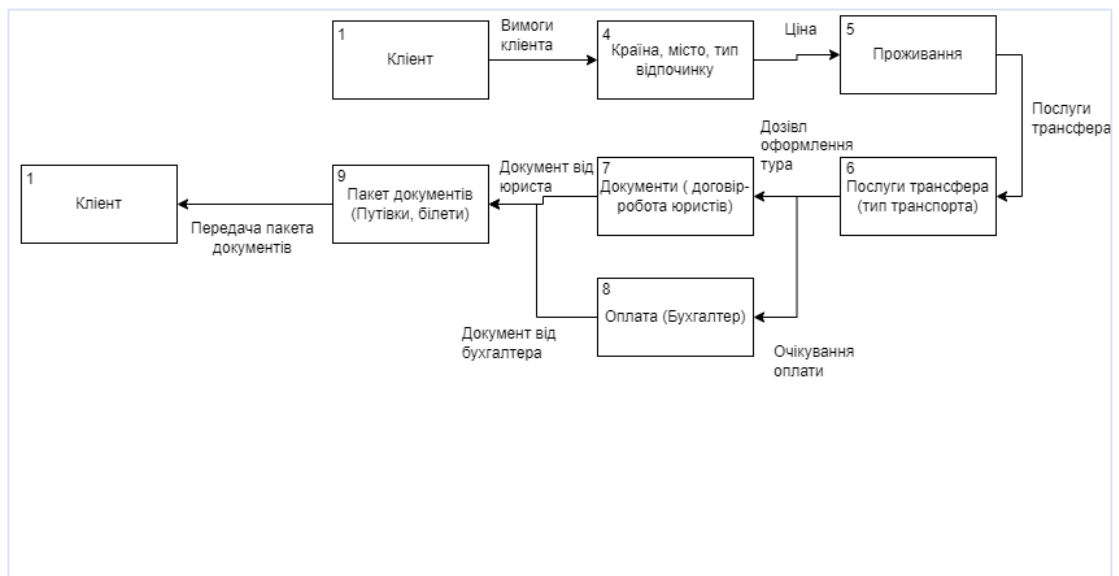


Рисунок Г.19 – Діаграма потоків даних Формування туру. Підбір варіантів

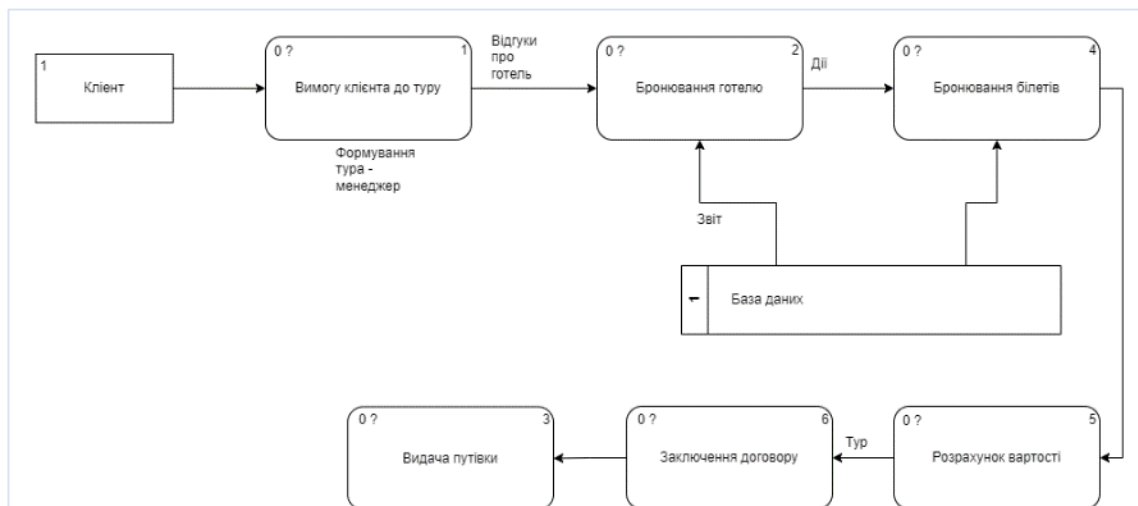


Рисунок Г.20 – Діаграма потоків даних. Робота менеджера туристичної фірми

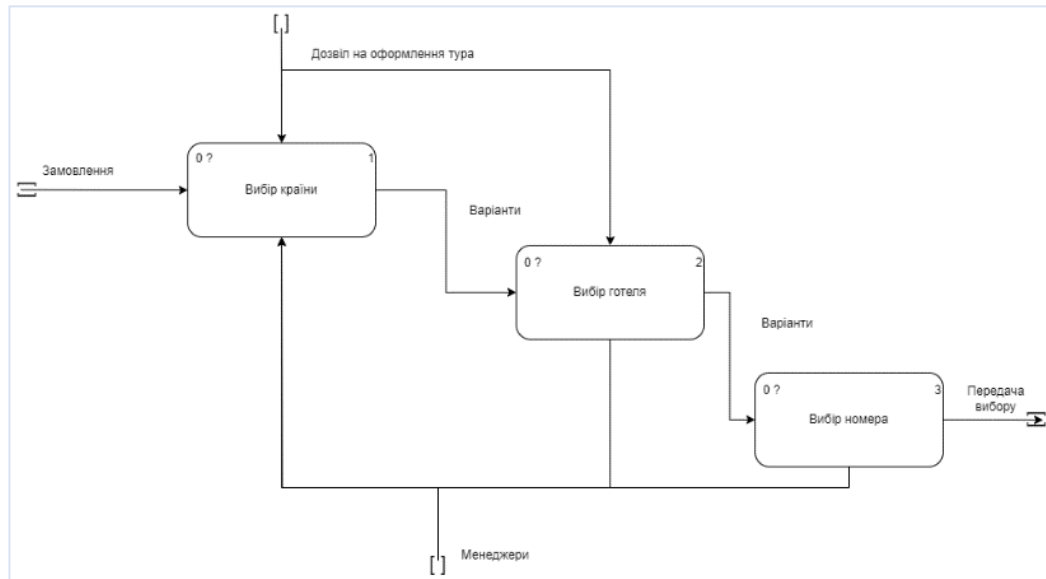


Рисунок Г.21 – Діаграма потоків даних. Робота менеджера туристичної фірми. Підбір варіантів для відпочинку

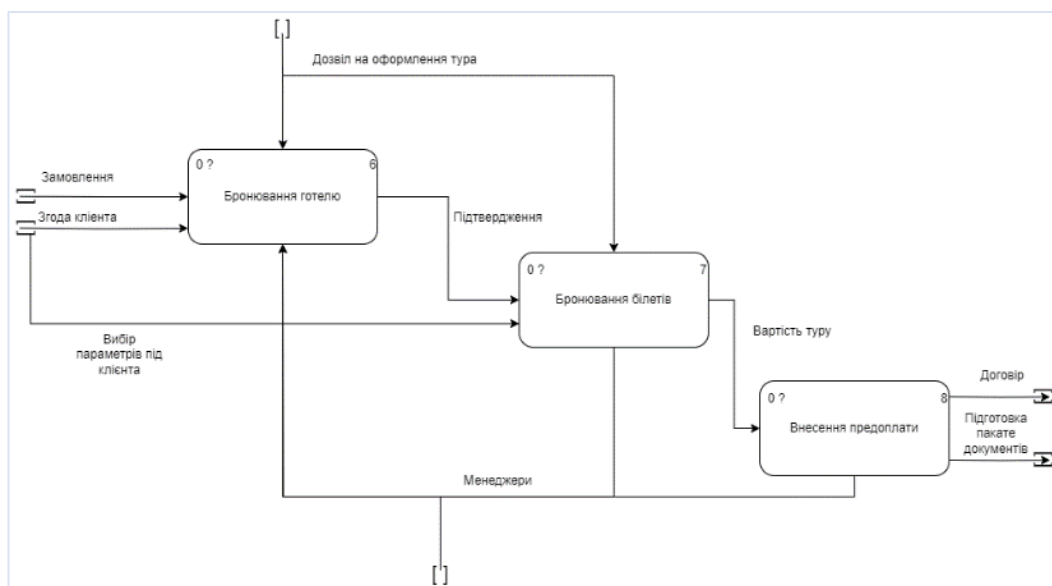


Рисунок Г.22 – Діаграма потоків даних. Робота менеджера туристичної фірми. Попереднє оформлення документів

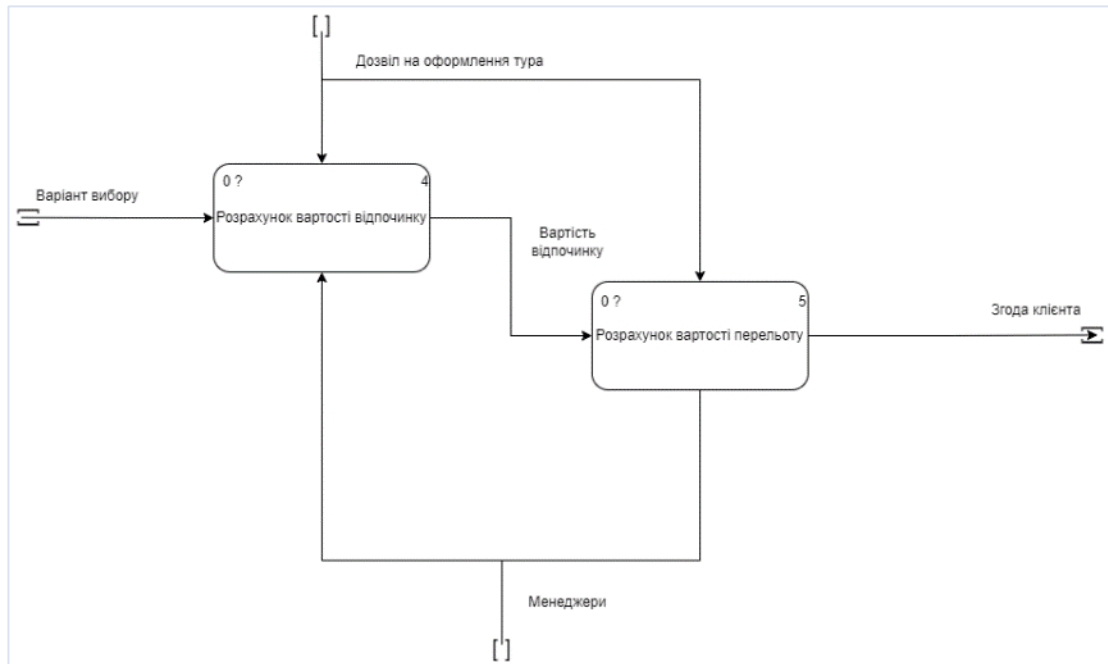


Рисунок Г.23 – Діаграма потоків даних. Робота менеджера туристичної фірми. Розрахунок вартості путівки

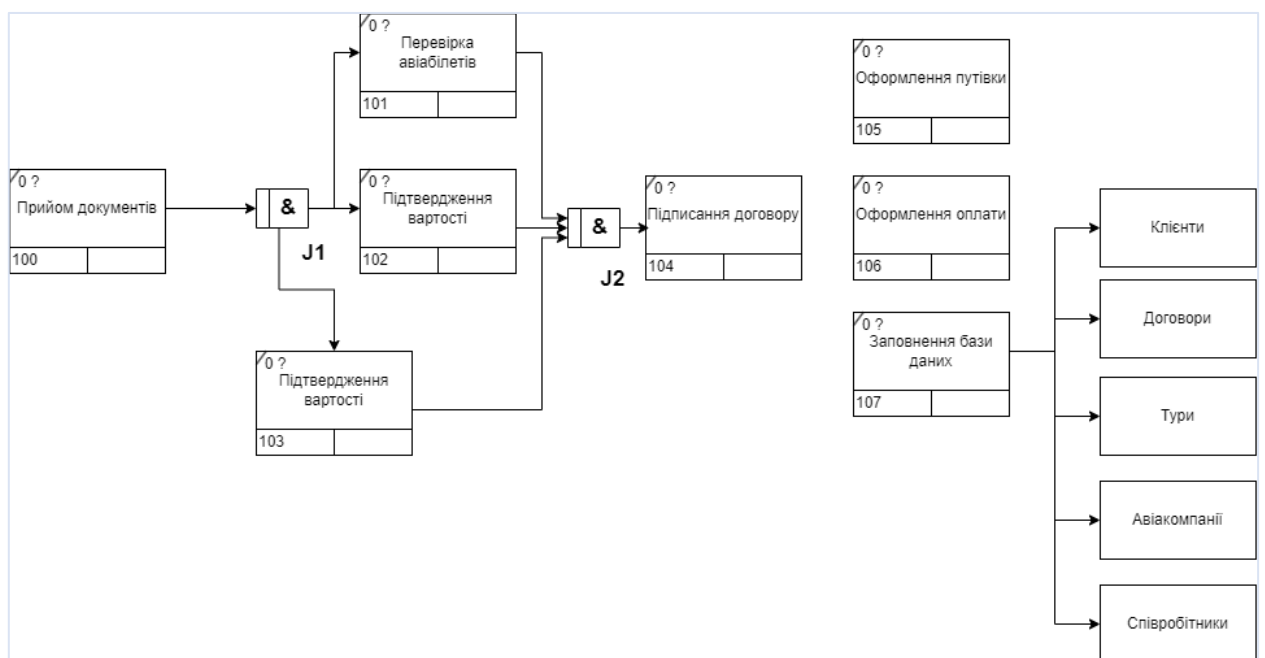


Рисунок Г.24 – Діаграма IDEF3. Робота менеджера туристичної фірми. Продаж квитків. Занесення даних в базу даних

ДОДАТОК Д
Слайди презентації



Рисунок Д.1 – Слайд презентації № 1

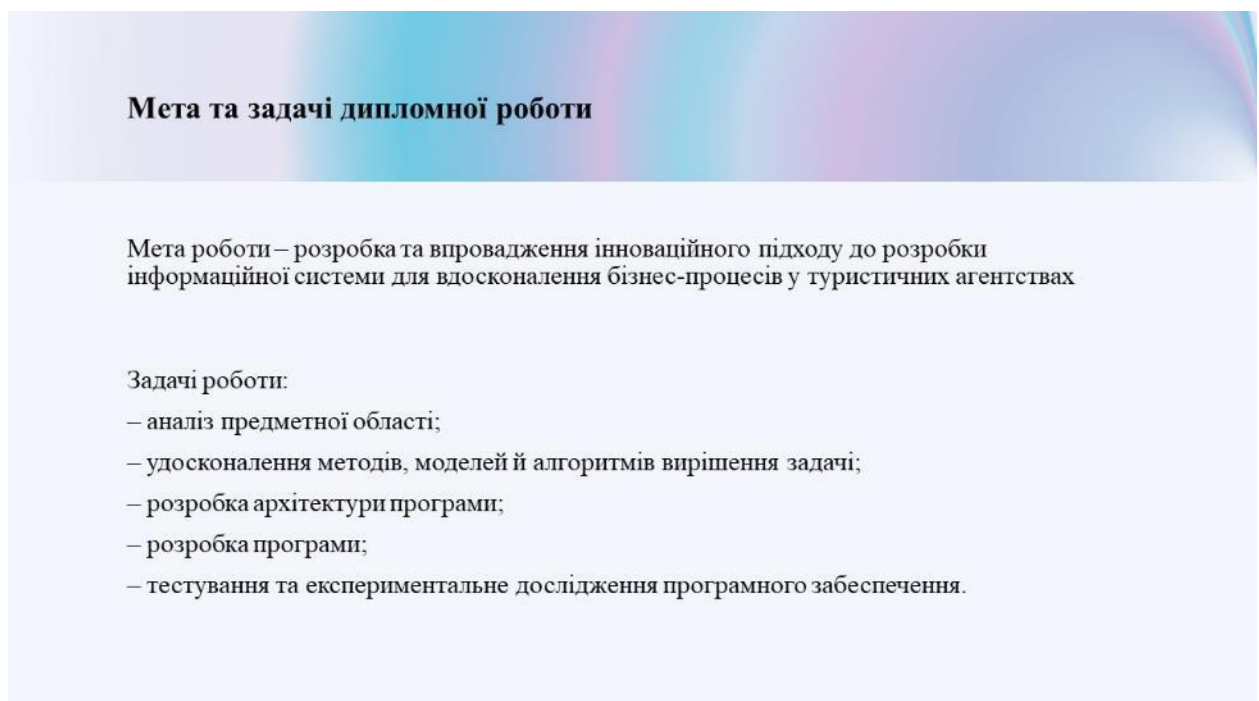


Рисунок Д.2 – Слайд презентації № 2

Аналіз предметної області: методи для оцінки ефективності

Метод параметричної лінійної комбінації даних:

$$a = \frac{a - \min(a)}{\max(a) - \min(a)}$$

$$S = w_1 * a_1 + w_2 * a_2 + w_3 * a_3 \dots + w_n * a_n$$

Метод параметричного порівняння експериментальних даних А/В тестування:

$$H_0 : KPI_{old} = KPI_{new}$$

$$H_1 : KPI_{old} \neq KPI_{new}$$

$$KPI_{old} = \frac{x_{old}}{n_{old}}$$

$$KPI_{new} = \frac{x_{new}}{n_{new}}$$

Рисунок Д.3 – Слайд презентації № 3

Аналіз предметної області: порівняння існуючих продуктів

| Criteria | Tourplan | Lemax | Dolphin Dynamics |
|---------------------------------|--------------------------------|------------------------------------|----------------------------------|
| License | Yes | Yes | Yes |
| Operating System | Cross-platform | Cross-platform | Cross-platform |
| Code Accessibility | Closed Code | Closed Code | Closed Code |
| Trial Version | Yes | Yes | Yes |
| Administration | Yes | Yes | Yes |
| Data Import/Export | Standard import/export options | Easy data exchange between modules | Integration with various formats |
| API | Limited | Available, but with limitations | Integrations with other systems |
| Multi-user Access | Yes | Yes | Yes |
| Modern Technology Compatibility | Yes | Yes | Yes |
| User Support Level | High | Medium | High |
| System Scalability | Yes | Yes | Yes |
| System Requirements | Medium | Medium | Medium |
| Customization Options | Yes | Yes | Yes |
| Data Confidentiality | Yes | Yes | Yes |
| Reporting System | Yes | Yes | Yes |

Рисунок Д.4 – Слайд презентації № 4

Проектування архітектури: порівняння обраних мов програмування

| Criteria | Python | Java | JavaScript | PHP |
|---------------------------------------|------------------------------------|---------|------------|-------------|
| License | Python Software Foundation License | GNU GPL | - | PHP License |
| Ease of Learning and Coding | + | + | + | + |
| Cross-platform Compatibility | + | + | + | + |
| Handling Large Data Operations | + | + | + | + |
| Parallel Programming | + | + | + | - |
| Libraries for Machine Learning | + | ± | + | - |
| Integration with Other Languages | + | + | + | + |
| Clear and Comprehensive Documentation | + | + | + | + |

Продовження таблиці

| | | | | |
|--|---|--|--|-----------------|
| High Data Visualization | + | - | + | - |
| Increased Productivity | + | + | + | + |
| Extensive List of Open-Source Domain-Specific Packages | + | + | + | + |
| Community Support | + | + | + | + |
| Compatible Platforms | Windows, macOS, Linux, Unix | Cross-platform | Cross-platform | Cross-platform |
| Purpose | General programming, web development, data analysis | General programming, enterprise applications | Web development, interactive interfaces creation | Web development |

Рисунок Д.5 – Слайд презентації № 5

Проектування архітектури: порівняння обраних середовищ розробки

| Criteria | PHPStorm | Visual Studio Code | Eclipse | Sublime Text |
|----------------------------------|-------------|--------------------|------------------------|--------------|
| License | Proprietary | MIT License | Eclipse Public License | Proprietary |
| Cross-platform Compatibility | + | + | + | + |
| Auto-completion of Code | + | + | + | + |
| Repository Support | + | + | + | + |
| Fast and Secure Code Refactoring | + | - | + | - |

Продовження таблиці

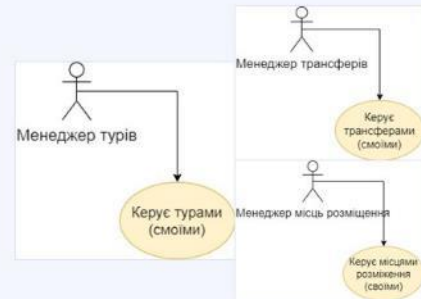
| | | | | |
|--|-------------------------|-----------------------|------------------|------|
| Code Inspection and Analysis | + | + | + | + |
| Ease of Installation | - | + | - | + |
| Presence of Interactive Console | + | + | + | + |
| Requires Less Memory for Storage | - | + | - | + |
| Syntax Highlighting | + | + | + | + |
| Cost | Paid | Free | Free | Free |
| Extensibility | + | + | + | + |
| Extensions for Working with Specific Programming Languages | +(main support for PHP) | +(extensions for PHP) | +(using plugins) | + |

Рисунок Д.6 – Слайд презентації № 6

Проектування архітектури: структурна схема програмного продукту



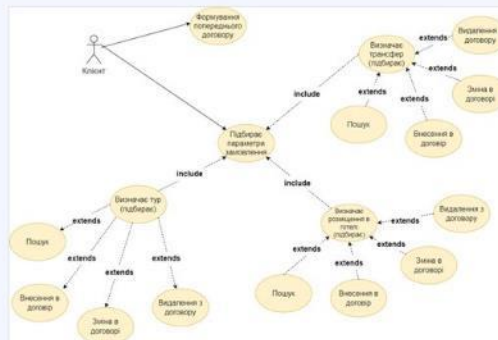
Діаграма варіантів використання Адмін



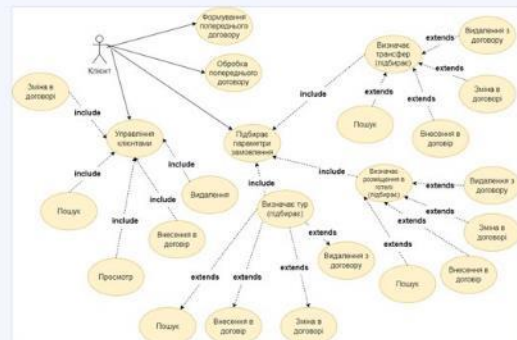
Діаграми варіантів використання Менеджер

Рисунок Д.7 – Слайд презентації № 7

Проектування архітектури: структурна схема програмного продукту



Діаграма варіантів використання Клієнт



Діаграма варіантів використання Менеджер фірми

Рисунок Д.8 – Слайд презентації № 8

Проектування архітектури: структурна схема програмного продукту

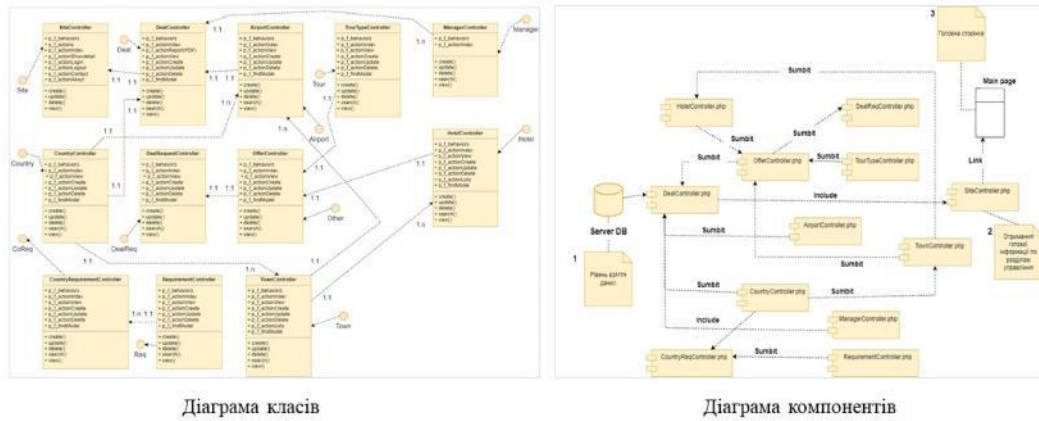


Рисунок Д.9 – Слайд презентації № 9

Проектування архітектури: структурна схема програмного продукту



Рисунок Д.10 – Слайд презентації № 10

Тестування та дослідження програмного забезпечення

| | | | |
|---|--|--------------------------------|--|
| Назва: | Створення заявки | | |
| Функція: | Створення заявки | | |
| Дія | Очікуваний результат | Результат тесту:
● пройдено | |
| Передумова: | | | |
| Відкрийте сайт
http://localhost/Travel_CRM/web/ | Сайт відкритий і доступний | пройдено | |
| Натисніть на кнопку "Details" | Сторінка "Details" відкрита і доступна | пройдено | |
| Натисніть на кнопку "Apply" | Сторінка "Create Deal Request" відкрита і доступна | пройдено | |
| Кроки тестування: | | | |
| Заповнити форму заявки: | Дані успішно введені | пройдено | |
| "Name": Melnikov Danylo Andriyovich | | | |

Тестування створення заявки

| | | | |
|---|--|--------------------------------|--|
| Назва: | Редагування заявки | | |
| Функція: | Редагування заявки | | |
| Дія | Очікуваний результат | Результат тесту:
● пройдено | |
| Передумова: | | | |
| Відкрийте сайт
http://localhost/Travel_CRM/web/ | Сайт відкритий і доступний | пройдено | |
| Натисніть на кнопку "Request Panel" | Сторінка "Request Panel" відкрита і доступна | пройдено | |
| Натисніть на кнопку "Update" | Сторінка "Update Deal Request" відкрита і доступна | пройдено | |
| Кроки тестування: | | | |

Тестування редагування заявки

Рисунок Д.11 – Слайд презентації № 11

Тестування та дослідження програмного забезпечення

| | IE8 | FF | Chrome |
|--|--------|--------|--------|
| Головна сторінка | | | |
| Коректне відображення сторінки | Passed | Passed | Passed |
| Header | Passed | Passed | Passed |
| Footer | Passed | Passed | Passed |
| Коректне відображення Fonts | Passed | Passed | Passed |
| Перевірка граматики контенту | Passed | Passed | Passed |
| Відображення кольорової гами, contrast-checker | Passed | Passed | Passed |
| Коректне відображення кнопок, CTA | Passed | Passed | Passed |
| Коректність скрола | Passed | Passed | Passed |
| Коректна робота реєстрації | Passed | Passed | Passed |
| Коректна відображення вказівника | Passed | Passed | Passed |
| Коректна робота функціоналу | Passed | Passed | Passed |
| Коректне відображення блоків | Passed | Passed | Passed |

Чек-лист верстки

| Вимоги | IE8 | IE9 | IE11 | FF | Chrome | Safari | Opera |
|----------------------------------|--------|--------|--------|--------|--------|---------|---------|
| Регістрація та профіль | | | | | | | |
| Регістрація та профіль | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Редагування профілю | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Функціонал адміністратора | | | | | | | |
| Робота з санкціями | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Робота з історією | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Робота з мітками | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Робота з країнами | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Робота з валютами | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Управління користувачами | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Функціонал користувача | | | | | | | |
| Створення записів | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Перегляд записів | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Редагування записів | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Видалення записів | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |
| Робота з санкціями | Passed | Passed | Passed | Passed | Passed | Skipped | Skipped |

Чек-лист функціоналу

Рисунок Д.12 – Слайд презентації № 12

Інструкція зі збирання та встановлення

Команди для запуску сервера:

```
# Запуск сервера
php yii serve
```

Команди для входу в локал:

```
# Вхід в локал
php yii serve --hostname=localhost --port=8080
```

Приклад методу створення готелю:

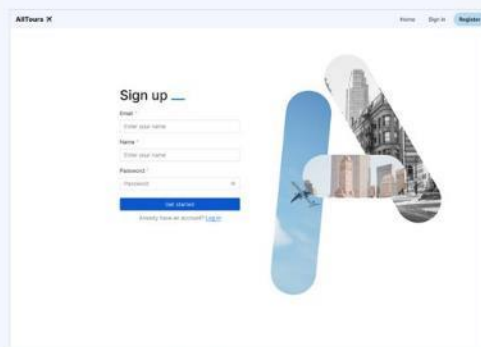
```
// HotelController.php
public function createAction()
{
    $model = new Hotel();

    if ($this->user->isGuest) {
        throw new MethodNotAllowedHttpException('Please register or log in!');
    }

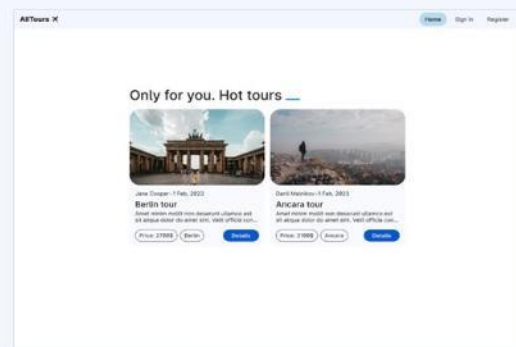
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    } else {
        return $this->render('create', [
            'model' => $model,
        ]);
    }
}
```

Рисунок Д.13 – Слайд презентації № 13

Склад та структура сайту: розділи



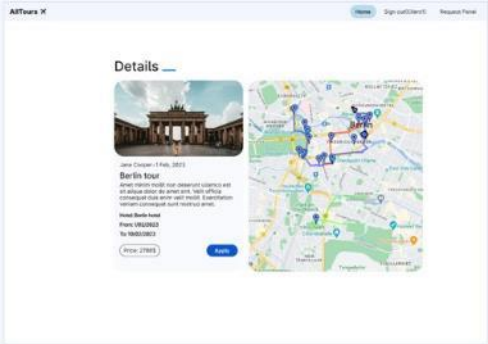
Реєстрація на Сайті



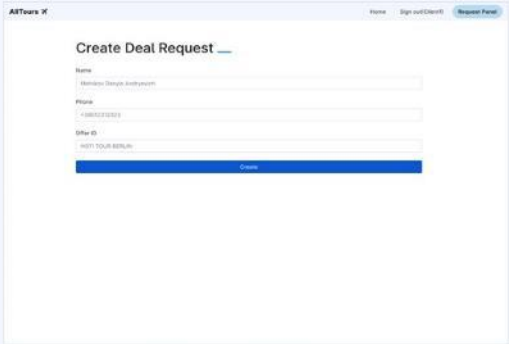
Знаходження Гість на сайті

Рисунок Д.14 – Слайд презентації № 14

Склад та структура сайту: розділи



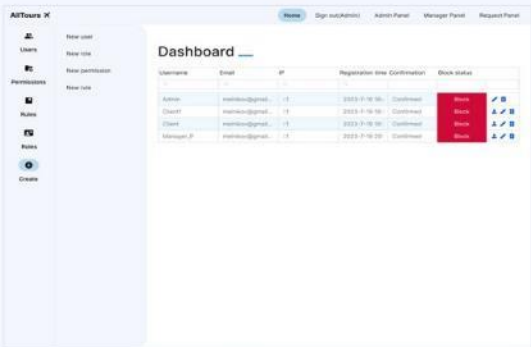
Сторінка вибору тура



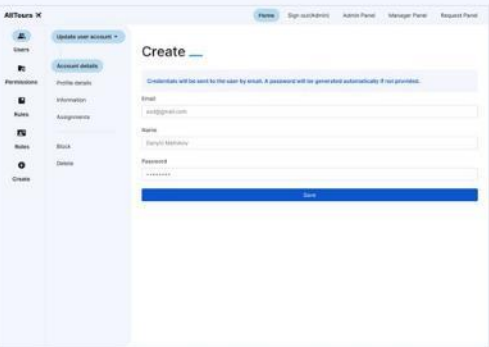
Сторінка для створення запиту тура

Рисунок Д.15 – Слайд презентації № 15

Склад та структура сайту: функціонал адміністратора



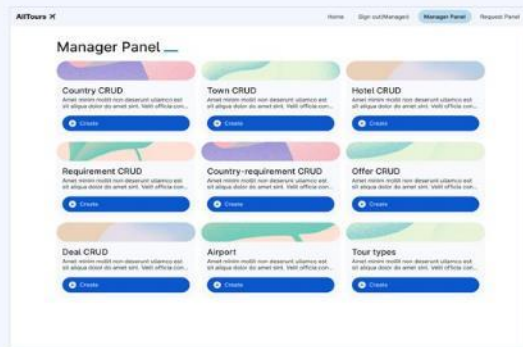
Функціонал панелі Адміністратора



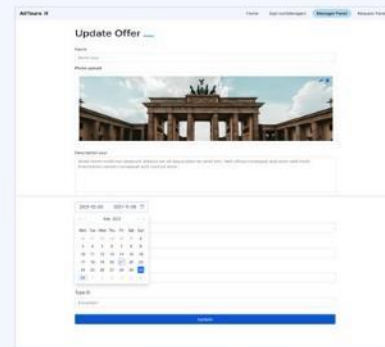
Створення акаунта користувача

Рисунок Д.16 – Слайд презентації № 16

Склад та структура сайту: функціонал менеджера



Панель управління турами Менеджер



Заповнення туру

Рисунок Д.17 – Слайд презентації № 17

Склад та структура сайту: функціонал менеджера-постачальника



Заповнення даних країн



Заповнення даних городів

Рисунок Д.18 – Слайд презентації № 18

Висновки

В результаті виконання дипломної кваліфікаційної роботи магістра було проведено повний цикл розробки програмного забезпечення, що включає аналіз предметної області, проектування архітектури системи, реалізацію компонентів системи, а також тестування отриманого продукту.

Аналіз предметної області виявив основні потреби та вимоги до системи з боку туристичних агенцій, що дозволило ефективно спланувати структуру та функціонал майбутнього програмного продукту. Вибір технологій та інструментарію був обґрунтований сучасними тенденціями в області розробки програмного забезпечення та специфікою завдань, що стояли перед системою.

Практична частина проєкту включала розробку бази даних, модулів для обробки даних, користувацького інтерфейсу, а також системи звітності. Реалізована система демонструє високий рівень надійності, зручності у використанні та гнучкості налаштувань під конкретні потреби користувача.

Тестування програмного продукту показало його ефективність та правильність роботи. Експериментальне дослідження підтвердило, що автоматизація дозволяє значно покращити швидкість та якість обслуговування клієнтів туристичної агенції.

Розділи керівництва для програміста та оператора забезпечують необхідну інформацію для ефективного використання системи та її подальшої підтримки. Важливою частиною роботи стала розробка заходів з охорони праці, які забезпечують безпеку використання системи в умовах реальної експлуатації.

Система може бути успішно впроваджена на практиці і принести значну користь для діяльності туристичних агенцій. Рекомендації, надані у висновках, можуть бути використанні для подальшого розвитку та масштабування системи.

Рисунок Д.19 – Слайд презентації № 19

ДЯКУЮ —————
ЗА УВАГУ

Рисунок Д.20 – Слайд презентації № 20