КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»
(назва освітньої програми)

Кваліфікаційна робота бакалавра

на тему: «Веб-сервіс із пошуку та підбору пропозицій для туристів»

Виконав _____
(Підпис)
Сухіна Андрій Юрійович
(прізвище, ім'я, по батькові)

Керівник Шолохов Олексій Вікторович
(прізвище, ім'я, по батькові)

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")
Завідувач кафедри _____ Плескач В.Л.
(Підпис) (Прізвище, ініціали) (Дата)

Засвідчую, що у цьому курсовому проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра прикладних інформаційних систем

Назва теми: «Веб-сервіс із пошуку та підбору пропозицій для туристів»

Освітня програма: Прикладні інформаційні системи
Спеціальність: Комп'ютерні науки

ПІБ	Підпис
Сухіна Андрій Юрійович	

Назва роботи українською та англійською мовами

Веб-сервіс із пошуку та підбору пропозицій для туристів

Web search and selection tourist offers service

Мета бакалаврської кваліфікаційної роботи, завдання

Мета роботи: Розробка сучасного веб-сервісу для спрощення знаходження корисної інформації та підбору цікавої, вигідної пропозиції туристичної подорожі.

План роботи:

1. Сучасні підходи до розроблення і впровадження застосунку
2. Аналіз архітектурних рішень і вибір програмних засобів для реалізації веб–систем
3. Програмна реалізація веб-сервісу із пошуку та підбору пропозицій для туристів

ПІБ, ступінь, звання наукового керівника роботи: Шолохов Олексій Вікторович, к.ф -м.н.

ВІДОМІСТЬ ДИПЛОМНОЇ РОБОТИ

Складові частини дипломної роботи	Обсяг, арк.
Титульний аркуш	1 ст.
Завдання до дипломної роботи	1 ст.
Відомість дипломної роботи	1 ст.
Календарний план	1 ст.
Реферат (Анотація)	1 ст.
Анотація (іноземною мовою-англійською)	1 ст.
Зміст	2 ст.
Перелік скорочень	1 ст.
Вступ	2 ст.
Розділ 1	12 ст.
Розділ 2	19 ст.
Розділ 3	12 ст.
Висновки	2 ст.
Перелік використаних джерел	2 ст.
Додатки	9 ст.

<table>
<tr><td></td><td></td><td></td><td></td><td colspan="3">ДП ХХХХ 00.000.00</td></tr>
<tr><td></td><td>ПІБ</td><td>Підп.</td><td>Дата</td><td colspan="3"></td></tr>
<tr><td>Розробн.</td><td>Сухіна А.Ю.</td><td></td><td></td><td rowspan="4">Відомість дипломної роботи</td><td>Лист</td><td>Листів</td></tr>
<tr><td>Керівн.</td><td>Шолохов О.В.</td><td></td><td></td><td></td><td></td></tr>
<tr><td>Н/контр.</td><td>Макаренко С.А.</td><td></td><td></td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Зав.каф.</td><td>Плескач В.Л.</td><td></td><td></td></tr>
</table>

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Номер	Назва етапів кваліфікаційної роботи бакалавра	Термін виконання етапів кваліфікаційної роботи бакалавра	Відмітка про виконання
1.	Вибір теми та наукового керівника кваліфікаційної роботи бакалавра	26.10.2020	Виконано
2.	Видача завдання кваліфікаційної роботи бакалавра	23.11.2020	Виконано
3.	Настановча групова співбесіда з питань кваліфікаційної роботи бакалавра	01.12.2020	Виконано
4.	Затвердження плану кваліфікаційної роботи бакалавра	18.02.2021	Виконано
5.	Підбір та вивчення літературних та інших джерел з теми дослідження	25.02.2021	Виконано
6.	Підготовка і подання науковому керівнику першого варіанту І розділу роботи	05.03.2021	Виконано
7.	Підготовка і подання науковому керівнику першого варіанту ІІ розділу роботи	09.04.2021	Виконано
8.	Підготовка і подання науковому керівнику першого варіанту ІІІ розділу роботи	07.05.2021	Виконано
9.	Подання роботи у першому варіанті	11.05.2021	Виконано
10.	Оформлення пояснювальної записки кваліфікаційної роботи бакалавра	12.05.2021	Виконано
11.	Подання кваліфікаційної роботи бакалавра на попередній захист	24.05.2021	Виконано
12.	Врахування зауважень керівника і подання роботи в остаточному варіанті (з відповідним висновком про допуск) на кафедру	28.05.2021	Виконано
13.	Затвердження роботи в цілому (підготовка письмового відгуку керівника, письмова рецензія на бакалаврської роботу)	11.06.2021	Виконано
14.	Захист кваліфікаційної роботи бакалавра	23.06.2021	Виконано

Здобувач вищої освіти _____

(підпис)

Керівник _____

(підпис)

АНОТАЦІЯ

До бакалаврської дипломної роботи Сухіни Андрія Юрійовича на тему «Веб-сервіс із пошуку та підбору пропозицій для туристів»

Дана дипломна робота присвячена створення туристичного веб-сервісу з використанням інтерфейсу прикладного програмування для серверу Web API, динамічної, інтерактивної технології користувацьких інтерфейсів веб-додатків Ajax, яка полягає в фоновому обміні даними браузера та сервера, а також використання Microsoft SQL Server для запису даних.

Для розробки веб-сервісу було проведено дослідження актуальних, на даний час, туристичних веб-серверів та технологій, які використовуються для розробки.

Доцільним було використання мов програмування JavaScript та C#. Також для проекту було використано Microsoft SQL Server та Microsoft SQL Server Management Studio для адміністрування.

У даній роботі описані результати використання програмних застосунків, програмних інструментів для розробки, детальний опис розробки та користувацького інтерфейсу.

Загальний обсяг роботи: 67 сторінок, 24 рисунки, 3 таблиці та 20 посилань.

Ключові слова: JavaScript, ASP .Net Core, C#, Ajax, web server, web interface, веб-сервіс, туризм.

ANNOTATION

To the bachelor's thesis of Sukhina Andrii Yuriiovych on a theme «Web service on search and selection of offers for tourists»

This thesis is devoted to the creation of a travel web service using the application programming interface for the Web API server, dynamic, interactive technology of user interfaces of web applications, which consists in the background exchange of browser and server data, and the use of Microsoft SQL Server to record data.

To develop the web service, a study was conducted of current, currently, travel web servers and technologies used for development.

It was advisable to use JavaScript and C # programming languages. Microsoft SQL Server and Microsoft SQL Server Management Studio for administration were also used for the project.

This paper describes the results of using software applications, software tools for development, a detailed description of the development and user interface.

Total workload: 67 sides, 24 figures, 3 tables and 20 references.

Keywords: JavaScript, ASP .Net Core, C #, Ajax, web server, web interface, web service, tourism.

Зміст

ПЕРЕЛІК СКОРОЧЕНЬ

БД – База даних

MVC – Model View Controller

СУБД – Система управління базами даних

ТЗ – Технічне завдання

JSON – JavaScript Object Notation

ПЗ – Програмне забезпечення

РСУБД – Реляційна система управління базами даних

XML – eXtensible Markup Language

REST – Representational State Transfer

CSS – Cascading Style Sheets

SOAP – Simple Object Access Protocol

JSON – JavaScript Object Notation

XML – eXtensible Markup Language

HTTP – Hyper Text Transfer Protocol

ВСТУП

Туристичні послуги з кожним роком охоплюють все більшу кількість людей. Найефективніше привернути увагу можна пропонуючи послуги онлайн. Це не тільки зручно, але й економніше ніж орендувати приміщення для вашої компанії. Підприємства, які усвідомили можливості інформаційних ресурсів і впровадили їх, змогли не тільки оптимізувати свою діяльність, але й рости з шаленою швидкістю.

Туризм в Україні, один з найоптимістичніших та ймовірніших способів підняти економіку. Україна чудово пристосована для впровадження великого туризму, цьому добре сприяє: географічне положення, яке можна вигідно використовувати як центр розв'язки наземної та повітряної комунікації між Європою та Азією, сприятливий клімат, різноманітність рельєфів, чудовий культурний спадок та традиції. Слабкі комунікації та інформаційне сповіщення уповільнюють розвиток туризму. Український ринок складається переважно з іноземних туристичних компаній, а українські туристичні фірми взагалі мало хто знає.

Актуальність цієї роботи зумовлено великою кількістю відвідувань таких веб-сервісів та розвитком туризму по всьому світу. Тому важливо мати доступ до інформації, мати змогу не витрачаючи часу на поїздку до туристичного агентства для того щоб визначитись з місцем подорожі.

Метою роботи є створення туристичного веб-сервісу, де можна переглянути інформацію, підібрати тур, побачити відгуки людей.

Завданням даної роботи є:

– Аналіз існуючих туристичних веб-сервісів;
– Програмні рішення для створення веб- сервісів;
– Створення веб- сервісу.

Об'єктом дослідження роботи є наявні веб-сервіси, які пропонують туристичні послуги.

Предметом дослідження роботи є використання сучасних технологій для розробки та програмна реалізація.

Обрані наступні методи дослідження:

– Спостереження – вивчення пропозицій, що надають туристичні агентства;

– Порівняння – встановлення різниці між послугами, перевагами та недоліками, які надають різна туристичні агентства.

Новизною слугують технічні рішення для створення даного проекту та унікальний інтерфейс та структура веб-сервісу, яка має бути зручною та задовільнити кожного користувача. Також схема розробки проекту та система підбору туристичних пропозицій, яка заключається в простоті та зрозумілості використання. Зменшуючи кількість запитів від клієнта до сервера, зменшується кількість передавання непотрібних даних, тим самим оптимізується робота системи.

Практичне значення полягає в тому, що отримана в рамках кваліфікаційної роботи схема розробки веб-сервісу може бути використана для створення будь-якого веб-сервісу обслуговування користувачів.

У процесі виконання кваліфікаційної роботи було використано:

– Мова програмування JavaScript;

– Мова програмування C#;

– РСУБД Microsoft SQL Server;

– HTML, CSS для верстки веб-сервісу;

Кваліфікаційна робота складається із переліку умовних позначень, вступу, трьох розділів, висновку, переліку використаних джерел і додатків.

РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ВЕБ-СЕРВІСИ

1.1 Теоретичні відомості та структура веб-сервісу

Веб-сервіс - це програмна система, яка розпізнається URI, а її загальнодоступні інтерфейси та з'єднання визначені та описані мовою XML. Описи цієї програмної системи мають змогу знаходитись іншими системами, які можуть взаємодіяти з повідомленнями на основі XML через Інтернет-протоколи відповідно до цього опису [1]. Веб-сервіси це концепція створення таких додатків, функціями яких можна користуватись за допомогою стандартних протоколів Інтернету. Зараз ця концепція використовується і розробляється багатьма провідними компаніями в IT-галузі. Концепція веб-сервісів реалізується з використанням ряду технологій, які стандартизовані World Wide Web Consortium (W3C).

Працюють веб-сервіси на основі таких стандартів і технологій:

– XML - це розширена мова розмітки, вона потрібна для розмітки та передачі відструктурованих даних;

– SOAP – протокол існуючий для обміну повідомленнями на основі XML;

– WSDL – мова призначена для опису зовнішніх інтерфейсів веб-сервісу на основі XML;

– UDDI - універсальний інтерфейс для ідентифікації, опису та інтеграції (Глобальне відкриття, опис та інтеграція). Список веб-сервісів та інформація про постачальників веб-сервісів в масове користування або для конкретних фізичних чи юридичних осіб.

На рисунку 1.1 можна побачити взаємодію між технологіями.

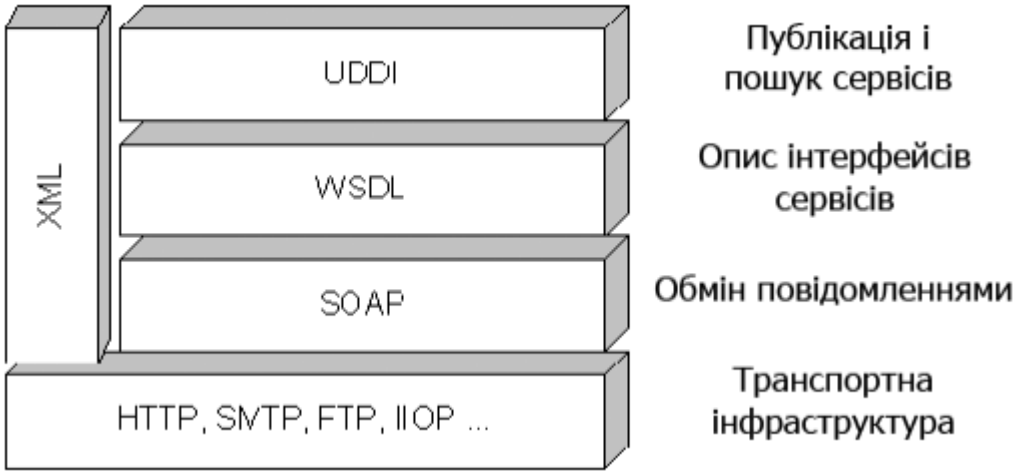


Рисунок 1.1 Схема структури веб-сервісів

XML написаний для вирішення проблем структурування даних, для цього створили таку просту мову, яка могла описувати дані такої структури. Це метамова, на ній пишуться спеціальні мови, які потрібні для того, щоб описати характеристики якоїсь певної структури. Такі мови мають назву словниками XML [2]. На відміну від HTML, XML не визначає спосіб відображення даних у документі. Яким чином відображаються дані для різних пристроїв, визначається мовою опису стилю XSL, яку можна порівняти з CSS для HTML.

Обробка XML-документів здійснюється наступним чином. Текст документа XML зчитує та аналізує програма, яка називається процесором XML. Процесор XML не знає семантичного значення даних у документі, він лише аналізує текст документа та перевіряє правила XML. Коли документ завершено, аналізовані результати передаються через XML-процесор програми, який виконує значущу обробку. Якщо документ відформатовано неправильно, тобто присутні якісь синтаксичні помилки, користувач XML-процесора повинен повідомити.

SOAP повинен визначити набір специфікацій правил даних на основі XML. Спільна мова XML, спільна для сервера SOAP та клієнта. Веб-служба, яка відповідає специфікаціям веб-служб SOAP, є веб-службою SOAP. SOAP - це простий протокол захисту доступу.

Це дозволяє нам створити окремий інтерфейс програмування (API). Ми можемо передавати дані з однієї програми в іншу. API отримує запити від клієнта на сервері та надсилає відгук клієнту через Інтернет-протоколи, такі як HTTP, SMTP та інші [3]. SOAP - це стандартизований протокол, який надсилає повідомлення за допомогою інших протоколів, таких як HTTP та SMTP. Специфікація SOAP - це офіційний веб-стандарт, розроблений та стандартизований World Wide Web (W3C). Оскільки SOAP є офіційним протоколом, він містить суворі правила та розширені функції безпеки, такі як відповідність ACID та внутрішні ліцензії.

WSDL - це мова, заснована на XML, яка визначає інтерфейс веб-служби, її структуру, функції та метод доступу. WSDL описує послуги як сукупність кінцевих точок мережі або портів [4]. WSDL забезпечує формат XML для специфікаційних документів. Абстрактний текст портів і повідомлень відрізняється від їх конкретного використання та характеру і дозволяє повторно використовувати ці формули. Адреси мережевих портів визначаються шляхом підключення декількох з'єднань, а набір портів визначає послугу. Повідомлення - це абстрактні описи даних, якими обмінюються, а типи портів - абстрактні набори підтримуваних операцій. Тобто, WSDL описує загальнодоступний інтерфейс для веб-служб.

Технологія Global Description Discovery and Integration (UDDI) - це реєстр веб-сервісів. Після підключення до цього реєстру користувач може шукати веб-служби, які пропонують та підходять найкраще. Набір послуг веб-реєстру надається замовнику для можливості пошуку та публікації в цих реєстрах. UDDI дозволяє публікувати послуги як у клієнтських програмах, так і особисто.

1.2 Актуальність інформаційних технологій в туристичному бізнесі

Актуальність розвитку і застосування інформаційних технологій у туризмі залежить основним чином від того, як, куди йде керуюча частина цього процесу. Оскільки сьогодні, в контексті стратегічних переваг туристичні фірми

знаходяться у боротьбі застосування інформаційних технологій, попит на свіжі ідеї організаційної і побудови стає дедалі актуальнішим.

Актуальність застосування інформаційних технологій також зростає через необхідність заохочувати економічний та виробничий підйом в країні.

Туризм в Україні, один з найоптимістичніших та ймовірних способів підняти економіку. Україна гарно підходить для застосування великого туризму, цьому добре сприяє: географічне положення, яке можна вигідно використовувати як центр розв'язки наземної та повітряної комунікації між Європою та Азією, сприятливий клімат, різноманітність рельєфів, гарне етнічне надбання та традиції. Слабкі комунікації та інформаційне сповіщення уповільнюють розвиток туризму [19].

До факторів розвитку інформаційних технологій належать:

– Величезний обсяг інформації, що дозволяє отримати всю інформацію про туристичну компанію, її відео та фотоматеріали, розробки та інші наочні матеріали;

– Можливість охопити велику кількість людей;

– Незалежність від місцезнаходження та цілодобовий доступ;

– Можливість миттєвого оновлення, що особливо важливо, якщо брати до уваги динаміку змін попиту та економічну доцільність пропозиції для його задоволення;

– Детальна статистика запитів - після встановлення лічильника входу користувачів на сайті можна визначити кількість запитів до джерела інформації, виявити потенційно цікаві нові пропозиції, реальний попит тощо.

– Ефективність використання інформаційних технологій у туризмі можна виявити проаналізувавши:

– Кількість користувачів глобальної мережі;

– Кількість продажів туристичних послуг через мережу;

– Кількість користувачів, які забронювали в глобальній розподільній системі.

За даними Всесвітньої туристичної організації, найбільша кількість користувачів мережі Інтернет складає США (200 млн. чоловік), Китай (111 млн. чоловік), Японія (85,29 млн. чоловік). В західній Європі приблизно 205 млн. користувачів. Згідно з дослідженням II-TOI, в Україні збільшилась частка користувачів на 46,2% за останні два роки, з 1,3 до 1,9 млн. це складає лише 0,2% від всіх користувачів Інтернету у світі. Останні роки обсяг онлайн-продажів туристичних послуг у США зростає найшвидше - на 19-20% щороку, у Західній Європі - на 37-49%, лідерами є Франція та Великобританія. Частка продажів в Україні через Інтернет туристичних послуг складає 7%, у США - 54%, а в Західній Європі - 51% [5].

Таким чином, нереально будувати туристичний бізнес без використання інформаційних технологій, особливо Інтернету, який має багато можливостей та переваг для росту бізнесу, а саме:

– Можливість миттєвого розміщення та пошуку інформації про «гарячі» поїздки, готельні номери, квитки та можливість їх бронювання в Інтернеті;

– Величезна, цілодобова, ефективна і відносно недорога реклама;

– Істотна економія за рахунок користуванням електронної пошти у співпраці з транспортними компаніями, готелями, туроператорами, іноземними партнерами тощо;

– Знижки, нові тури, політична та економічна ситуація в різних країнах, новини законодавства про туризм у цих країнах тощо. Можливість отримувати своєчасну інформацію про все вище зазначене.

Ефективність, яка може бути застосована при розробці веб-сервісу: реєстрація / ліцензія користувача, форма розгорнутого пошуку, блок новин, інтерактивний магазин сервісів, модуль підписки та інформаційний бюлетень, відповіді на поширені запитання (поширені питання), фото та відео галереї, опитувальники для відвідувачів на порталі, форум, відгуки, співтовариство, зворотній зв'язок, замовлення реклами.

Однією з основоположних відмінностей веб-сервісу є можливість спілкування між відвідувачами. Ця функція дозволяє створити спільноту користувачів, які регулярно відвідують сайт. За допомогою стандартної збірки реалізується більш складна чат або соціальна мережа.

Для теперішніх туристичних компаній в нових умовах веб-сайт є дуже важливим, це дозволяє демонструвати свою пропозицію широкому колу потенційних клієнтів з іншого боку. Електронна платформа компанії має можливість автоматизувати процес шляхом надсилання запитів, їх сортуванню, прийому та реєстрації замовлень, укладання угод та їх послуг. Гарна продуктивність можливостей Інтернет-комунікацій дозволяє зменшити час участі, приймати рішення, реалізовувати угоди та вигадувати нові туристичні продукти. Інформація та послуги доступні в Інтернеті цілодобово. Крім того, його комунікаційні функції відрізняються гарною гнучкістю, що допомагає легко змінювати інформацію та таким чином підтримувати її зв'язок без короткочасних затримок та витрат на поширення.

Ці дії зараджують сильно понизити трансакційні витрати, тобто витрати, пов'язані із встановленням та підтримкою взаємодії між компанією, стороною яка замовляє та постачає. При цьому вартість комунікацій зводиться до мінімуму порівняно з традиційними пристроями, а їх продуктивність і масштаб значно збільшуються.

Все, що вам потрібно зробити, - це створити веб-сайт компанії та розмістити його в Інтернеті - і ви можете швидко очікувати значного збільшення продажів туристичних послуг, швидкої віддачі інвестицій тощо. Однак ці очікування не завжди виправдані, оскільки продукт інформації, як і інші продукти, потребує матеріально-технічного забезпечення, маркетингу та реклами.

Потенційні клієнти часто розглядають веб-сторінку як віртуальний офіс підприємства. Для цього потрібна належна конструкція, і етап заснування не повинен бути вичерпним і обмежуватися технічними питаннями. Особливу увагу та контроль слід зосередити на роботі з даними відділу маркетингу та інформації.

Підсумовуючи це, створена, наповнена контекстом й актуалізована веб-сторінка туристичної компанії може підтримувати свою діяльність у цій галузі:

– Доступу до нових клієнтів;
– Формування візерунку;
– Популяризації послуг;
– Продажів туристичних продуктів;
– Налагодження діалогу з клієнтами;
– Накопичення даних про клієнтів.

Запорукою успіху є догодити інформаційним потребам відвідувачів веб-сайту. Потенційних клієнтів часто цікавить інформація про надані послуги та ціни на їх купівлю та власне туристичну компанію.

1.3 Порівняння існуючих туристичних веб-сервісів

В наш не легкий час пов’язаний з COVID19 туристичні агентства переживають не найкращі часи, але незважаючи на це їх залишається багато і бажаючому відкривається вибір.

Серед багатьох я хочу розглянути та співставити компанії «TUI GROUP», «Coral Travel» та «Феєрія Мандрів».

Німецьке туристичне агентство «TUI GROUP» одне з найкращих в світі. Активами цієї компанії є безліч готелів (близько 300), круїзні лайнери, агенції, авіакомпанії та магазини. Штаб-квартира знаходиться в Ганновері, Німеччина [6].

На рисунку 1.2 зображений веб-сайт туристичного агентства «TUI GROUP».

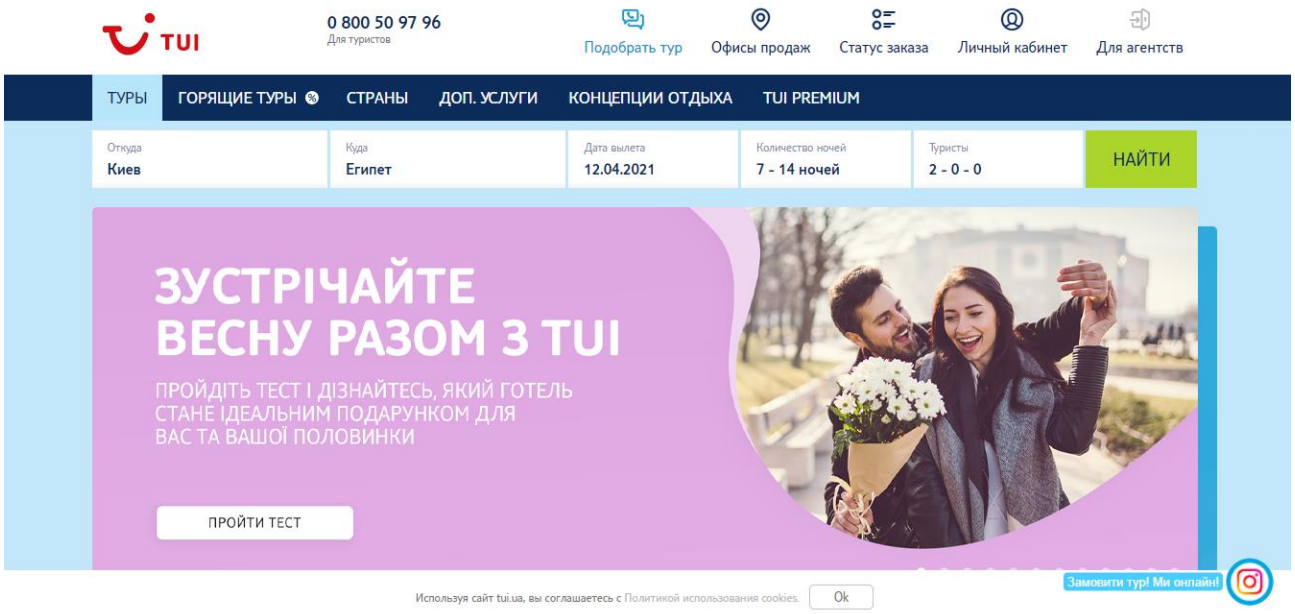


Рисунок 1.2 – Інтерфейс головної сторінки «TUI»

Проаналізувавши веб-сайт туристичного агентства «TUI» можна виділити наступні позитивні сторони:

– Приємний світлий інтерфейс;

– Присутня інформація про компанію, місцезнаходження офісу;

– Зручний пошук турів;

– Велика кількість відгуків;

– Наявність додаткових послуг таких як оренда авто за кордоном, онлайн страхування, подарункові сертифікати, SIM-карти роумінгу, перевірка агентства тощо.

Також варто відмітити декілька негативних сторін веб-сайту:

– Присутня лише російська мова;

– Довго завантажуються сторінки, інколи вимушено чекаєш до 10-15 секунд;

– Не відразу зрозуміло як забронювати тур.

Компанія «Coral Travel» (Україна, Білорусь, Грузія, Німеччина, Польща, Туреччина) входить в міжнародну структуру OTI Holding. «Coral Travel» має понад 25 років професійного досвіду у сфері туризму та пропонує лише якісні туристичні товари на українських ринках. «Coral Travel» пропонує найкращі

курорти та готелі 28 країн [8]. На рисунку 1.3 можна побачити інтерфейс головної сторінки.

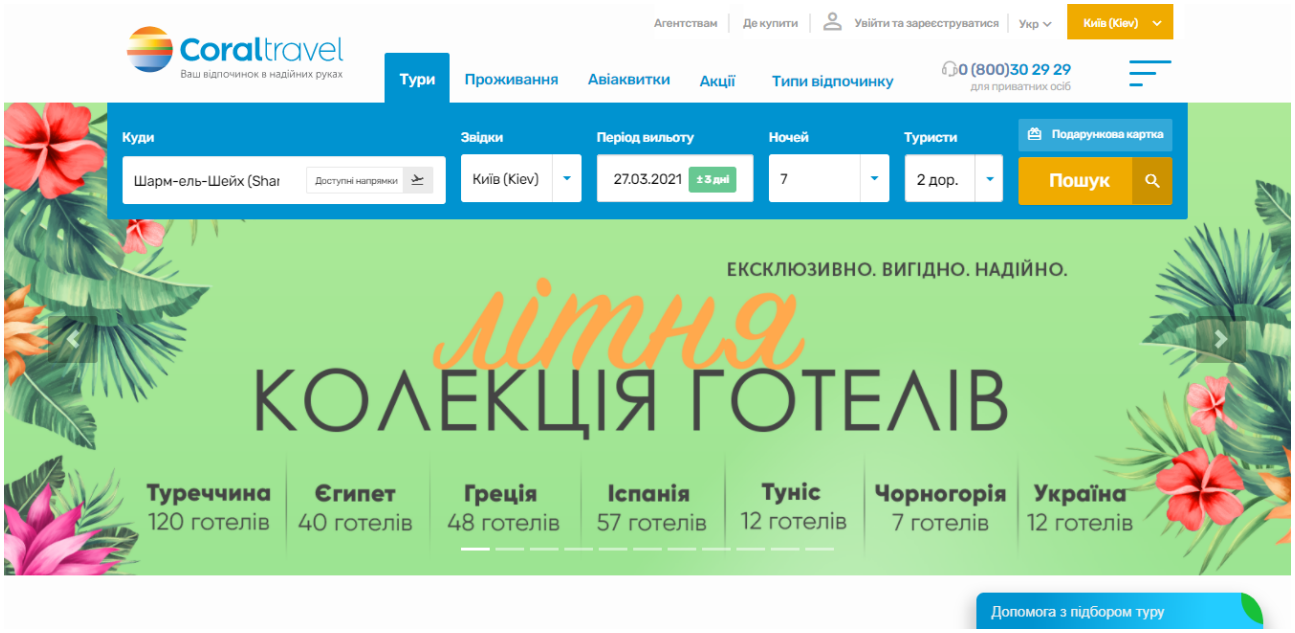


Рисунок 1.3 Інтерфейс сайту «Coral Travel»

Проаналізувавши сайт туристичної компанії «Coral Travel» можна виділити наступні позитивні сторони:

– Присутня як російська так і українська мова;
– На сайті опубліковані реквізити компанії та місцезнаходження;
– Взаємодія з соціальними мережами;
– Опубліковані відгуки та подяки;
– Є розділ з акціями;
– Швидке завантаження сторінок;
– Зручний пошук.
– Також на сайті потрібно виділити декілька негативних сторін:
– Постійно з’являється чат з консультантом на половину екрану;
– Немає додаткових послуг;
– Складно знайти відгуки про готель.

«Феєрія Мандрів» - українська компанія, штаб-квартира знаходиться у Києві, засновник Ігор Захаренко. Компанія була заснована у 2001 році та одразу взяла курс на туристичні послуги. Також в них є своя телепередача, де вони

розповідають про подорожі [7]. На рисунку 1.4 можна побачити інтерфейс головної сторінки сайту «Феєрія Мандрів»:

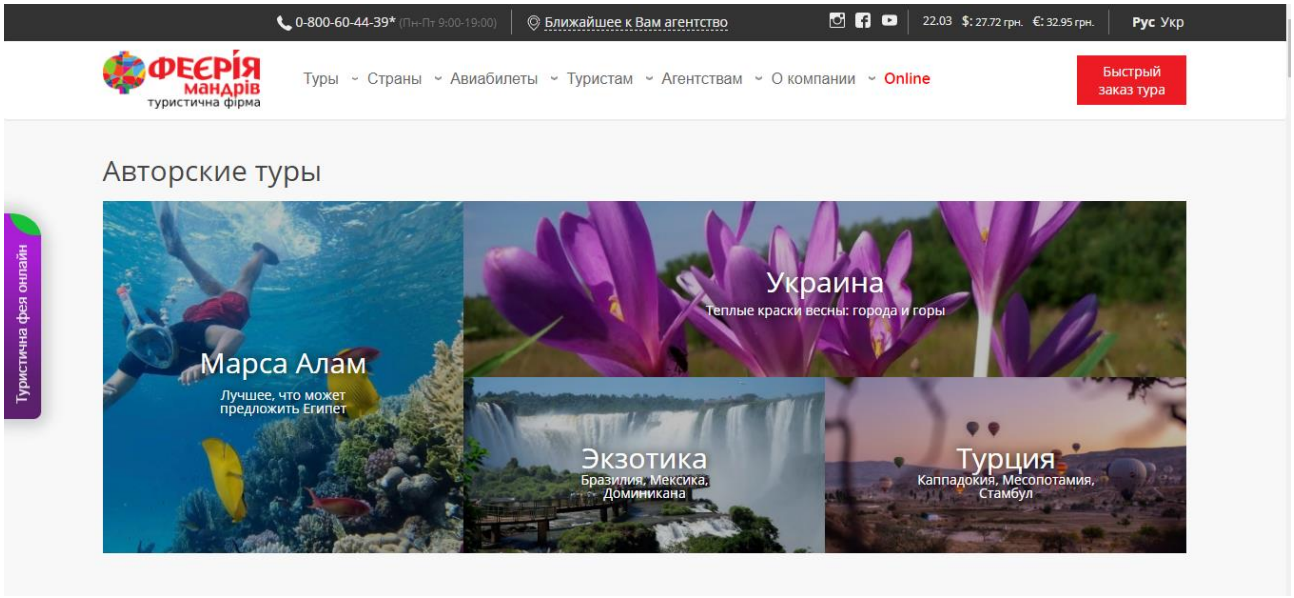


Рисунок 1.4 Інтерфейс веб-сайту компанії «Феєрія Мандрів»

Можна виділити декілька плюсів цього сайту:

– Багатомовність;

– Взаємодія з соціальними мережами;

– Є форма для заповнення швидкого замовлення туру після якої вам перетелефонує консультант та забронює тур.

– Негативних сторін набагато більше чим позитивних на мою думку:

– Малий фільтр пошуку;

– Незрозуміла побудова архітектури сайту;

– Мало інформації.

Підведемо підсумки в порівняльній таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця обраних туристичних веб-сервісів

Критерії	TUI	Coral Travel	Феєрія Мандрів
Дизайн	+	+	-
Зручність пошуку	+	+	-
Зручність бронювання	-	+	-

Продовження таблиці 1.1

Контактні дані	+	+	+
Багатомовність	-	+	+
Взаємодія з соціальними мережами	+	+	+
Додаткові послуги	+	-	-
Нав’язлива реклама та чат-бот	-	+	-
Наявність всієї інформації	+	+	-

1.4 Постановка задачі. Технічне завдання на розробку

Основне завдання кваліфікаційної роботи є створення туристичного веб-сервісу, який матиме можливість підібрати тур, побачити та прочитати потрібну інформацію, прочитати відгуки користувачів та поділитися своїм досвідом. Для виконання поставленої мети необхідно послідовно вирішити наступні задачі:

– Проаналізувати існуючі туристичні веб-сервіси, веб-сайти, веб-портали тощо;

– Знайти та визначитись з концепцією свого веб-сервісу;

– Врахувати думки користувачів задля запобігання їх втрати на власному веб-сервісі;

– Створення дизайну веб-сервісу зі зручним та зрозумілим інтересом;

– Адаптувати веб-сервіс для комп'ютерів, планшетів та мобільних телефонів;

– Виконати роботу над створенням HTML розмітки для веб−сайту;

– Стилізувати сайт та адаптувати стилізацію для всіх пристроїв;

– Створити Web-API;

– Створити БД;

– Зробити підключення БД до Web-API та веб-сервісу;
– Виконати тестування функціоналу та адаптивності веб-сервісу.

1.5 Висновки

Проаналізувавши існуючі веб-сервіси було зроблено висновки про недоліки та переваги цих сервісів, обрано концепцію побудови дизайну та структуру. Проаналізувавши мови програмування було обрано JavaScript для клієнтської частини та C# для серверної. Важливою частиною кожного веб-застосунку є база даних. Я обрав РСУБД Microsoft SQL Server тому що:

– Зручна в використанні;
– Є безкоштовним продуктом в навчальних цілях;
– Підтримка більшості ОС;
– Зрозумілі засоби управління.

РОЗДІЛ 2 ЕТАПИ РОЗРОБКИ. ПРОГРАМНІ РІШЕННЯ ТА АРХІТЕКТУРА ВЕБ-СЕРВІСУ

Проаналізувавши задачу та засоби, за допомогою яких можна було вирішити цю задачу, було обрано програмний комплекс веб-технологій. Головною перевагою є те, що клієнт не залежить від операційної системи користувача. Ще однією вагомою перевагою є універсальність і можливість використання на будь-яких пристроях.

2.1 Вибір архітектури побудови веб-сервісу

Для реалізації я використовував архітектуру, яка складається з трьох компонентів: база даних, сервер та клієнт. Схема зображена на рисунку 2.1.

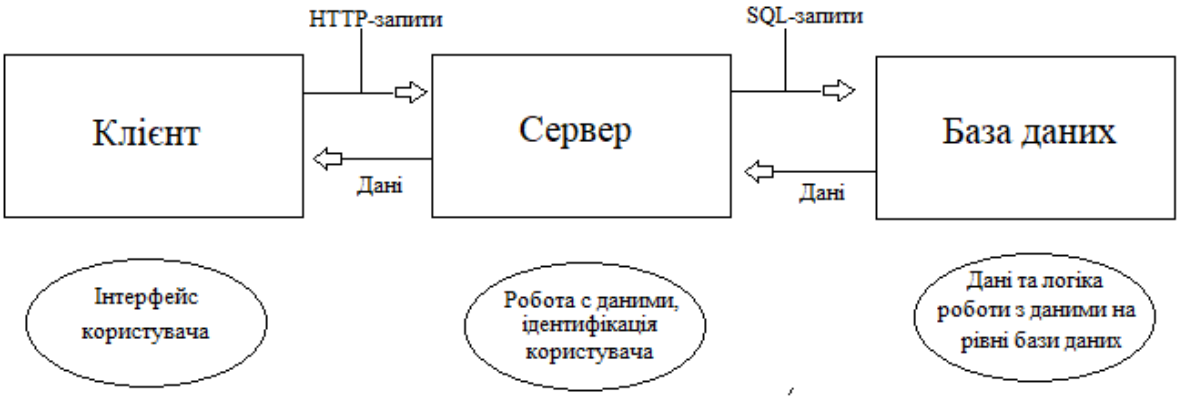


Рисунок 2.1 – Архітектура побудови веб-сервісу

2.1.1 Опис архітектури серверу

Для реалізації серверу я розробив прикладний програмний інтерфейс – API.

Інтерфейс прикладного програмування (API) - це набір визначень підпрограм, протоколів взаємодії та інструментів для розробки. API надає можливість використовувати та власноруч створювати інструменти для власного створення ПЗ, оскільки розробник може брати інші готові об’єкти до власного продукту за правилами взаємодії. API може бути для веб-систем, операційних систем, баз даних, бібліотек програмного забезпечення.

При використанні API у контексті веб-програмування, прикладний програмний інтерфейс зазвичай визначається набором повідомлень запиту HTTP та структурою повідомлень відповідей. Повідомлення можуть бути в інших форматах, як правило, XML чи JSON. Доступ з однієї або декількох загальнодоступних точок.

Кінцеві точки - це важливі частини праці з веб-інтерфейсами на стороні сервера, оскільки вони вказують, де розміщуються дані, куди матимуть змогу достукатись сторонні програми. Як правило HTTP-запити надсилаються на URI, через який здійснюється доступ і отримується відповіді. Кінцеві точки мають бути статичними, інакше від програмного забезпечення, яке працює з ним, не гарантується нормальної роботи [11]. У випадку зміни місця розташування ресурсу, а разом з ним і кінцевої точки, то програмне забезпечення, яке було створене до цього буде перервано, оскільки необхідний ресурс не буде знайдено в одному місці.

Інтерфейси Web 2.0 часто будують взаємодії, основані на таких технологіях, як REST та SOAP. Веб-інтерфейси REST, як правило, засновуються на методах HTTP для доступу до ресурсів, які створюють JSON або XML для передачі ресурсів за допомогою кодованих URL параметрів. До того, веб-API на основі SOAP використовує перевірку XML для забезпечення структурної цілісності повідомлень, застосовуючи схеми XML, надані документами WSDL. Документ WSDL чітко визначає повідомлення XML та транспортні посилання веб-служб [12].

Під час застосування деяких веб-API, які потребують ідентифікації програмного забезпечення, що викликається, потрібно надати ключ API. Ключ інтерфейсу прикладного програмування (API-ключ) - це код, наданий комп'ютерними програмами, який викликає інтерфейс прикладного програмування (API) на веб-сайті для ідентифікації програми, що викликається, її створювача або людини, яка використовує його. Ключі API використовуються для моніторингу та контролю використання API, наприклад, для запобігання зловмисному використанню або зловживанню API (як це визначено умовами

обслуговування). Ключ API зазвичай діє як унікальний ідентифікатор та секретний токен для автентифікації, і зазвичай має набір прав доступу до пов'язаного API. Ключі API мають можливість створюватись на універсальному унікальному ідентифікаторі (UUID), щоб кожний користувач міг бути унікальним.

2.1.2 Обмін даними за допомогою JSON

JSON (JavaScript Object Notation) - набір відструктурованих однотипних даних, яким можна легко обмінюватись, створювати його, зчитувати та записувати дані. Він базується на стандарті ECMA-262, третя версія, і на підґрунті мови програмування JavaScript. JSON - це текстовий формат, який не залежить від мови створення, але він використовує відомі правила для розробників, які програмують на C, C ++, C #, Java, JavaScript, Perl, Python та інших. Тому такі властивості показують, що JSON є чудовою мовою для обміну даними.

Є два структурні дані, на яких базується JSON:

– Встановити пару ключ / значення. Ця концепція була реалізована у різних мовах як, словники, хеші, списки імен, об'єкти, записи, структури або допоміжні масиви.

– Відсортований список значень. У багатьох мовах він застосовується як масив, вектор, список або послідовність.

Це глобальні інформаційні структури. На даний момент підтримка здійснюється майже усіма сучасними мовами програмування. Керуючись такими структурами можна створювати різноманітні формати, які не залежать від мови написання.

Позначення JSON виглядає так:

Об’єкт - набір пар ключ - значення. Об'єкт обов’язково розпочинається фігурною дужкою та закривається. Після кожного ім'я потрібно поставити двокрапку, а пари ключ - значення потрібно розділити комами.

Використовує JSON extension.json. Цей формат легко передається між веб-сервером та клієнтом або браузером. JSON створений щоб замінити XML, зробити його більш доступним та легким.

2.2 Засоби створення веб-сервісу

На кожному архітектурному рівні використовувались різні технології та різні мови програмування. Для серверної частини використовувалась мова програмування C# та технологія ASP .Net Core Web API. Для клієнтської частини було використано мову програмування JavaScript, мова текстової розмітки HTML5, та CSS3 для стилізації. Базою даних служить РСУБД Microsoft SQL Server, також використовувався інструмент Microsoft SQL Server Management Studio для конфігурування і управління структурними елементами Microsoft SQL Server.

2.2.1 Мова програмування C#

C # - це мова з подібним до C синтаксисом і в цьому відношенні близька до C ++ та Java. Отже, якщо ви знайомі з однією з цих мов, вам буде легше вивчити C #.

C # є об'єктно-орієнтованим і в цьому плані багато чому навчилась від Java та C ++. Наприклад, C # підтримує поліморфізм, спадковість, перевантаження оператора та статичне введення тексту. Об'єктивний підхід дозволяє вирішити проблему створення великих, але в той же час гнучких, масштабованих та надійних додатків [9].

Зазвичай, під C # загалом розуміють технології платформи .NET (Windows Forms, WPF, ASP.NET, Xamarin). І навпаки, коли люди говорять .NET, вони зазвичай мають на увазі C #. Однак навіть якщо ці поняття пов’язані, їх неправильно співставляти. Мова C # була створена спеціально для роботи з платформою .NET, але сама концепція .NET дещо ширша.

Структура .NET забезпечує потужну структуру для побудови додатків. Можна виділити наступні основні ознаки:

– Багатомовна підтримка. Платформа базується на Common Language Runtime (CLR), за допомогою якої .NET підтримує кілька мов: VB.NET, C ++, F # поряд із C #, а також різні діалекти інших мов, підключених до .NET, наприклад Delphi. NET [10]. На момент компіляції він зосереджений у збірці Intermediate Language (CIL), яка є різновидом збірки платформи .NET на будь-якій з цих мов. Тому за певних умов ми можемо створювати окремі модулі програми окремими мовами.

– Крос-платформа. .NET портативна (з деякими обмеженнями). Наприклад, остання версія платформи .NET 5 в даний час підтримується в більшості сучасних операційних систем Windows, MacOS та Linux. Використовуючи різні технології на платформі .NET, ви можете розробляти програми на C # для Windows, MacOS, Linux, Android, iOS, Tizen для різних платформ.

– Сильна бібліотека класів. .NET надає бібліотеку класів для всіх підтримуваних мов. Незалежно від того, пишемо ми в текстовому редакторі C #, чаті чи на складному веб-сайті, ми використовуємо бібліотеку класів .NET.

– Різноманітність технологій. Час обробки загальної мови є основою CLR, а базовою бібліотекою класу є технологія, яку розробники використовують для створення додатків. Наприклад, ADO.NET та Entity Framework Core призначені для роботи з базами даних у цьому стеці технологій. Створювати графічні додатки з розширеним інтерфейсом із технологіями WPF та UWP, створювати прості графічні додатки - Windows Forms. Розробити мобільний додаток - Xamarin. Для створення веб-сайтів та веб-додатків - ASP.NET тощо.

Додайте до цього постійно зростаючу популярність Blazor, який працює над .NET і дозволяє створювати веб-додатки як на сервері, так і на стороні клієнта. А в майбутньому він підтримуватиме створення мобільних додатків і, можливо, настільних додатків.

Продуктивність. У деяких тестах веб-програми .NET 5 перевершують веб-програми, побудовані за іншими технологіями, у ряді категорій. Додатки .NET 5 в основному мають високу продуктивність.

Варто також відзначити особливості мови C # та середовища .NET, такі як автоматичний збір сміття. Отже, в більшості випадків, на відміну від C ++, нам не доведеться турбуватися про звільнення пам’яті. Згаданий вище загальномовний час роботи (CLR) активує збирач сміття та очищає пам’ять.

2.2.2 ASP .Net Core Web API

WebAPI - це фреймворк, або простіше кажучи інструмент, який допомагає створювати сервіси HTTP.

Є дві основні причини, які заохочують користувачів використовувати веб-API замість RESTful-сервісів:

– Веб-API розширює підхід TDD (Test Data Driven) на розробку RESTful сервісів.

– Якщо ми хочемо розробити служби RESTful у WCF, вам знадобиться безліч налаштувань конфігурації, схеми URI, угоди та кінцеві точки для розробки служб RESTful за допомогою веб-API [12].

Навіщо обирати веб-API?

– Він використовується для створення простих служб HTTP на основі SOAP

– Він заснований на HTTP і його легко ідентифікувати, відобразити та використовувати в режимі REST.

– Він ідеально підходить для пристроїв з обмеженою пропускною здатністю, таких як смартфони, так як в нього проста архітектура.

Веб-API пропонує спосіб створення програми ASP.NET, призначеної для роботи в стилі REST. Архітектура REST передбачає використання таких методів або типів запитів HTTP для зв'язку з сервером:

– Get

– Post

– Put

– Delete

Загалом, стиль REST, корисний при створенні різних типів односторінкових сайтів, які використовують спеціальні фреймворки JavaScript, такі як Angular, React або Vue.js. В основному, Web API - це веб-служба, до якої можуть звертатися інші програми. Крім того, ці програми можуть відображати будь-які технології та платформи, будь то веб-програми, мобільні або настільні клієнти.

Приклад контролера на ASP .Net Core Web API:

```
// GET api/<OneTourController>/5
[HttpGet("{id}")]
public IActionResult Get(int id)
{
    List <OutOneTour> tours = new List<OutOneTour>();
    string sql = @$"SELECT countryName, days, price, name, stars, sDescribe, lDescribe, location, AllPhotos, All
Cities, AllReviews
                FROM Tours, CountryCities, Countries,
                 (SELECT Tours.id_tour TourId, STRING_AGG(photo, ';') AllPhotos
                 FROM Tours, Photos
                 WHERE Tours.id_tour = Photos.id_tour
                    and Tours.id_tour = 1
                 GROUP BY Tours.id_tour) PhotoTable,
                 (SELECT Tours.id_tour TourId, STRING_AGG(cityName, ' - ') AllCities
                 FROM Tours, Cities
                 WHERE (Tours.id_city_in = Cities.id_city
                  or Tours.id_city_out = Cities.id_city)
                  and Tours.id_tour = 1
                 GROUP BY Tours.id_tour) CityTable,
                 (SELECT id_tour TourId, STRING_AGG(CONCAT(login, ':', review), ';') AllReviews
                 FROM Reviews,Users
                 WHERE Reviews.id_user = Users.id_user
                  and id_tour = 1
                 GROUP BY id_tour) ReviewTable
                WHERE PhotoTable.TourId = Tours.id_tour
```

```
                    AND CityTable.TourId = Tours.id_tour
                    AND ReviewTable.TourId = Tours.id_tour
                    AND Tours.id_city_in = CountryCities.id_city
                    AND CountryCities.id_country = Countries.id_country";
        using (SqlConnection connection = new SqlConnection(DBConn.ConnStr))
        { try
          { connection.Open();
            SqlCommand command = new SqlCommand(sql, connection);
            using (SqlDataReader reader = command.ExecuteReader())
            {  if (!reader.HasRows)
                 return NotFound(new { errorText = "Тур не знайдено" });
               while (reader.Read())
               {
                 tours.Add(new OutOneTour()
                 {
                   CountryName = reader.GetString(0),
                   Days = reader.GetByte(1),
                   Price = reader.GetInt32(2),
                   Name = reader.GetString(3),
                   Stars = reader.GetByte(4),
                   SDescribe = reader.GetString(5),
                   LDescribe = reader.GetString(6),
                   Location = reader.GetString(7),
                   AllPhotos = reader.GetString(8),
                   AllCities = reader.GetString(9),
                   AllReviews = reader.GetString(10)
                 }); } }}
            catch (Exception e)
            {
              string str = e.Message;
              return (new ObjectResult(new { errorText = "Виникла помилка при роботі з базою даних" }) { StatusCo
de = 503 });
            }  }
        return Ok(tours);
      } }}
```

2.2.3 Мова програмування JavaScript

JavaScript - це інтерпретована мова програмування з об’єктно-орієнтованими можливостями. Синтаксично мова JavaScript схожа на C, C ++ та Java у таких конструкціях програмування, як оператор if, цикл while та оператор &&. JavaScript - це нетипізована мова, тобто їй не потрібно визначати типи змінних. Об'єкти в JavaScript відображають властивості властивостей до будь-яких значень [14]. Таким чином, вони більше схожі на асоціативні масиви Perl, ніж структура C або об'єкти C ++ або Java. Механізм JavaScript більше схожий на об’єктно-орієнтований механізм успадкування Self і повністю відрізняються від механізму успадкування C ++ та Java. Як і Perl, JavaScript також є декодованою мовою та деякими її інструментами, такими як регулярні вирази та інструменти для роботи з масивами, реалізовані за аналогією мови Perl.

Ядро JavaScript підтримує прості типи даних, такі як цифри, рядки та булеві значення. Крім того, він включає масиви, дати та об'єкти регулярних виразів.

Як правило, JavaScript використовується у веб-браузерах, і розширення його можливостей шляхом вставки об’єктів дозволяє користувачеві взаємодіяти, керувати веб-браузером та змінювати вміст документа, що відображається у вікні веб-браузера. Ця версія JavaScript запускає сценарії, вбудовані в HTML-код веб-сторінок. Зазвичай ця версія називається клієнтським JavaScript, щоб підкреслити, що сценарій виконувався на комп'ютері клієнта, а не на веб-сервері.

Мова JavaScript та типи даних, які вона підтримує, базуються на міжнародних стандартах для найкращої взаємодії. Деякі частини JavaScript на стороні клієнта формально стандартизовані, інші частини стали стандартними в народі, але деякі частини є специфічними. Сумісність інструментів JavaScript у різних браузерах приносить багато проблеми для програмістів на JavaScript.

Технологія AJAX

AJAX (Asynchronous Javascript And Xml) - це технологія доступу до сервера без перезавантаження сторінки. Як результат, час відгуку скорочується,

а веб-програма більше нагадує робочий стіл з точки зору інтерактивності. Хоча назва технології містить літеру X (від слова XML), XML взагалі не потрібно використовувати [18]. AJAX - це зв’язок із сервером із підтримкою JavaScript без перезавантаження сторінки.

По-перше, AJAX корисний для форм та кнопок, пов’язаних із початковими діями: додавання кошика, підписка тощо. Зараз на сайтах такі дії проводяться без перезавантаження сторінки [15]. Прямий пошук - класичний приклад AJAX, який використовується сучасними пошуковими системами. Користувач починає вводити пошукову фразу, а JavaScript витягує з сервера список можливих доповнень і пропонує можливі варіанти як на рисунку 2.2.

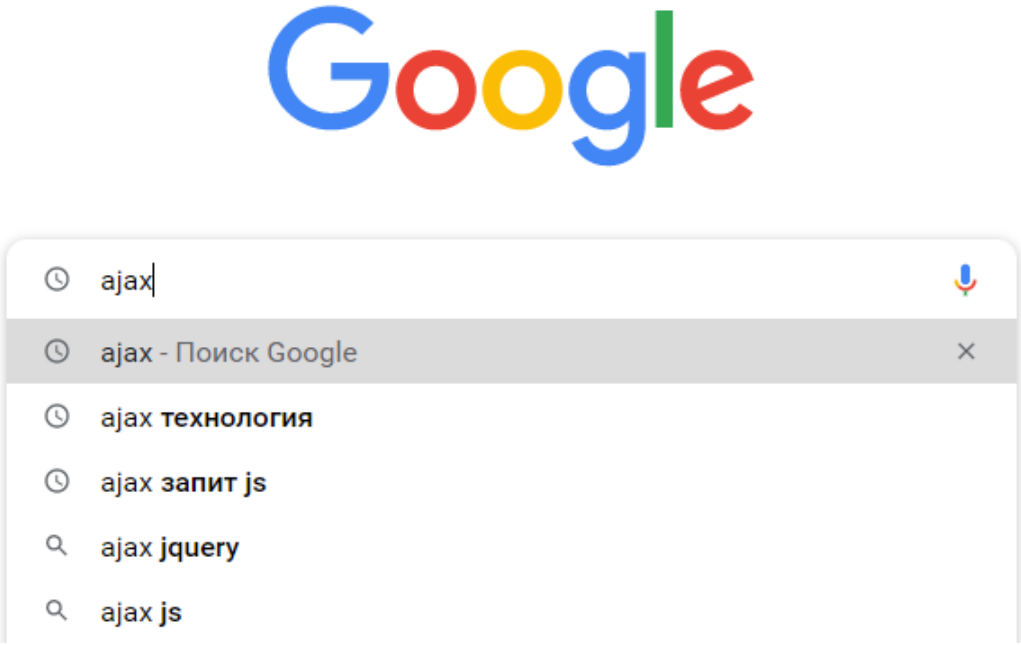


Рисунок 2.2 Приклад дії технологія ajax

Часто використовувані формати:

– JSON - для надсилання та отримання структурованих даних та об'єктів.

– XML - якщо з якихось причин сервер працює у форматі XML, ви можете ним скористатися, є інструменти.

– HTML / текст - ви можете просто завантажити HTML-код або текст із сервера і відобразити його на сторінці [16].

– Двійкові дані та файли зустрічаються рідше; Сучасні браузери мають зручні для них інструменти. Підведемо підсумки в порівняльній таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця використання технології Ajax

Звичайна модель	AJAX модель
HTTP-запит відправляється з веб-браузера на сервер	Браузер створює виклик JavaScript, який потім активує XMLHttpRequest
Сервер отримує і в наслідок витягує дані	В фоновому режимі веб браузер створює HTTP-запит до сервера
Сервер відправляє запрошені дані на веб-браузер	Сервер отримує, витягує і відправляє дані назад на веб-браузер
Веб-браузер отримує дані і перезавантажує сторінку для їх відображення	Веб-браузер отримує запрошені дані, які будуть відображатися на сторінці. Перезавантаження не потребується

2.2.4 РСУБД Microsoft SQL Server

Microsoft SQL Server - це реляційна система управління базами даних, розроблена корпорацією Microsoft. Основною мовою запитів є спільна розробка Microsoft та Sybase Transact-SQL. Transact-SQL - реалізація розширеного стандарту ANSI / ISO мови структурованих запитів SQL [13]. Використовується для роботи з базами даних, починаючи від персональних даних і закінчуючи великими базами даних для підприємства.

Переваги SQL Server:

– РСУБД масштабується, тому ви можете працювати з нею на ноутбуках або потужній мультипроцесорній техніці. Процесор може одночасно обробляти велику кількість запитів.

– Розмір сторінок до 8 кб, тому дані отримуються швидко, зручно зберігати докладні та складні дані. Система має динамічне блокування, що дозволяє обробляти транзакції інтерактивно.

– Щоденні адміністративні завдання автоматизовані: управління блокуваннями, пам'ять, коригування розміру файлу. Система має продумані налаштування, ви можете створити профіль користувача.

– Є пошук фраз, тексту, слів, ви можете створити ключові індекси.

– SQL Server має реплікацію в Інтернеті, забезпечується синхронізація. Існує повноцінний веб-помічник для форматування сторінок.

– Сервер інтерактивного аналізу інтегрований у систему для прийняття рішень та корпоративної звітності. Існують служби трансформації даних.

– Запити можна створювати англійською мовою, без програмування.

– РСУБД підтримує роботу з іншими продуктами Microsoft: Access, MS Excel.

Недоліки:

– Залежність від робочої області: РСУБД працює лише з Windows.

– Висока вартість програми для комерційного використання.

2.3 Етапи розробки веб-сервісу

Складання технічного завдання

Перший крок - розробка та створення веб-сервісу, конструювання інтерфейсу майбутніх ресурсів та виконання технічної роботи. Важливість технічного завдання для зросту порталу для майбутніх ресурсів не слід недооцінювати. Добре виконане технічне завдання запобігає непорозумінням між клієнтом та підрядником, що може зекономити чимало часу на роботі проекту. Тому ТЗ - це документ, який має бути в обов'язковій формі. Технічне завдання - головний орієнтир та керівництво над діями. Тільки за умови дотримання всіх вимог, визначених у ТЗ, проект може бути завершений [20].

Розробка дизайну веб-сервісу

Як і програмування всього сервісу загалом, робота над його дизайном - це покроковий процес. Перше з чого я почав - це створення дизайн у домашньої сторінки, що починалась з розробки схеми та концепції сторінки. Можна скласти

кілька варіантів, щоб переглянути і обдумати всі та вибрати найбільш вдалий. Тоді підсумкова концепція узгоджується і відбирається та вдосконалюється. Результатом цього етапу є затвердження дизайну для всіх сторінок сервісу в форматі PSD.

Верстка сторінок веб-порталу

Після того, як відбулось кінцеве затвердження усіх сторінок веб-сервісу, відбувається їх верстка. Веб-портал використовує блочну верстку, оскільки віна має більше функцій, ніж таблична і дозволяє робити код компактнішим, тим самим збільшуючи швидкість завантаження веб-сторінки. Крім того, блочна верстка дозволяє створити набагато ефективніший сайт, що нормально відображається в браузерах. Макет виготовлений відповідно до всіх свіжих вимог, перевіряється на дійсність та сумісність із наступними браузерами:

– Internet Explorer,
– Opera,
– Chrome,
– Safari

Результат, отриманий після закінчення робіт етапу, - зверстані html-шаблони всіх сторінок веб-порталу.

Програмування

На основі ТЗ і макетів сторінок веб-сервісу наступає етап втілення в життя функціоналу. Починається з налаштування на роботу всіх систем, ПЗ, компонентів, де вносяться ті чи інші зміни в залежності від запланованого функціоналу, відбувається програмування необхідних модулів. Результатом етапу є готовий веб-сервіс.

Тестування

Перевіряється відповідність веб-порталу описаному в ТЗ функціоналу, коректність відображення верстки у всіх підтримуваних браузерах і відповідність порталу внутрішнім вимогам якості. У разі виникнення недоліків чи багів, розробник має виправити всі знайдені помилки. Якщо після повторного

тестування зауважень не виникає, сервіс можна вважати готовим до експлуатації або подальших дій, наприклад, наповнення контентом чи публікація на хостингу.

2.4 Схема бази даних

База даних (БД) - це організована структура, призначена для зберігання, модифікації та обробки взаємопов’язаної інформації. База даних активно використовується для динамічних веб-сайтів із значним обсягом даних - переважно інтернет-магазинів, порталів, корпоративних сайтів. Такі сайти, як правило, розробляються з використанням серверної мови програмування (наприклад, PHP, C#). Сторінки динамічних сайтів створюються "моментально" в результаті взаємодії скриптів і баз даних після запиту клієнта до веб-сервера.

У контексті бази даних необхідно розглянути поняття системи управління базами даних. СКБД - це набір програмних засобів, необхідних для структурування нової бази даних, її вмісту, редагування вмісту та відображення інформації. Переважно бази даних MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Це приклади клієнт-сервера типу СКБД, такі СКБД є загальними в контексті концепції хостингу. Їх особливості:

- розташування СКБД на сервері з базами даних;
- прямий доступ до бази даних;
- Централізована обробка запитів клієнтів на обробку даних;
- високий рівень надійності, доступності та безпеки;
- Збільшене навантаження сервера.

У свою чергу, для зручності роботи з базою даних вони використовують спеціальний веб-додаток, що дозволяє за допомогою графічного інтерфейсу керувати сервером бази даних, запускати спеціальні команди, а також працювати з таблицями та вмістом бази даних, за відсутності додатка можна використовувати консоль.

Я використовував Microsoft SQL Server Management Studio для адміністрування РСУБД Microsoft SQL Server. Для вирішення своєї задачі я

використав лише 7 таблиць. Задача полягала в запиті з 3 полів – місце вильоту, місце прильоту, дата - вивести всі тури, які будуть в таблиці. Також завантажити всю інформацію про тури та вивести на окрему сторінку детальну інформацію про кожний тур при бажанні користувача. Детальніше схему таблиць можна побачити на рис 2.3.

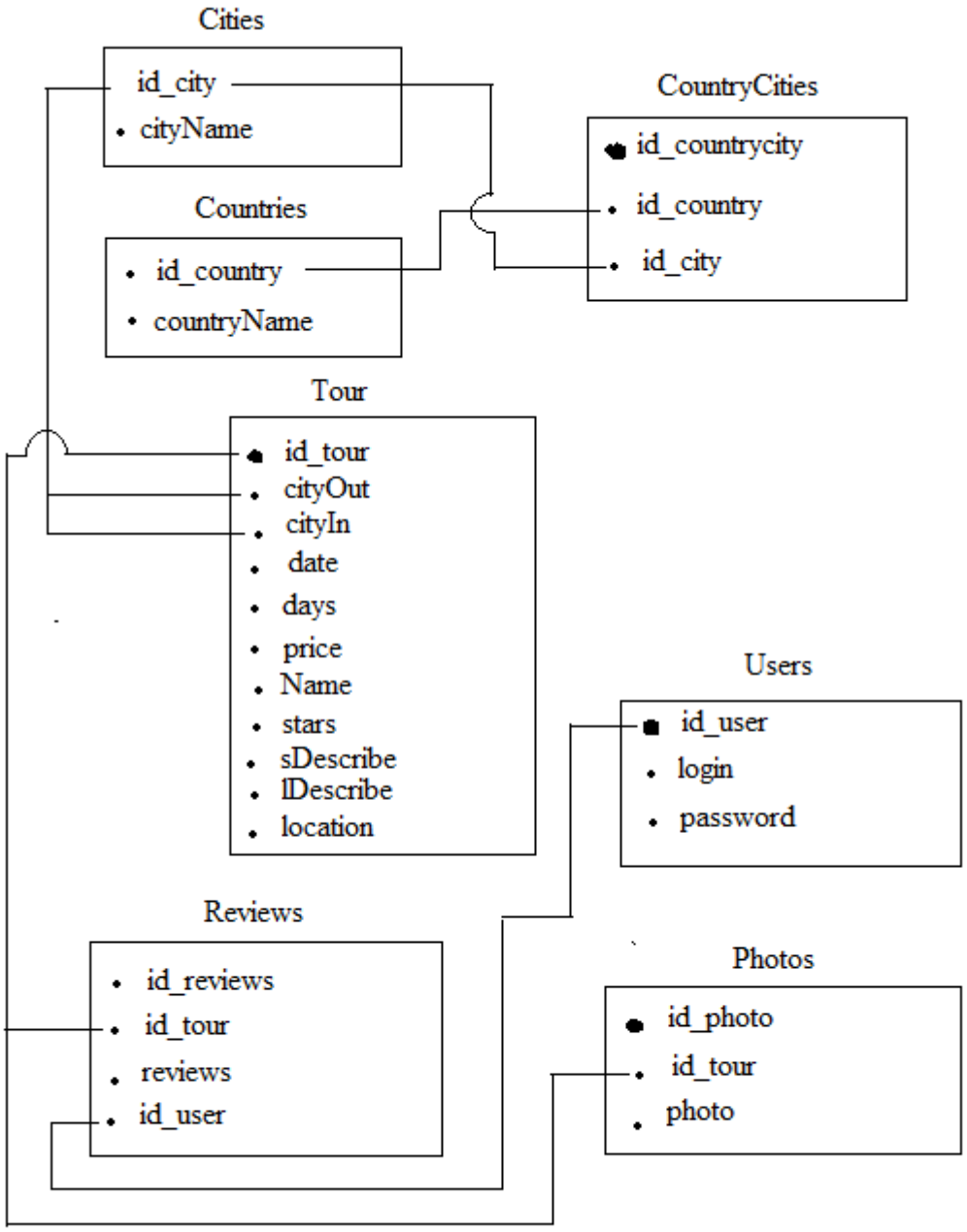


Рисунок 2.3 Схема бази даних проекту

2.5 Програмне забезпечення та структура клієнтської частини

Веб-сервіс створений за допомогою HTML, CSS, мови програмування JavaScript, бібліотеки jQuery, мови програмування C#, технології / фрейворка ASP .NET Core Web API. Використовувались середовища розробки Microsoft Visual Studio 2019, Microsoft Visual Studio Code, Microsoft SQL Server Management Studio та Microsoft SQL Server.

2.5.1 Опис структури файлів

Ці папки пояснюють структуру папок клієнтської частини веб−сервісу. Кожна з них відповідає за окрему частину. Детальний опис папок можливо побачити в таблиці 2.2.

Таблиця 2.2 - Опис структури папок клієнтської частини веб−сервісу

Назва папки	Опис
Коренева папка	Папка в якій знаходяться всі файли які пов’язані з веб−сайтом.
.vscode	В цій папці зберігається json файл, в якому прописані налаштування порта liveServer.
eventsPeriod	Папка в якій зберігаються html файли, які використовуються при AJAX запитах.
fonts	Папка з шрифтами.
photos	Папка з усіма зображеннями, відео та gif.
places	Папка з html документами всіх туристичних направлень.

На рисунку 2.4 можна побачити папки та файли при розробці, коли вони ще не пройшли структурування.

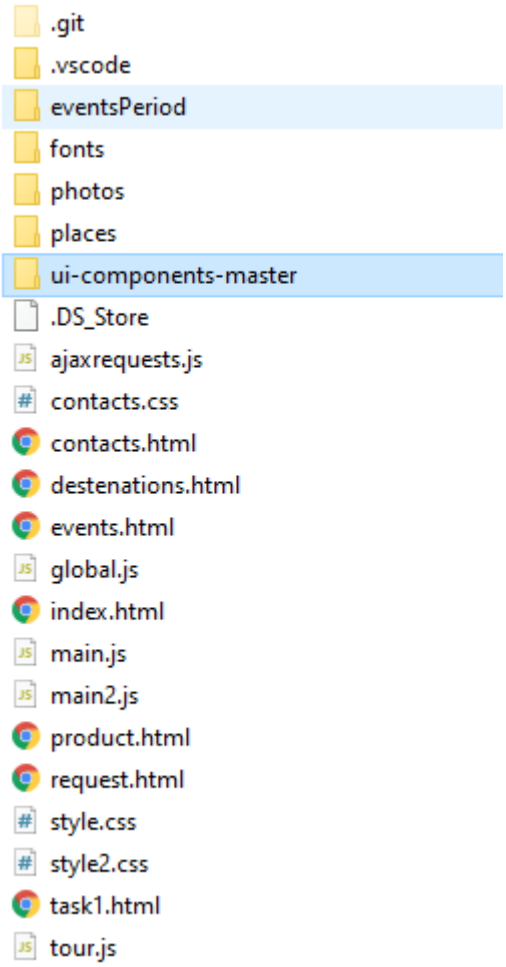


Рисунок 2.4 Структура файлів та папок клієнтської частини

Після структурування файлів та папок можна робити подальші дії, наприклад, завантажити на хостинг.

2.5.2 Структура інформаційних об’єктів

Структура сторінок веб-сервісу була створена завдяки аналізу та тестуванню на зручність вже існуючих тематичних сервісів, детальніше можна побачити на рисунку 2.5.

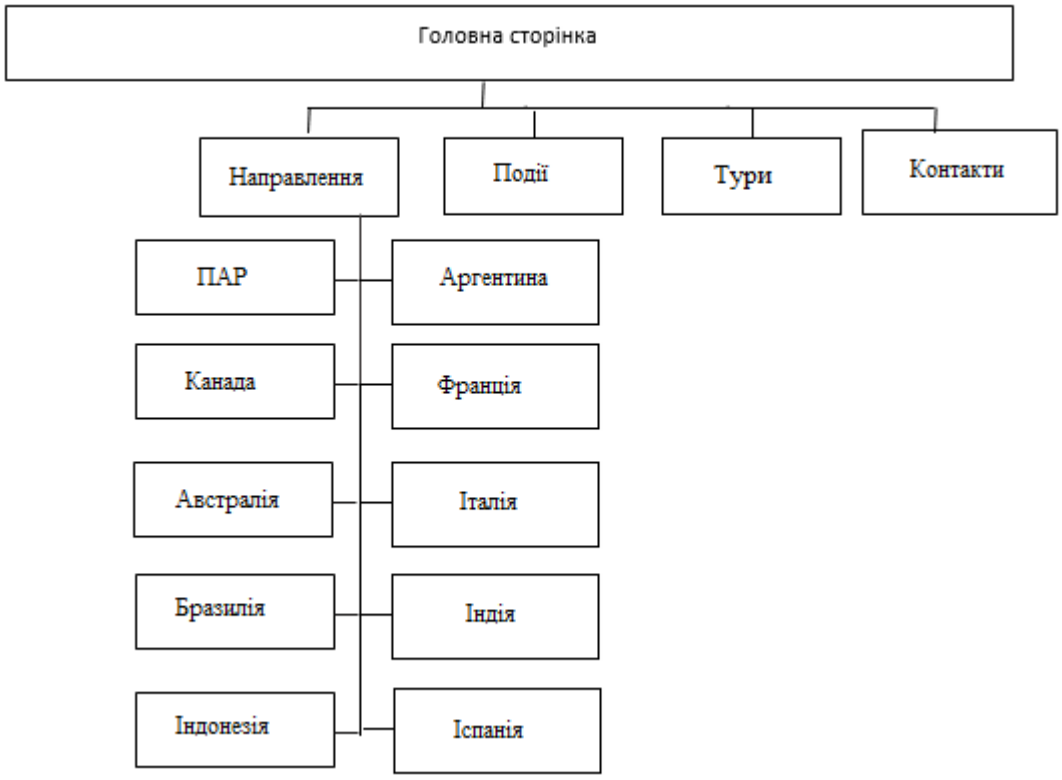


Рисунок 2.5 Структура сторінок веб-сервісу

Існують три основні логічні структури сайту: лінійна модель, "ґрати" та "дерево", вона ж ієрархічна модель. Окрім цього, можна створити будь-яку комбінацію з цих трьох основних моделей, що й дозволяє нам зробити будь-яку логічну структуру розроблюваного веб-сервісу.

Структура сайту використовується для різноманітних інтернет-презентацій, що описують покрокові процедури. Найважливішою перевагою використання цієї моделі організації сайту є прогнозування поведінки відвідувачів сайту, тому дії відвідувачів можна оцінити та спрогнозувати на етапі створення сайту.

Інтернет-магазини дуже часто застосовують саме логічну структуру "ґрати" свого веб-сайту . Ґрати - це двосторонні лінійні структури, в таких структурах завжди присутні і вертикальні, і горизонтальні зв'язки.

Наприклад, якщо список товарів інтернет-магазину класифікований за типом та ціновою категорією, ця структура дозволяє відвідувачам переглядати

товар в обох напрямках (типу та ціни). Як результат, відвідувачі можуть вільно вибирати.

Логічна структура сайту "Дерево" є загальною моделлю організації сайту. Дерево - дозволяє відвідувачам сайту контролювати глибину відвідування сайту за власним бажанням. Відвідувачі можуть отримати доступ лише до сторінок ієрархії верхнього рівня або перейти до нижчого рівня.

Якщо відвідувачам доводиться багато натискати для досягнення кінцевого пункту призначення, ієрархічна структура сайту може бути дуже вузькою. Користувачів дратують нескінченні «клацання», які не приносять очікуваних результатів.

У той же час, дуже велике «дерево», засноване на великій кількості варіантів, відвідувачі можуть витратити багато часу на вивчення запропонованих варіантів. Це також шкодить відвідувачам і залишає сайт.

2.6 Висновки

Проаналізувавши переваги та недоліки, можна зробити висновки, що для туристичного веб-сервісу є правильним обрання тришарової клієнт-серверної архітектури. Ріст потужності клієнтських ПК та смартфонів дозволяє реалізувати роботу системи в рамках моделі клієнт-сервер з «товстим» клієнтом, що є актуальним у галузі веб-додатків. Також були зроблені висновки щодо програмного забезпечення на якому буде розроблятися туристичний веб-сервіс.

Опрацювавши теоретичні відомості про інформаційне наповнення та структуру веб-сторінок, було зроблено висновки щодо цього. Туристичний веб-сервіс буде мати логічну структуру "Дерево". Логічна структура сайту "Дерево" є загальною моделлю організації сайту. Дерево - дозволяє відвідувачам сайту контролювати глибину відвідування сайту за власним бажанням.

Якщо відвідувачам доводиться багато натискати для досягнення кінцевого пункту призначення, ієрархічна структура сайту може бути дуже вузькою. Користувачів дратують нескінченні «клацання», які не приносять очікуваних результатів.

РОЗДІЛ 3 ФУНКЦІОНАЛ ТА ОПИС РОБОТИ СЕРВІСУ

В цьому розділі буде описаний функціонал та інструкція користувачу по навігації веб-сервіса. Веб-сервіс повинен бути адаптивним та однаково відображатися в браузерах, таких як Google Chrome, Mozilla Firefox, Opera та інших.

На сайті повинна міститись інформація про популярні напрямки, про світові свята та фестивалі, інформація про країну, про кожний тур мають бути відомі такі дані: місто вильоту, країна і місто призначення, дата початку туру, кількість ночей, приблизна ціна за одну людину. Додатково може бути наявний опис готелю, кількість зірок.

Дизайн веб-сайту повинен бути стриманим і не містити зайвого. Дизайн має містити кольори, які будуть співпадати на підходити до загальних кольорів сайту. Щоб запобігти введення випадкових даних користувачем потрібно обмежувати вибір дати, надавати перелік можливої інформації та робити підказки до полів.

3.1 Опис структури інтерфейсу

Відповідно до загально-прийнятих стандартів реалізації структури веб-сайтів кожна сторінка складається з трьох головних частин:

– Header (шапка) – блок у верхній частині сторінки сайту. Містить меню та постійно присутній на сторінці. Змінюється в залежності від девайсу з якого зайшов користувач;

– Body (тіло сайту) – основний блок відображення контенту, який змінюється в залежності від сторінки;

– Footer (підвал) – блок в нижній частині сторінки, який постійно присутній на сторінці як і Header.

Але при зміні розмірів екрану, наприклад на телефоні, додається ще одна головна частина. Макет структури інтерфейсу веб-сайту можна побачити на рисунку 3.1.

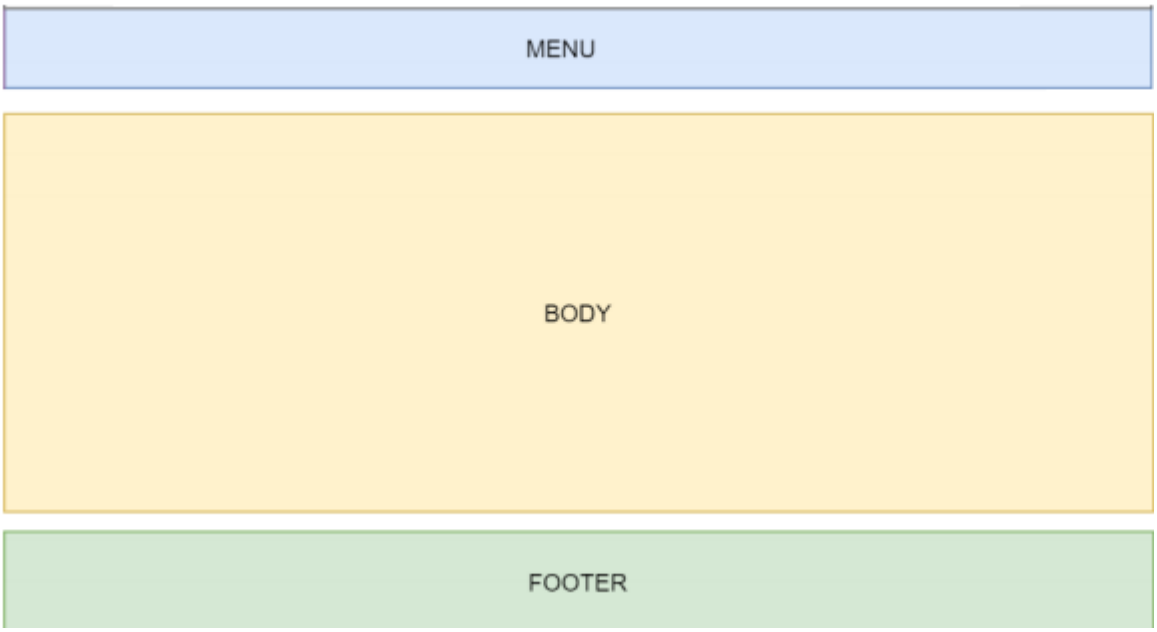


Рисунок 3.1 Структура інтерфейсу веб-сторінок

Зі зміною ширини екрану структура змінюється. На телефона інтерфейс адаптовано під екрани меншого розміру, детальніше можна побачити на рисунку 3.2.

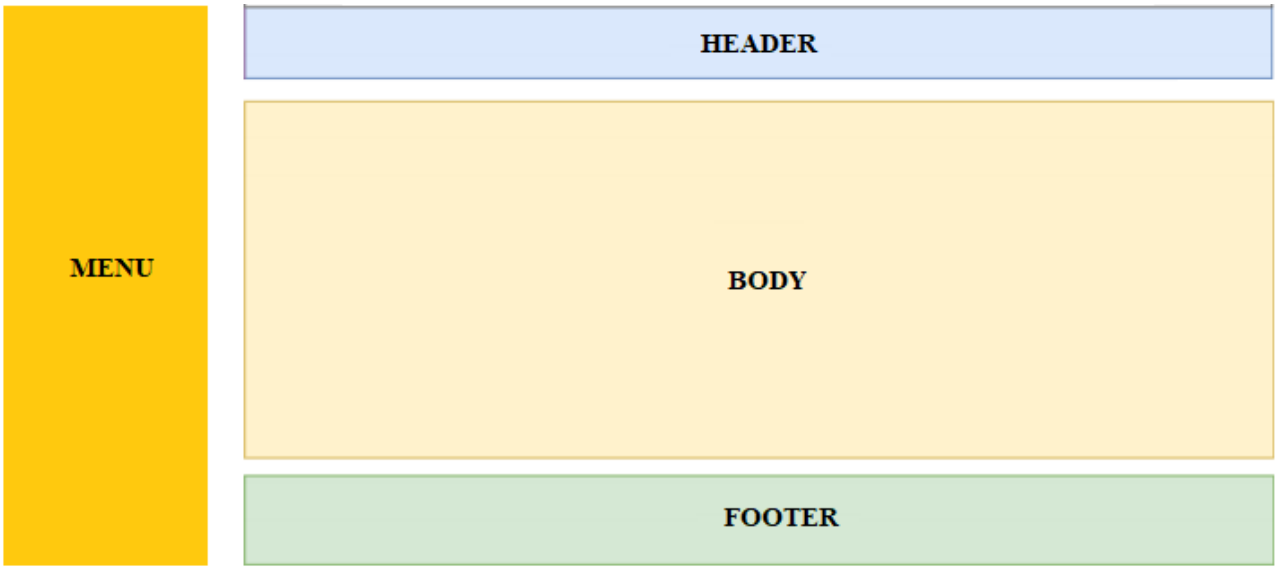


Рисунок 3.2 Структура інтерфейсу веб-сторінок на маленьких екранах

Веб-сервіс має логічну структуру “Дерево”, тому перейти на головну сторінку можна з будь-якої іншої.

Опис головної сторінки веб-сервісу. Головна сторінка відразу під шапкою містить слайдер, на якому основні популярні напрямки вибору користувачів. Нижче розділ головними подіями року та перехід на сторінку з

подіями, до якої можна перейти як через меню так і по кнопці “Всі події 2021”. Далі знаходиться інтерактивна мапа світу та деякі направлення. В підвалі містяться корисні посилання та ім’я розробника веб-сайту.

Направлення. На цій сторінці містяться посилання на сторінки з описом та короткою інформацією про головні туристичні місця кожної з країн.

Події. На сторінці події можна побачити блоки з короткою інформацією та посиланням на повний опис фестивалю чи свята. Також є можливість відфільтрувати по порі року.

Тури. На сторінці тури користувач бачить лише шапку, три поля, які потрібно заповнити для відображення турів та підвал. Коли користувач заповнить ці поля, він побачить тури, які співпадають з фільтрами, які задав користувач. При натисканні на кнопку “Перейти” можна побачити повну інформацію по конкретному туру.

Контакти. На сторінці контакти є номер телефону, можливість відразу подзвонити при натисканні на іконку, посилання на соціальні мережі та пошту.

3.2 Тестування веб-сервісу

Завершальним етапом усього процесу є веб-тестування сайту. Ця процедура відіграє важливу роль у створенні ресурсів, оскільки майбутнє проекту залежить від якості тестування. Практика показує, що багато розробників не приділяють цьому етапу достатньої уваги, покладаючись лише на власні знання та досвід.

Основна мета тестування - перевірити відповідність веб-сервіса вимогам. Етапи тестування веб-сервісу:

- Перевірка оптимізації та функціональності;
- Перевірка зручності користування (юзабіліті);
- Тестування продуктивності;
- Тестування інтерфейсу.

Перевірка оптимізації. Цей тест дасть зрозуміти продуктивність та поведінку пошукової системи. Також перевірка покаже, які частини потребують виправлень.

Перевірка юзабіліті. Цей етап тестування дозволяє перевірити, наскільки зручний сайт для користувача, наскільки легко йому знайти ту чи іншу інформацію. Потрібно провести тест на швидкість завантаження сторінок та вагу сайту.

Основна мета тесту на зручність:

– Переконатися, що веб-сервіс зручний;
– Зрозуміти, наскільки зручна навігація;
– Оцінити, що може бути зайвим у ресурсі.

Тестування продуктивності: завантаження сайту. Важливим тестом є перевірка ефективності. Необхідно знати, чи може система нести певне навантаження.

Перевірити інтерфейс, контрольний список:

– Перевірте відповідність усім стандартам графічного інтерфейсу;
– Тестування з різною роздільною здатністю екрана;
– Перевірка сумісності для всіх браузерів та їх версій (сумісність браузера);
– Тестування інтерфейсу смартфонів, планшетів.

Для тестування веб-сервісу на різні параметри я використав Google PageSpeed Insights, WebPageTest та інструменти “Network”, “Google audits”.

На рисунку 3.3 можна побачити результати тестування за допомогою інструменту “Google audits”.

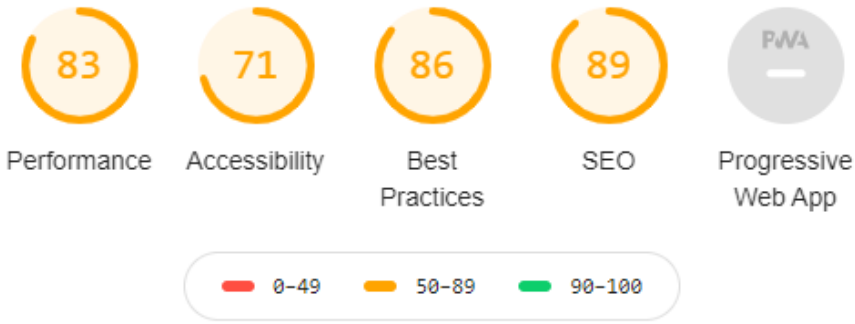


Рисунок 3.3 Тестування веб-сервісу в “Google audits”

Цей тест показує результат що до семантично правильного використання HTML тегів, оцінці швидкості, навантаження на систему та оптимізації всього веб-сервісу. За результатами тестування в інструменті "Google Audits" було отримано наступні значення (рис 3.3). На основі базового аналізу можна прийняти оптимізаційні заходи для підвищення видимості сайту, які призведуть до збільшення числа відвідувачів. На рисунку 3.4 можемо побачити аналіз від Google PageSpeed Insights.

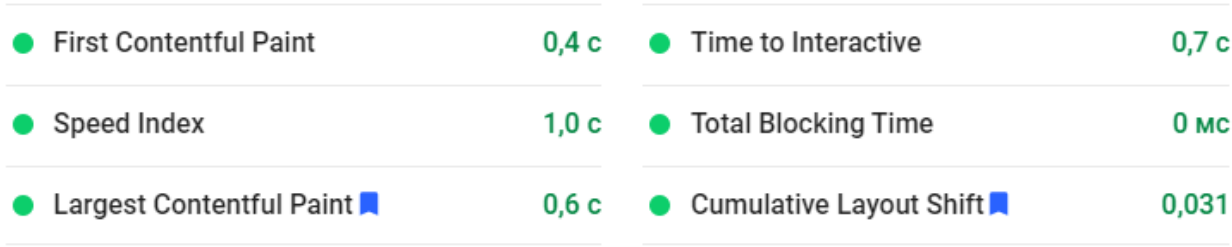


Рисунок 3.4 Аналіз від Google PageSpeed Insights

На основі цього аналізу Google PageSpeed Insights рекомендує зменшити час відповіді сервера, показувати зображення в нових форматах, показувати статичні об’єкти за допомогою ефективних правил кешування.

Наступне тестування покаже швидкість завантаження сайту та його вагу. Тестування швидкості можна побачити на рисунку 3.5. Тестування зроблене за допомогою інструменту “Network” в браузері Google Chrome.

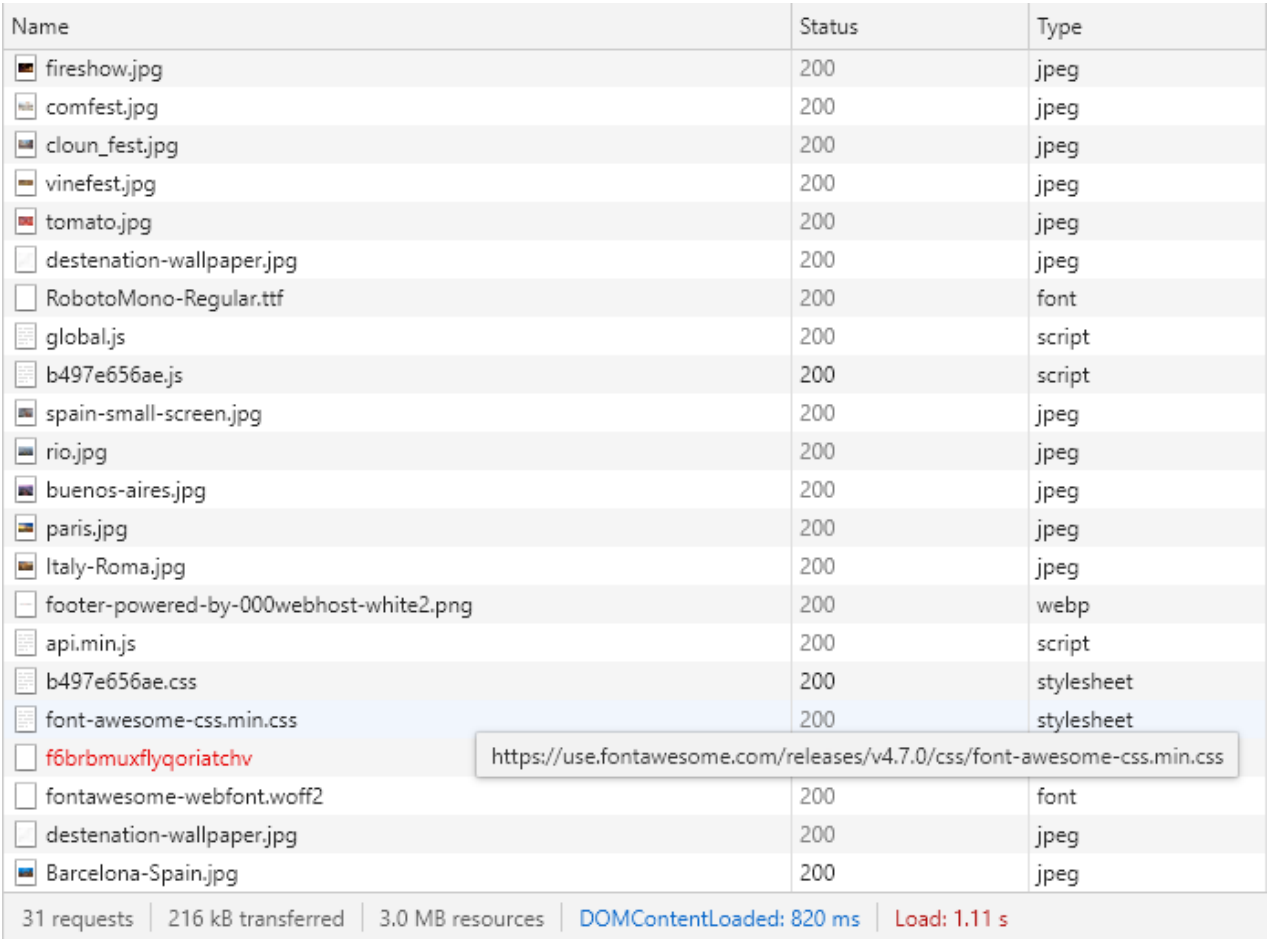
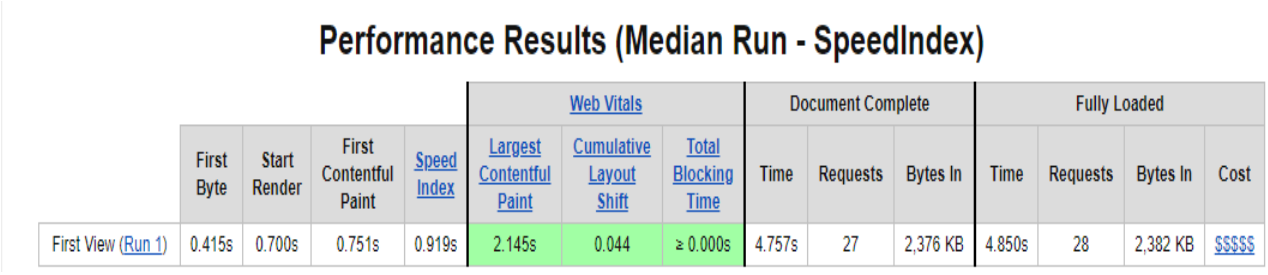


Рисунок 3.5 Результат тестування швидкості та ваги

Дане тестування показує середній час завантаження сторінок з завантаженням всіх зображень, шрифтів, текстів, стилів та скриптів – 1.11 сек. Сайт має вагу 3 MB. Таке тестування дає рекомендації щодо оптимізації завантаження сторінки, збільшення швидкості відгуку.

На рисунках 3.6 та 3.7 можна побачити звіт від WebPageTest.



Performance Results (Median Run - SpeedIndex)

					Web Vitals			Document Complete			Fully Loaded			
	First Byte	Start Render	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time	Time	Requests	Bytes In	Time	Requests	Bytes In	Cost
First View (Run 1)	0.415s	0.700s	0.751s	0.919s	2.145s	0.044	≥ 0.000s	4.757s	27	2,376 KB	4.850s	28	2,382 KB	$$$$$

Рисунок 3.6 Результат продуктивності

MIME Type	Requests
image	19
css	3
font	2
js	2
html	1
flash	0
other	0
video	0

MIME Type	Bytes	Uncompressed
image	2,135,725	2,135,725
font	186,372	186,372
html	91,949	229,025
css	14,911	64,800
js	4,283	10,530
flash	0	0
other	0	0
video	0	0

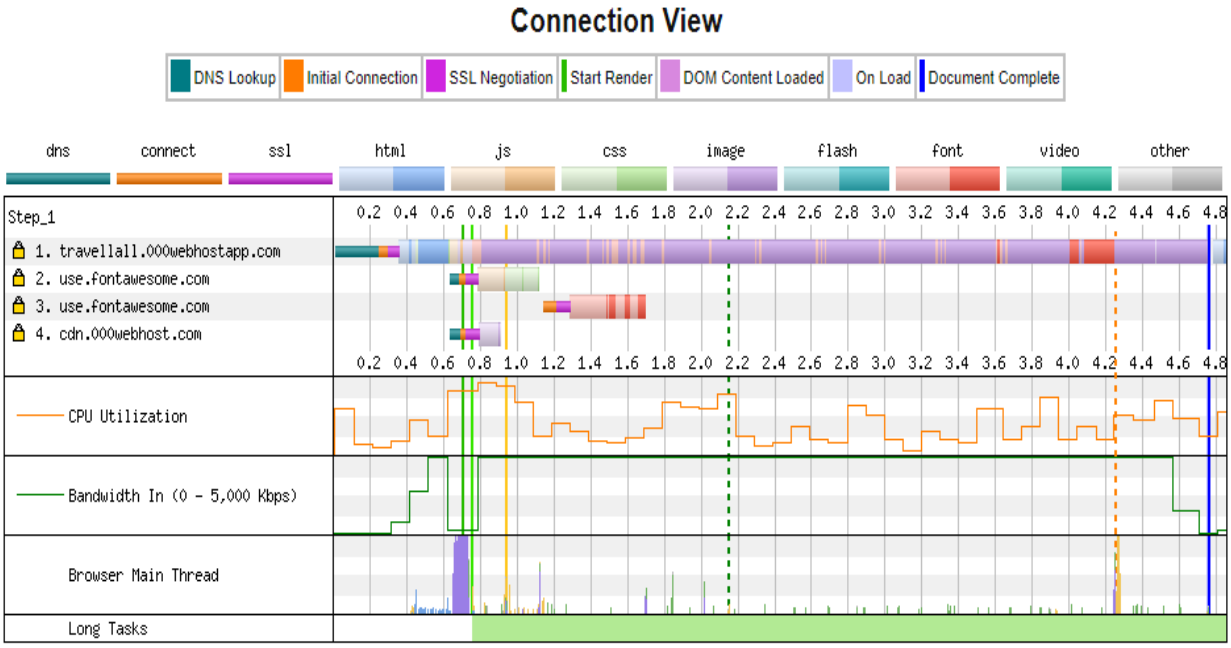


Рисунок 3.7 Розбивка контенту по типу MIME

Можна зробити висновок, що потрібно пришвидшувати завантаження сторінок. Основним є медіа файли, тому потрібно працювати над швидкістю їх передачі та кешування.

3.3 Інструкція користувача

Інструкція користувача містить загальну інформацію про веб-сервіс, опис функціональних можливостей кожної окремо взятої сторінки. Інструкція створена щоб показати можливі шляхи пересування по сайту. При відкритті веб-сервісу, ви попадаєте на головну сторінку (рис.3.8).

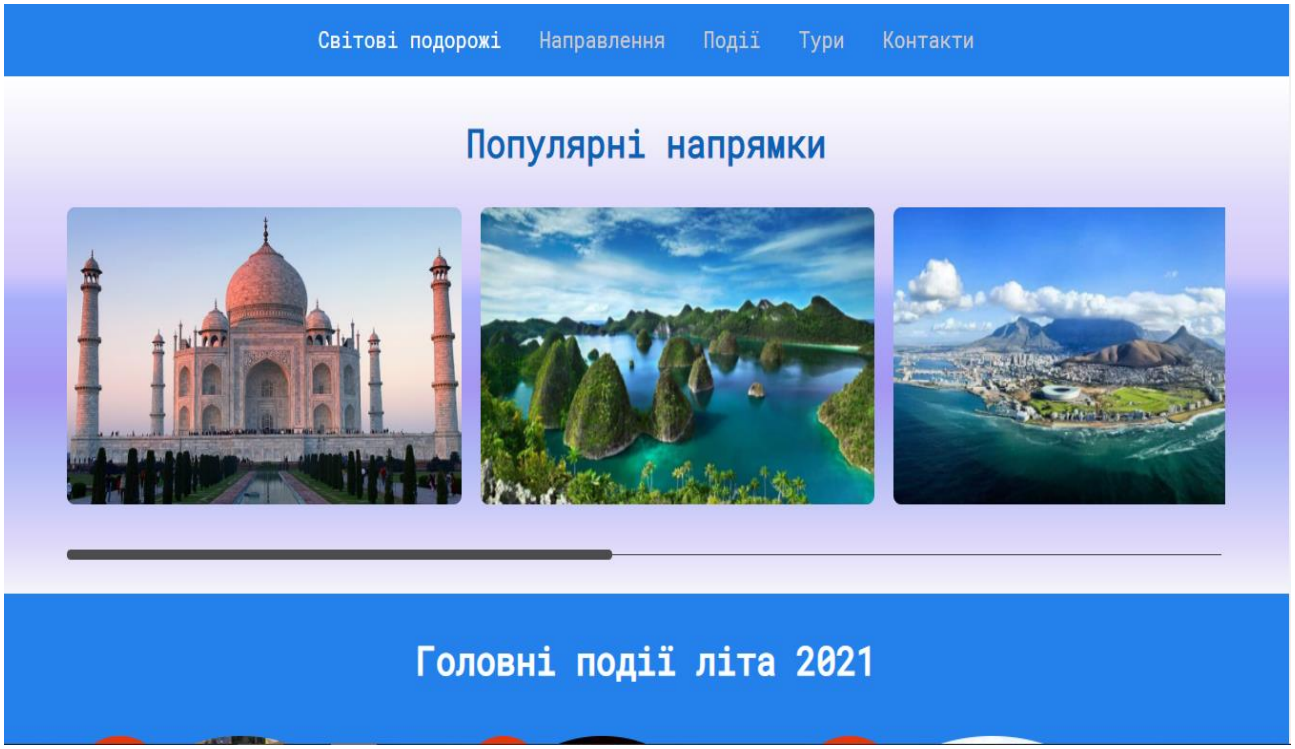


Рисунок 3.8 Головна сторінка веб-сервісу

На цьому рисунку можна побачити меню та популярні туристичні напрямки, нижче головні події цього року та вибрати власний напрямок. При виборі напрямку, бачимо інтерактивну мапу за допомогою якої також можна вибрати напрямок.

Мапа створена за допомогою SVG графіки. Scalable Vector Graphics (скорочено SVG) - специфікація мови розмітки, що базується на XML, та формат файлів для двомірної векторної графіки, як статичної, так і анімованої та інтерактивної. SVG може бути виключно декларативним, або містити описи сценаріїв. Кожна країна малюється окремо, має власне ім’я, ідентифікаційний номер та малюється за допомогою координат. Розділ вибору напрямку та мапи можна побачити на рисунку 3.9.

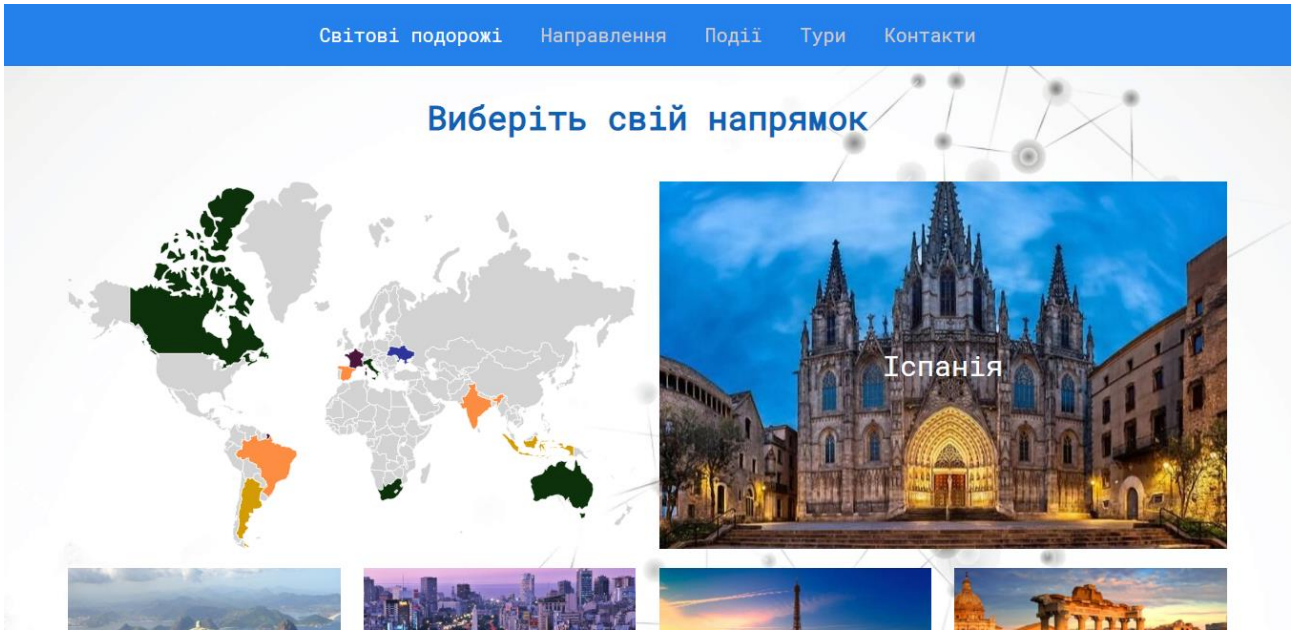


Рисунок 3.9 Розділ з вибором напрямку на головній сторінці

Далі користувач може перейти на сторінку направлення або за допомогою меню, або натиснувши на кнопку <<Всі направлення>>, яка знаходиться на головній сторінці в кінці підрозділу з напрямками. На сторінці з направленнями, Ви можете обрати одну з країн(рис. 3.10).

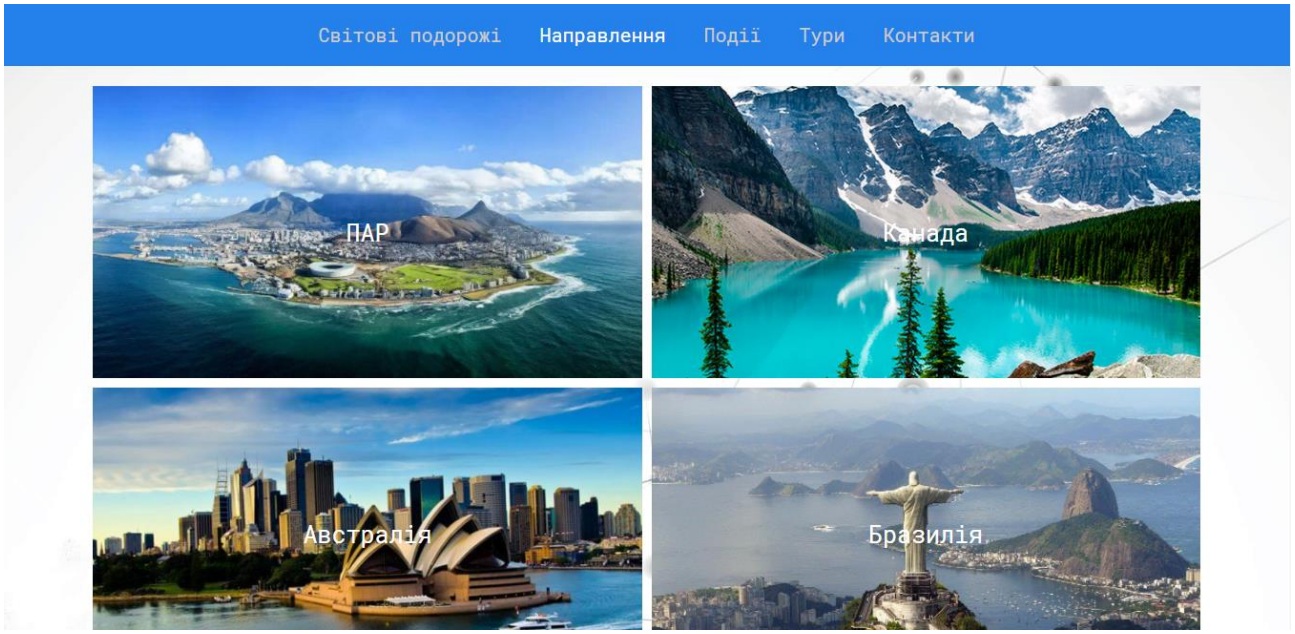


Рисунок 3.10 Веб-сторінка з вибором країни

Вибравши одну з країн та натиснувши на неї, Ви потрапите на сторінку з описом цієї країни та цікавими місцями, які варто відвідати. Приклад такої сторінки можна побачити на рисунку 3.11.

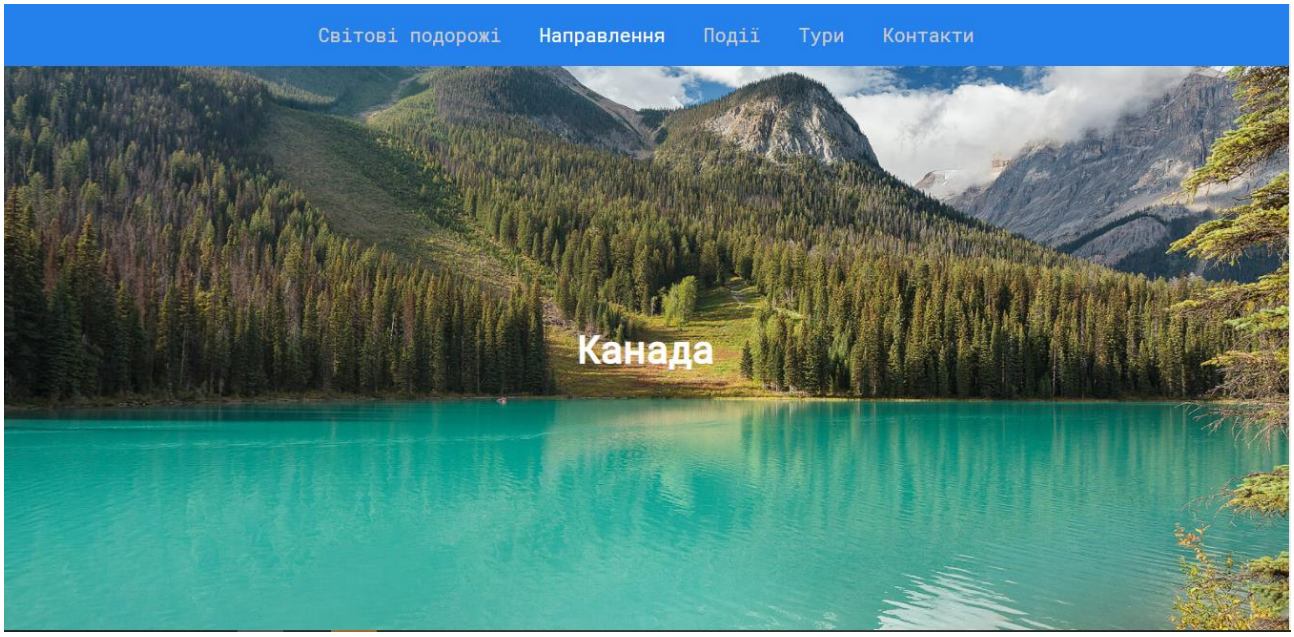


Рисунок 3.11 Веб-сторінка однієї з країн

Нижче ви зможете почитати інформацію та переглянути фото.

Наступний пункт меню – події. Тут можна почитати про найцікавіші події цього року у світі. Ви можете відсортувати події по порі року та побачити добірку подій (рис.3.12).

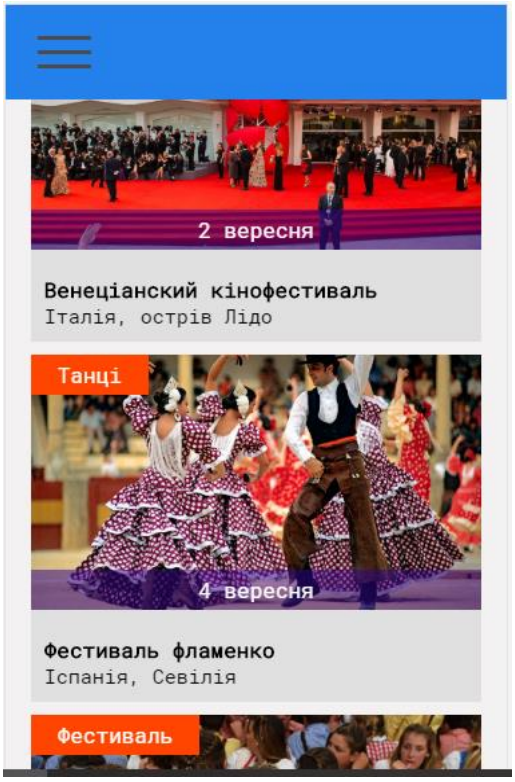


Рисунок 3.12 Веб-сторінка подій (вид з телефону)

Для того щоб підібрати тур потрібно перейти на сторінку Тури. В першу чергу потрібно заповнити поля, які пропонуються задля фільтрації та показу лише тих, які цікаві користувачеві. На рисунку 3.13 можна побачити тури, які завантажились по запиту.

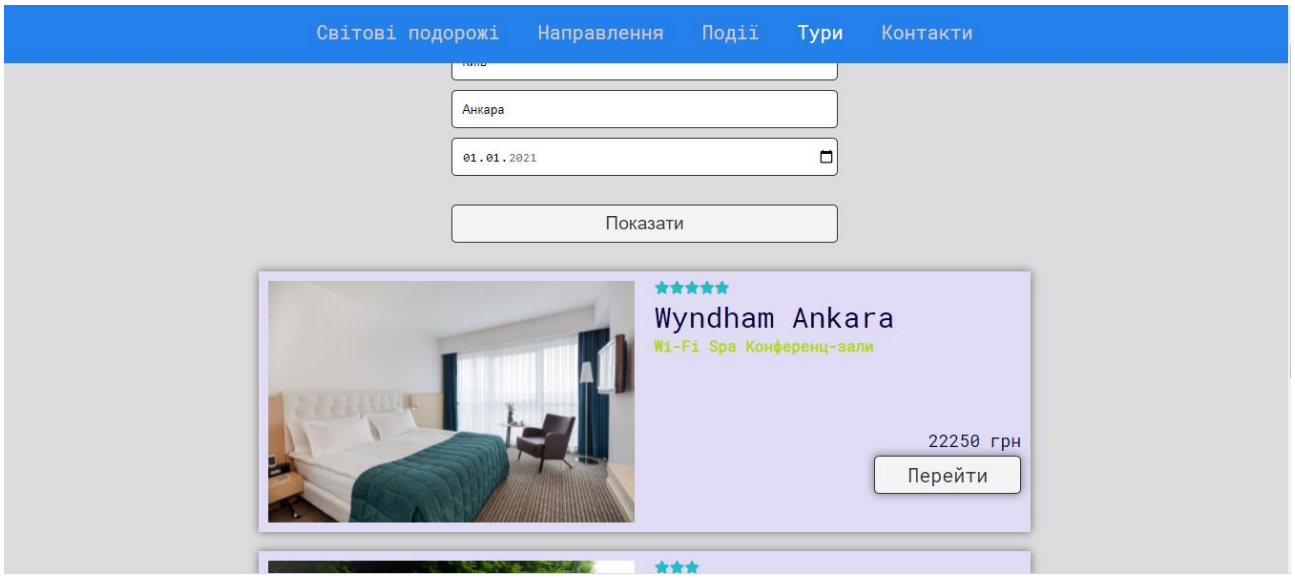


Рисунок 3.13 Веб-сторінка з турами

При натисканні кнопки “Перейти” відкриється сторінка з детальнішим описом конкретного туру та місцезнаходженням на карті. На рисунку 3.14 можна побачити інтерфейс цієї сторінки.

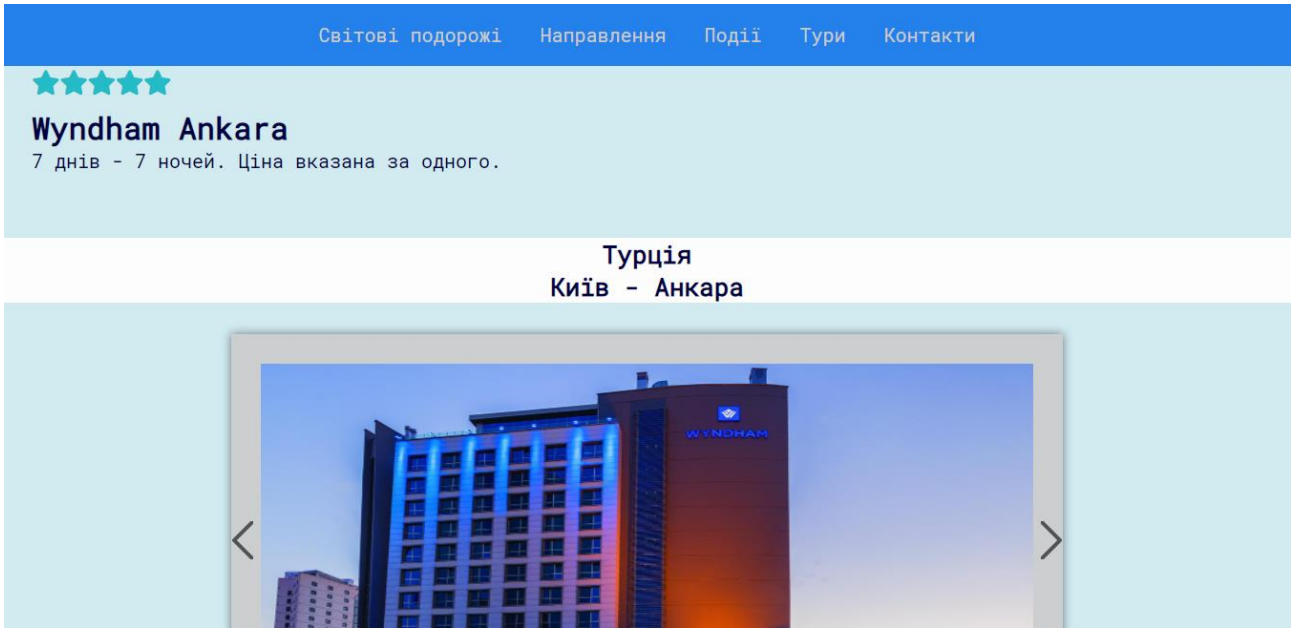


Рисунок 3.14 Веб-сторінка з контактами

І остання веб-сторінка в списку меню контакти. Тут користувач може написати на пошту, в соціальні мережі Instagram та Telegram, також зателефонувати. Цю сторінку можна побачити на рисунку 3.15.

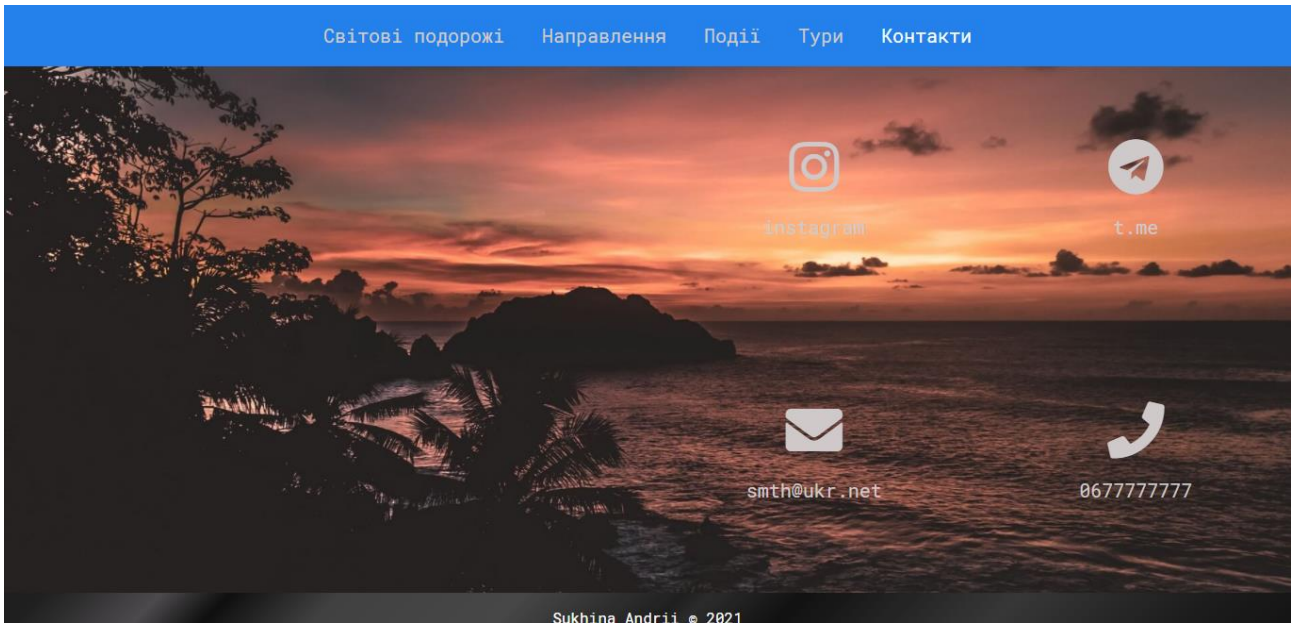


Рисунок 3.15 Веб-сторінка з контактами

При розробці дизайну головним було зручність, простота та зрозумілість можливих шляхів.

ВИСНОВКИ

У результаті виконаної кваліфікаційної роботи було розроблено туристичний веб-сервіс.

Під час виконання кваліфікаційної роботи було досліджено теоретичні основи написання веб-сайтів та окремо досліджено туристичні веб-сервіси. Для дослідження теоретичних основ було взято існуючі програмні забезпечення та теоретичні відомості про функціонал, досліджено мету та ціль туристичного веб-сервісу.

Веб-сервіс був зроблений за допомогою: мови програмування C# та РСУБД Microsoft SQL Server - серверна частина, мови програмування - JavaScript, HTML5 та CSS3 – клієнтська частина.

За результатами досліджень найважливішими в каталозі туристичного веб-сервісу є підбір турів, актуальність новин, достатній обсяг тематичної інформації та висока швидкість завантаження сторінок.

Було протестовано веб-сервіс та зроблено висновки для його покращення. Для тестування веб-сервісу на різні параметри було використано Google PageSpeed Insights, WebPageTest та інструменти “Network”, “Google audits”.

Були проведені наступні тестування веб-сервісу:

– Перевірка оптимізації та функціональності;
– Перевірка зручності користування (юзабіліті);
– Тестування продуктивності;
– Тестування інтерфейсу.

Була виконана мета кваліфікаційної роботи: створити веб-сервіс де можна підібрати тур, мати можливість дізнатися актуальну інформацію, мати можливість прочитати відгуки інших, надати користувачеві можливість дізнатись все в одному місці з метою найкращого вибору для подорожі.

Веб-сервіс може бути покращено наступними шляхами:

– Розробка системи бронювання та оплати;
– Реєстрація користувачів на сайті;

- Створення акційних пропозицій;
- Оповіщення зареєстрованих клієнтів про нові пропозиції;
- Надання можливості більш точної фільтрації.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загальні відомості про веб-сервіси. Веб-сайт. URL: http://www.its.kpi.ua/itm/tkot/Students/Lec3_1-Wf-Implementation (дата звернення 01.05.2021);

2. Що таке XML? Структури даних: веб-сайт. URL: https://www.taina.com.ua/shho-take-xml/ (дата звернення 02.05.2021);

3. Що таке SOAP? Правила передачі даних: веб-сайт. URL: https://uk.photo-555.com/9679948-what-is-soap (дата звернення 02.05.2021);

4. WSDL. Мова опису веб-сервісів: веб-сайт. URL: https://uk.wikipedia.org/wiki/WSDL (дата звернення 02.05.2021);

5. IT в туризмі. Проблеми в українському туризмі: веб-сайт. URL: https://tourlib.net/ (дата звернення 03.05.2021);

6. Туроператор TUI. Туристичний веб-сервіс: веб-сайт. URL: https://www.tui.ua/ (дата звернення 05.05.2021);

7. Туроператор Feerie. Туристичний веб-сервіс: веб-сайт. URL: https://feerie.com.ua/ru (дата звернення 05.05.2021);

8. Туроператор CoralTravel. Туристичний веб-сервіс: веб-сайт. URL: https://www.coraltravel.ua/ (дата звернення 05.05.2021);

9. Введення в C#. Мова C# і платформа .NET Core: веб-сайт. URL: https://metanit.com/sharp/tutorial/1.1.php (дата звернення 07.05.2021);

10. Програмування на C#. Об’єктно-орієнтована мова: веб-сайт. URL: http://youngdeveloper.com.ua/csharp.html (дата звернення 07.05.2021);

11. Клиент-сервер. Основные понятия и особенности клиент-серверной архитектуры: веб-сайт. URL: https://testmatick.com/ru/osnovnye-ponyatiya-i-osobennosti-klient-servernoj-arhitektury/ (дата звернення 10.05.2021);

12. Интервью Asp.Net Web API. Вопросы и ответы: веб-сайт. URL: https://coderlessons.com/tutorials/veb-razrabotka/uchebnik-asp-net/14-interviu-asp-net-web-api-voprosy-i-otvety (дата звернення 11.05.2021);

13. Что такое SQL Server? Плюсы и минусы использования: веб-сайт. URL: https://muzeon.ru/medicina/2912-chto-takoe-sql-server-plyusy-i-minusy-ispolzovaniya.html (дата звернення 16.05.2021);

14. Дэвид Флэнаган. JavaScript. Подробное руководство, 5-е издание: учебник. Санкт-Петербург-Москва: Символ-Плюс, 2008. 992 с. URL: http://kharchuk.ru/JavaScript.pdf (дата звернення 18.05.2021);

15. Що таке Ajax? Введення в Ajax: веб-сайт. URL: https://learn.javascript.ru/ajax-intro (дата звернення 18.05.2021);

16. Практичні приклади Ajax. Hostinger уроки: веб-сайт. URL: https://www.hostinger.com.ua/rukovodstva/chto-takoje-ajax/ (дата звернення 18.05.2021);

17. Майкл Моррисон. Изучаем JavaScript: учебник. Санкт-Петербург: Питер, 2012. 608 с. URL: https://library.kre.dp.ua/ (дата звернення 19.05.2021);

18. JavaScript. Метод append: веб-сайт. URL: http://code.mu/ru/javascript/manual/dom/append/ (дата звернення 20.05.2021);

19. Туристичні підприємства. Інтернет у діяльності туристичних підприємств: веб-сайт. URL: https://pidruchniki.com/1209061343731/turizm/internet_diyalnosti_turistichnih_pidpriyemstv (дата звернення 03.05.2021);

20. Наповнення сайту контентом. Контент сайту та його складові: веб-сайт. URL: https://webstudio2u.net/ua/studio-web/686-kontent-saita.html (дата звернення 12.05.2021).

Додаток А. Отримання даних по запиту від сервера та відправка даних, які ввів користувач для отримання турів за допомогою Ajax

```
loadData.addEventListener("click", sendJSON);
function sendJSON() {
   var vuvidPropoz = document.getElementById('vuvidPropoz');
   vuvidPropoz.style.display="block";
   var xhr = new XMLHttpRequest();
   xhr.open("POST", url, true);
   xhr.setRequestHeader('Content-Type', 'application/json');
   xhr.onreadystatechange = function () {
      if (xhr.readyState === 4 && xhr.status === 200) {
         var data = JSON.parse(xhr.responseText);
         console.log(data);
         var i = 0;
         for (key in data) {
            var borderToTour = document.createElement("div");
            var ContallPhotos = document.createElement("div");
            var allPhotos = document.createElement("li");
            var numOfDays = document.createElement("div");
            var nameOfHotel = document.createElement("div");
            var priceOfTour = document.createElement("div");
            var sDescribeOfTour = document.createElement("div");
            var starsOfHotel = document.createElement("div");
            var url = document.createElement("a");
            var buttonToDisc = document.createElement("div");
            let id;
            vuvidPropoz.appendChild(borderToTour);
            borderToTour.appendChild(ContallPhotos);
            ContallPhotos.appendChild(allPhotos);
```

```
borderToTour.appendChild(numOfDays);
borderToTour.appendChild(nameOfHotel);
borderToTour.appendChild(priceOfTour);
borderToTour.appendChild(sDescribeOfTour);
borderToTour.appendChild(starsOfHotel);
borderToTour.appendChild(url);
url.appendChild(buttonToDisc);
var str;
str = data[i].allPhotos;
var subs = str.split(";");
for(j=0;j<subs.length;j++){
   var imgAllPhotos = document.createElement("img");
   allPhotos.appendChild(imgAllPhotos);
   imgAllPhotos.setAttribute("src","photos/"+subs[j]);
   imgAllPhotos.style.width = "100px";
   imgAllPhotos.style.height = "100px";
}
numOfDays.innerHTML = "Кількість днів: "+ data[i].days;
nameOfHotel.innerHTML = "Готель: "+data[i].name;
priceOfTour.innerHTML = "Ціна: "+data[i].price;
sDescribeOfTour.innerHTML = data[i].sDescribe;
starsOfHotel.innerHTML = "Рейтинг: "+data[i].stars;
id = data[i].idTour;
url.setAttribute("href", "product.html");
url.setAttribute("target", "_blank");
url.style.fontSize = "20px";
buttonToDisc.innerText = "Детальніше";
buttonToDisc.addEventListener("click", function () {
   url = "http://www.travel.somee.com/api/tours/" + id;
   var xhr = new XMLHttpRequest();
```

```
                xhr.open("GET", url, true);
                xhr.onreadystatechange = function () {
                    if (this.readyState == 4 && this.status === 200) {
                        try {
                            var data = JSON.parse(xhr.responseText);
                            console.log(data);
                            var i = 0;
                            for (key in data) {
                                localStorage.setItem('tour', JSON.stringify(data[i]));
                                i++;
                            }
                        } catch (err) {
                            document.write(err.message + " in " + xhr.responseText);
                        } }
                    };
                    xhr.send(null);
                });
                i++;
            }
        } else if (xhr.readyState === 4) {
            vuvidPropoz.innerHTML = "Турів не знайдено";
            console.log(this);
        }};
    var data = JSON.stringify({
        "cityOut": cityOut.value,
        "cityIn": cityIn.value,
        "date": date.value
    });
    xhr.send(data);
}
```

Додаток Б. Фільтрація за допомогою Ajax

```
window.addEventListener("load", () => {
    let eventsList = document.getElementById("events-list");
    let winterButton = document.getElementById("winter");
    let springButton = document.getElementById("spring");
    let summerButton = document.getElementById("summer");
    let autumnButton = document.getElementById("autumn");


    xhr = new XMLHttpRequest();
    winterButton.addEventListener("click", () => {
        loadData("GET", "./eventsPeriod/winter.html");
    });
    springButton.addEventListener("click", () => {
        loadData("GET", "./eventsPeriod/spring.html");
    });
    summerButton.addEventListener("click", () => {
        loadData("GET", "./eventsPeriod/summer.html");
    });
    autumnButton.addEventListener("click", () => {
        loadData("GET", "./eventsPeriod/autumn.html");
    });


    function loadData(_query1, _query2) {
        xhr.open(_query1, _query2, true);
        xhr.onreadystatechange = function () {
            if (xhr.readyState == 4) {
                if (xhr.status == 200) {
                    eventsList.innerHTML = '';
```

```
                eventsList.innerHTML += xhr.responseText;
            }
        }
    }
    xhr.send();
  }
});
```

Додаток В. Частина коду розробки інтерактивної мапи світу

```
<metadata>
                <views>
                  <view h="647.825177808" padding="0" w="1000">
                    <proj flip="auto" id="mercator" lon0="65.3146660706" />
                    <bbox h="4064.12" w="6283.19" x="-3141.59" y="-2891.13" />
                  </view>
                </views>
</metadata>
              <g class="" id="countries">
                <path
d="M705.095473,347.358975L703.185541,347.101503L699.259904,349.734781L6
98.161758,348.598999L698.420314,345.662495L696.381534,343.613834L692.8091
90,346.703602L691.914707,348.301657L690.980523,347.628542L688.972844,348.
746781L688.125371,348.323054L686.033561,347.587345L684.488245,347.564174
L683.652811,347.455957L682.344409,346.520899L681.909851,347.764979L679.5
72241,348.438463L679.018582,351.205398L675.266223,352.766604L674.686909,3
54.312810L672.595672,354.766275L669.767179,353.474360L668.636641,357.6945
27L667.874589,360.130031L669.081805,360.621575L667.895801,362.437648L669
.021466,367.138651L671.122018,367.686762L671.349186,369.770679L668.833711
,372.682517L673.480983,374.311011L680.406385,373.822025L684.010834,372.49
4781L684.107864,369.760841L685.653897,367.928997L692.251661,365.986526L6
92.098448,364.018170L693.276139,362.026616L695.041314,361.187620L693.9511
94,358.982338L696.589497,359.086936L697.236317,356.586326L698.617671,355.
165008L697.644933,352.022243L699.264491,350.523488L706.834993,348.754680
L708.449535,348.364676L707.955785,347.364479L705.095473,347.358975Z "
                  data-id="AF" data-name="Afghanistan" id="AF">
                  <title>Afghanistan</title>
                </path>
```

Додаток Г. Серверна частина. Контролер туру

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Data.Common;
using System.Data.SqlClient;
using TravelAgency.Models;
namespace TravelAgency.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   public class ToursController : ControllerBase
   {
      // GET: api/<ToursController>
      [HttpPost]
      public IActionResult Post(InTourReq inTourReq)
      {
         List<OutTourReq> tours = new List<OutTourReq>();
         string sql = @$"SELECT Tours.id_tour, days, price, name, stars, sDescribe,
AllPhotos
                  FROM Tours,
                    (SELECT Tours.id_tour TourId, STRING_AGG(photo, ';') AllPhotos
                        FROM Tours, Photos
                        WHERE Tours.id_tour = Photos.id_tour
                          and id_city_out = (select distinct id_city from Cities where city
Name = N'{inTourReq.CityOut}')
                          and id_city_in = (select distinct id_city from Cities where cityN
ame = N'{inTourReq.CityIn}')
                          and date >= '{inTourReq.Date}'
```

```
                    GROUP BY Tours.id_tour) PrepTable
                WHERE TourId = Tours.id_tour";

        using (SqlConnection connection = new SqlConnection(DBConn.ConnStr) /*
DBConn.getConnection().Connection*/)
        {
            try
            {
                connection.Open();
                SqlCommand command = new SqlCommand(sql, connection);
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    // UVAGA !!!
                    if (!reader.HasRows)
                        return NotFound(new { errorText = "Жодного туру не знайдено"
});
                    while (reader.Read())
                    {
                        tours.Add(new OutTourReq()
                        {
                            IdTour = reader.GetInt32(0),
                            Days = reader.GetByte(1),
                            Price = reader.GetInt32(2),
                            Name = reader.GetString(3),
                            Stars = reader.GetByte(4),
                            SDescribe = reader.GetString(5),
                            AllPhotos = reader.GetString(6)
                        });
                    }
                }
```

```
            }
            catch (Exception e)
            {
                string str = e.Message;
                return (new ObjectResult(new { errorText = "Виникла помилка при ро
боті з базою даних" }) { StatusCode = 503 });
            }
        }
        return Ok(tours);
```