ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ І. І. МЕЧНИКОВА

(повне найменування закладу вищої освіти)

Факультет математики, фізики та інформаційних технологій

(повне найменування факультету)

Кафедра інформаційних технологій

(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «Бакалавр»

«Розробка інтерактивної туристичної карти міста Одеси з використанням веб-технологій»

(тема кваліфікаційної роботи українською мовою)

«Development of an interactive tourist map of the city of Odessa using web technologies»

(тема кваліфікаційної роботи англійською мовою)

Виконала: здобувачка заочної форми навчання

спеціальності 122 Комп'ютерні науки

(код, назва спеціальності)

Освітня програма Комп'ютерні науки

(назва)

Салманова Фаріда Ільдирим кизи

(прізвище, ім'я, по-батькові здобувача)

Керівник ст. викладач Вохменцева Т.Б. (науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент Корчемний П.А.

(науковий ступінь, вчене звання, прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
Інформаційних технологій .
№ 1 від 09 червня 2024 р.

Завідувачка кафедри
(підпис) КАЗАКОВА Надія .
(прізвище,ім'я)

Захищено на засіданні ЕК № 13 .
протокол № 6 від 19 червня 2024 р.

Оцінка задовільно / E / 60 .
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК
(підпис) КОПИЧЕНКО Іван .
(прізвище,ім'я)

Одеса 2024

ЗМІСТ

ТЕРМІНИ, СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

База даних – це структурована колекція даних, яка зберігається та організовується для ефективного доступу, оновлення та аналізу. Бази даних використовуються для зберігання інформації у веб-застосунках та інших програмах.

Бекенд – це частина веб-застосунка, яка відповідає за обробку запитів від клієнтів, доступ до бази даних та бізнес-логіку. Це зазвичай серверна частина застосунка.

Веб-застосунок – це програмне забезпечення, яке виконується на веб-сервері та доступне користувачам через веб-браузер. Веб-застосунки можуть виконувати широкий спектр функцій, ІС зі статичним контентом до складних застосунків з інтерактивними можливостями.

Веб-технології – це набір технологій, які використовуються для розробки та виконання веб-застосунків та ІС. Це може включати мови програмування, фреймворки, бібліотеки, протоколи, бази даних та інші інструменти.

Геоінформаційна система (ГІС) – це програмне забезпечення, яке дозволяє збирати, зберігати, аналізувати та відображати географічні дані. ГІС використовується для роботи з картами, відображення географічних об'єктів та аналізу просторових взаємозв'язків.

Інтерактивна карта – це графічне зображення географічної області (наприклад, міста, регіону або світу), яке дозволяє користувачам взаємодіяти з інформацією на карті, зазвичай шляхом навігації, зумування, перегляду додаткових даних тощо.

Інформаційна система – це система, яка забезпечує збір, збереження, обробку та поширення інформації для підтримки бізнес-процесів та прийняття рішень в організації.

Фронтенд – це частина веб-застосунка, яка відповідає за відображення та взаємодію з користувачем у веб-браузері. Це може включати HTML, CSS та JavaScript код.

API – це набір протоколів, інструментів та визначень, які дозволяють різним програмам взаємодіяти одна з одною. API використовується для передачі даних та інструкцій між програмами.

DOM – це стандартне програмне інтерфейсу, яке представляє структуру документа веб-сторінки та дозволяє програмам змінювати його структуру, стиль та вміст. DOM використовується веб-браузерами для маніпуляції веб-сторінками за допомогою JavaScript.

JSX – це розширення мови JavaScript, яке дозволяє використовувати синтаксис схожий на XML для опису структури веб-інтерфейсів. JSX використовується разом з бібліотеками, такими як React, для розробки інтерактивних веб-застосунків.

БД – база даних.

ГІС – геоінформаційна система

ІС – інформаційна система.

API – Application Programming Interface

DOM – Document Object Model

JSX – JavaScript XML

UML – unified modeling language.

ВСТУП

Одеса, місто з багатою історією, культурним розмаїттям та неповторним архітектурним шармом, впродовж багатьох років приваблює туристів своєю унікальною атмосферою та неповторним характером. Це місто, що ніколи не спить, постійно пульсує життям, розвивається та оновлюється, завдяки своїй неповторній ауро та особливій енергетиці. Будучи одним з найбільш привабливих туристичних напрямків як в Україні, так і у світі, Одеса щороку приймає численних гостей, які прагнуть нових вражень, культурного збагачення та незабутнього відпочинку.

У сучасному глобалізованому світі туризм набуває все більшої популярності, перетворюючись на важливу складову суспільного та економічного розвитку країн. Туристи стають більш прискіпливими та вимогливими до якості послуг, прагнучи відкривати нові місця, занурюватися у культурне та історичне надбання, смакувати місцеву кухню та відчувати справжню атмосферу міст, які вони відвідують. У цьому контексті створення зручних та інноваційних інструментів, які полегшать та розширять можливості туристів у вивченні міста, стає надзвичайно важливим.

Розробка інтерактивної туристичної карти міста Одеси з використанням сучасних веб-технологій відповідає цим потребам сучасного туриста. Цей проєкт не лише надасть туристам можливість ефективно планувати свої маршрути та ознайомлюватися з цікавими місцями міста, але й сприятиме залученню нових відвідувачів, підвищенню туристичного потенціалу Одеси, а також розвитку інновацій та використанню новітніх технологій у туристичній галузі.

Метою кваліфікаційної роботи бакалавра є створення інтерактивної туристичної карти міста Одеси з використанням веб-технологій. Ця карта стане зручним та ефективним інструментом, який дозволить туристам максимально комфортно та цікаво ознайомлюватися з містом. Завдяки відображенню на карті різноманітних туристичних об'єктів, ресторанів, музеїв, готелів та ін-

ших цікавих місць, процес планування подорожей стане простішим, а самі подорожі – більш насиченими та захопливими.

Актуальність даної роботи полягає у високій конкуренції в туристичній галузі та постійному зростанні вимог туристів до якості та доступності туристичних сервісів. Завдяки стрімкому розвитку Інтернету та смартфонів, туристи все частіше користуються веб-застосунками для планування своїх подорожей та отримання інформації про визначні місця, ресторани та готелі. У цьому контексті розробка інтерактивної туристичної карти міста Одеси відповідає сучасним вимогам та потребам сучасного туриста, допомагаючи підвищити якість та доступність туристичної інформації для відвідувачів міста.

Дана кваліфікаційна робота бакалавра складається з 52 сторінки, 11 рисунків, 2 таблиць та 10 джерел посилання.

1 РОЗГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІСНУЮЧИХ ТУРИСТИЧНИХ КАРТ

Одеса – це чудове місто з багатою історією, архітектурою та культурою. Як туристичний центр, воно має багато місць, які варто відвідати: архітектурні пам'ятки, музеї, культурні заклади, пляжі та парки. Розробка інтерактивної туристичної карти міста дозволить туристам легко знайомитися з головними пам'ятками, перевагами та можливостями Одеси.

Предметна область включає в себе основні туристичні місця Одеси, такі як:

– архітектурні пам'ятки: Одеський Оперний театр, Приморський бульвар, Потьомкінські сходи, Воронцовський палац, Палац Графа Воронцова, Катеринінська площа, Будинок Руссова, Пале-Рояль та інші;
– музеї: Одеський археологічний музей, Літературний музей, Художній музей, Музей західного і східного мистецтва, Одеський муніципальний музей особистих колекцій ім. О.В. Блещунова та ін.;
– культурні заклади: Одеський національний академічний театр опери та балету, філармонія, різноманітні концертні зали;
– визначні місця: Одеський морський торговельний порт, Ланжерон, Аркадія, Траса здоров'я, парки, площі, пам'ятники, церкви та інші;
– пляжі та рекреаційні зони: Аркадія, Лузанівка, Ланжерон, Отрада, Чкаловський пляж та інші.

Інтерактивна карта повинна містити інформацію про ці місця, їх розташування, години роботи, вартість вхідних квитків, опис та фотографії. Важливо також включити відомості про готелі, ресторани, транспорт, зони відпочинку та розваги, щоб надати туристам повну картину міста.

Веб-технології дозволять створити динамічний, зручний та інформативний ресурс. Інтерактивна карта може містити функції пошуку, навігації, створення маршрутів, додавання відгуків та фотографій користувачами.

Загалом, інтерактивна туристична карта Одеси, розроблена з використанням веб-технологій, стане корисним інструментом для ознайомлення з містом туристів, дозволить легше знаходити місця для відвідування та значно покращить туристичний досвід.

На даний момент існують різноманітні туристичні карти міста Одеси, які представлені у форматі паперових карт, електронних застосунків та інформаційних систем. Деякі з них надають базові картографічні дані та перелік визначних місць, інші – додаткові сервіси, такі як пошук за категоріями (ресторани, музеї, пам'ятники) або можливість побудови маршрутів.

Переваги існуючих рішень:

– доступність (більшість існуючих туристичних карт доступні онлайн та безкоштовно);
– базовий функціонал (багато карт надають базову інформацію про головні визначні місця та об'єкти міста);
– побудова маршрутів (деякі застосунки та інформаційні системи (ІС) дозволяють користувачам планувати свої маршрути та навіть розраховувати час подорожі).

Недоліки існуючих рішень:

– обмежений функціонал (більшість існуючих рішень мають обмежений функціонал та недостатню інтерактивність);
– нестабільність (деякі застосунки та можуть працювати нестабільно або містити помилки);
– недостатня актуальність (інформація на деяких картографічних рішеннях може бути застарілою або неповною).

Нове інтерактивне рішення повинно враховувати наступні вимоги:

– зручність використання (інтерфейс повинен бути інтуїтивно зрозумілим та зручним для користувача будь-якого рівня);
– актуальність інформації (карта має містити актуальну та повну інформацію про визначні місця, ресторани, магазини та інші об'єкти);

– інтерактивність (рішення повинно надавати можливість взаємодії з картою, таку як пошук, фільтрація та побудова маршрутів);
– мобільність (карта має бути адаптована для використання на мобільних пристроях та планшетах);
– надійність (система повинна працювати стабільно та без збоїв навіть при великому навантаженні);
– широкий функціонал (окрім основних функцій, веб-застосунок повинен надавати додаткові сервіси, такі як оглядові екскурсії, відгуки користувачів тощо).

1.1 Дослідження туристичного потенціалу міста Одеси

Одеса – місто з величезним туристичним потенціалом, завдяки своєму унікальному розташуванню, історії, архітектурі та культурній спадщині. Розглянемо детальніше чинники, які роблять Одесу привабливою для туристів: географічне розташування та клімат, історична та архітектурна спадщина, культурний туризм, пляжний відпочинок, місцева кухня та напої, розважальна інфраструктура та транспортна доступність.

Одеса розташована на узбережжі Чорного моря, що забезпечує м'який помірний клімат з теплим літом і м'якою зимою. Це робить її привабливою для відпочинку та оздоровлення протягом більшої частини року.

Одеса має багату історію, яка сягає корінням у 18 століття. Місто славиться своїми архітектурними пам'ятками, такими як Потьомкінські сходи, Оперний театр, Приморський бульвар, Воронцовський палац та інші. Історичний центр міста занесений до списку Всесвітньої спадщини ЮНЕСКО.

Одеса відома як культурний центр з багатою традицією в літературі, музиці, театрі та образотворчому мистецтві. Місто пропонує численні музеї, галереї, театри, концертні зали та фестивалі.

Завдяки теплому морю та чудовим піщаним пляжам, Одеса приваблює величезну кількість туристів, які шукають можливість для відпочинку на березі моря.

Одеська кухня має свої унікальні особливості, які є частиною культурної спадщини міста. Відвідувачі можуть скуштувати традиційні страви та місцеві вина та коньяки.

Одеса пропонує широкий вибір розваг та активного відпочинку: нічні клуби, парки розваг, кінотеатри, концертні майданчики, спортивні комплекси та багато іншого.

Одеса має міжнародний аеропорт, морський порт та зручне сполучення автомобільним і залізничним транспортом з іншими містами України та Європи.

Отже, Одеса володіє всіма необхідними факторами для успішного розвитку туризму. Місто пропонує безліч можливостей для культурного, пляжного, оздоровчого, гастрономічного та розважального туризму. Активний розвиток туристичної інфраструктури, промоція міста та залучення інвестицій можуть зробити Одесу одним з найпривабливіших туристичних центрів Чорного моря.

1.2 Формулювання завдання розробки інтерактивної туристичної карти

Мета проєкту: розробити інтерактивну туристичну карту міста Одеси з використанням веб-технологій, яка буде корисним та зручним інструментом для ознайомлення з основними туристичними пам'ятками, інфраструктурою та можливостями міста.

До ключових завдань можна віднести, по-перше створення інформаційної бази даних про туристичні об'єкти Одеси. Включити відомості про архітектурні пам'ятки, музеї, культурні заклади, визначні місця, пляжі та зони відпочинку, готелі, ресторани, транспорт та розваги. Зібрати детальну інфор-

мацію про кожний об'єкт: місцезнаходження, години роботи, ціни, опис, фотографії тощо. По-друге, розробка інтерактивної веб-карти міста. Інтегрувати базу даних з геоінформаційною системою та картографічним матеріалом, щоб створити інтерактивну карту із зображенням розташування туристичних об'єктів. По-третє, реалізація функціональності та інтерфейсу веб-карти. Передбачити можливість пошуку за різними критеріями, навігацію, створення маршрутів, відображення додаткової інформації про об'єкти, перегляд фотографій, додавання коментарів та оцінок користувачами. По-четверте, інтеграція додаткових сервісів. Передбачити функції бронювання готелів, покупки квитків на заходи, замовлення трансферу або екскурсій через веб-інтерфейс карти. По-п’яте, забезпечення мобільності та кросплатформеності. Розробити адаптивний дизайн та функціональність для використання веб-карти на різних пристроях, включаючи смартфони та планшети. По-шосте, просування та маркетинг веб-карти. Розробити стратегію маркетингу та промоції веб-ресурсу, щоб забезпечити високу відвідуваність серед туристів та жителів міста.

Нижче описані основні технічні вимоги [1]:

– веб-технології: HTML, CSS, JavaScript, React/Angular/Vue тощо;
– бази даних: MySQL, MongoDB або інші;
– геоінформаційні системи та картографічний матеріал: OpenStreetMap, Google Maps або інші;
– веб-сервер: Apache, Nginx або інший;
– хостинг та доменне ім'я для розміщення веб-ресурсу.

До ключових результатів, в свою чергу можна віднести:

– інформативна база даних про туристичні об'єкти Одеси;
– інтерактивна веб-карта міста з відображенням туристичних пам'яток;
– зручний та інформативний інтерфейс для користувачів;
– функціональність пошуку, навігації, створення маршрутів, додавання фото та коментарів;

– інтеграція додаткових сервісів (бронювання, купівля квитків, замовлення послуг);
– адаптивний дизайн для різних пристроїв;
– промоція та популяризація веб-ресурсу серед туристів.

1.3 Визначення вимог до розробки системи

Функціональні вимоги до розробки інтерактивної туристичної карти міста Одеси з використанням веб-технологій [2]:

– інтерактивна карта міста з можливістю переміщення, масштабування та вибору зручного для користувача режиму перегляду;
– відображення на карті основних туристичних об'єктів міста, таких як архітектурні пам'ятки, музеї, парки, пляжі, культурні заклади, готелі, ресторани, транспортні вузли тощо;
– подання детальної інформації про кожен туристичний об'єкт при натисканні на його значок на карті, включаючи опис, фотографії, години роботи, контакти, ціни (при наявності) та інші корисні відомості;
– можливість пошуку та фільтрації туристичних об'єктів за різними критеріями (тип об'єкту, розташування, ціновий діапазон, оцінки відвідувачів тощо);
– функція побудови маршрутів між обраними туристичними об'єктами з використанням різних видів транспорту (пішки, громадський транспорт, автомобіль) та отримання інформації про приблизний час та відстань маршруту;
– реєстрація користувачів та можливість додавання ними власних коментарів та фотографій до туристичних об'єктів, а також виставлення оцінок (наприклад, зірочок);
– інтеграція додаткових сервісів, таких як бронювання готелів, купівля квитків на заходи, замовлення трансферу або екскурсій через веб-інтерфейс карти;

– адаптивний дизайн та відповідна функціональність для використання веб-карти на різних пристроях, включаючи мобільні телефони та планшети.

Нефункціональні вимоги до розробки інтерактивної туристичної карти міста Одеси з використанням веб-технологій:

– доступність: ІС та карта повинні бути доступні цілодобово та мати високу швидкість завантаження;
– надійність: система повинна бути стабільною та швидко реагувати на дії користувача;
– безпека: захист особистих даних користувачів, використання протоколу HTTPS та інших заходів безпеки;
– масштабованість: можливість легкого додавання нових функцій та розширення системи в майбутньому;
– простота використання: інтуїтивно зрозумілий інтерфейс та легка навігація для людей із різним рівнем технічних навичок;
– кросбраузерність та кросплатформеність: коректна робота карти у різних веб-браузерах та операційних системах.

Технічні вимоги до розробки інтерактивної туристичної карти міста Одеси з використанням веб-технологій:

– використання сучасних веб-технологій, таких як HTML5, CSS3, JavaScript, React/Angular/Vue (вибір конкретних фреймворків залежить від преференцій та досвіду розробників);
– інтеграція з геоінформаційними системами та картографічним матеріалом, наприклад, OpenStreetMap, Google Maps або іншими подібними сервісами;
– використання реляційної або NoSQL бази даних (MySQL, MongoDB або подібні) для зберігання інформації про туристичні об'єкти та користувачів;
– розміщення ІС та карти на надійному веб-сервері (Apache, Nginx або інші) з належним налаштуванням безпеки та продуктивності;

- використання власного доменного імені та сертифіката SSL/TLS для забезпечення безпечного з'єднання з ІС;
- налагодження системи контролю версій (Git, SVN або інше) для спільної роботи над проєктом;
- документування коду та процесів розробки для полегшення подальшої підтримки та розвитку системи.

Загалом, вимоги до розробки інтерактивної туристичної карти міста Одеси спрямовані на створення зручного, інформативного та безпечного веб-ресурсу, який забезпечить високу якість обслуговування туристів та надасть їм всю необхідну інформацію для кращого ознайомлення з містом.

2 АНАЛІЗ ВИБОРУ АРХІТЕКТУРИ ТА ВЕБ-ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНТЕРАКТИВНОЇ ТУРИСТИЧНОЇ КАРТИ

Для реалізації інтерактивної туристичної карти міста Одеси необхідно ретельно проаналізувати вибір архітектури та веб-технологій, які будуть використовуватися в процесі розробки. Цей аналіз дозволить визначити найбільш ефективні рішення, які забезпечать створення якісного, масштабованого та гнучкого веб-застосунку. Розробка веб-застосунків, особливо таких, що містять картографічний функціонал, вимагає ретельного підходу до вибору архітектури та технологій. Правильний вибір забезпечить ефективну роботу системи, її масштабованість, гнучкість та відповідність сучасним стандартам веб-розробки. Архітектура застосунку визначає загальну структуру, взаємодію компонентів та потоки даних. Існує кілька поширених архітектурних підходів, таких як монолітна, мікросервісна та архітектура з мікрофронтендами. Кожен підхід має свої переваги та недоліки, і вибір залежить від масштабу проекту, вимог до продуктивності, гнучкості та досвіду команди розробників. Веб-технології включають вибір інструментів, бібліотек та фреймворків для розробки клієнтської та серверної частин застосунку, управління даними та картографічним функціоналом. На клієнтській стороні (фронтенд) доцільно використовувати сучасні JavaScript-фреймворки та бібліотеки, такі як React, Angular або Vue.js, які дозволяють створювати високопродуктивні інтерфейси з компонентною архітектурою, віртуальним DOM та реактивним програмуванням. Для відображення інтерактивних карт можна використовувати такі бібліотеки, як Leaflet, OpenLayers або Google Maps API.

2.1 Аналіз архітектурних рішень для веб-застосунків з картографічним функціоналом

Одним з ключових рішень є вибір архітектури застосунку, яка визначає загальну структуру, взаємодію компонентів та потоки даних. Нижче описано

три основні підходи: монолітна архітектура, мікросервісна архітектура та архітектура з мікрофронтендами.

В випадку “монолітної архітектури” весь веб-застосунок є єдиним цілим, де всі компоненти (серверна логіка, інтерфейс користувача, управління базами даних тощо) тісно інтегровані та розгортаються як єдине ціле. Переваги використання – простота в розробці та розгортанні, швидка комунікація між компонентами. А до недоліку можна віднести низьку масштабованість, складність внесення змін та можливих помилок, що можуть вплинути на весь застосунок [3].

Монолітна архітектура передбачає, що весь застосунок є єдиним цілим, де всі компоненти, такі як серверна логіка, інтерфейс користувача, управління базами даних і картографічний функціонал, тісно інтегровані та розгортаються як єдина одиниця. Цей підхід має перевагу простоти в розробці та швидкої комунікації між компонентами.

Проте, монолітна архітектура має певні обмеження щодо масштабованості та гнучкості. Оскільки всі функції об'єднані в одному застосунку, кожна зміна або оновлення вимагає перебудови всієї системи. Крім того, помилки в одному компоненті можуть вплинути на весь застосунок, що ускладнює виявлення та усунення проблем.

Для веб-застосунків з картографічним функціоналом монолітна архітектура підходить лише для невеликих проектів з обмеженими вимогами до масштабованості та гнучкості. Однак, у міру зростання застосунку і збільшення кількості користувачів, цей підхід може стати обмежуючим фактором (рис.1).

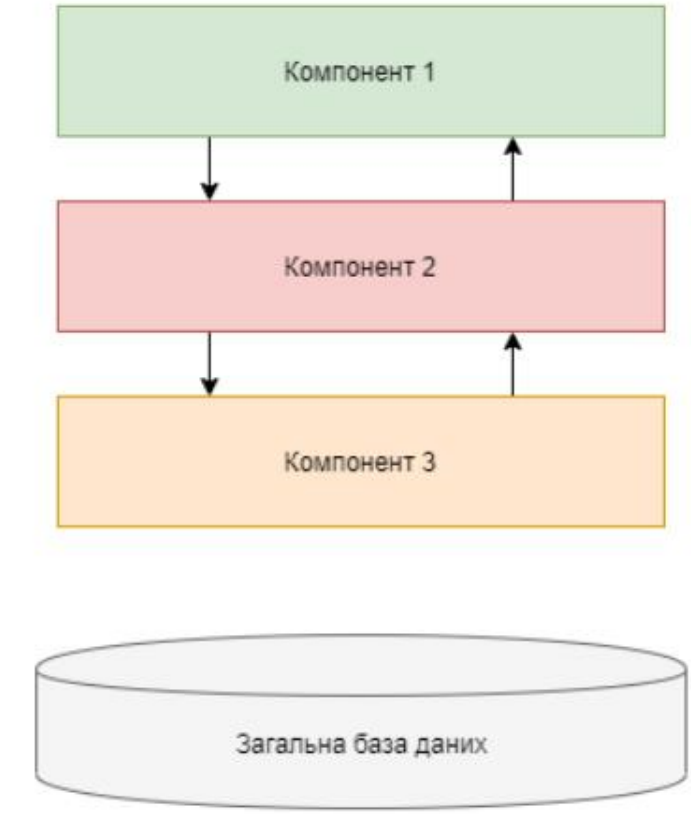

Рисунок 1 – Приклад “монолітної архітектури”

При підході “мікросервісна архітектура”, веб-застосунок розбивається на невеликі незалежні сервіси, кожен з яких має чітко визначену відповідальність та функціонал. Сервіси взаємодіють між собою за допомогою APIs та легкозв'язаних протоколів, таких як HTTP, AMQP або WebSocket. Перевагою є гнучкість, масштабованість, ізольованість помилок, незалежність розгортання компонентів. Недоліком – підвищена складність розробки, налагодження комунікації між сервісами, необхідність додаткових інструментів та інфраструктури [4].

Мікросервісна архітектура є альтернативним підходом, де веб-застосунок розбивається на окремі, незалежні сервіси. Кожен сервіс має чітко визначену відповідальність та функціонал і здатний масштабуватися та оновлюватися індивідуально. Сервіси взаємодіють між собою за допомогою доб-

ре визначених інтерфейсів, таких як RESTful API або протоколи передачі повідомлень.

У контексті веб-застосунків з картографічним функціоналом мікросервісна архітектура дозволяє розбити застосунок на окремі сервіси, такі як:

– картографічний сервіс (відповідає за відображення та керування картами, маркерами, шарами даних тощо);
– сервіс даних (управляє даними про географічні об'єкти, маршрути, користувачів та їх коментарі);
– сервіс аналітики (обробляє дані для аналітики та візуалізації на картах);
– сервіс авторизації та аутентифікації (керує процесами реєстрації, входу та авторизації користувачів);
– сервіс повідомлень (забезпечує обмін повідомленнями між користувачами та компонентами застосунку (наприклад, оновлення даних на карті в реальному часі)).

Кожен мікросервіс може бути розгорнутий окремо, мати власний життєвий цикл та масштабуватися незалежно від інших сервісів. Це забезпечує гнучкість, модульність і полегшує виявлення та усунення помилок, оскільки вони ізольовані в межах конкретного сервісу.

Проте, мікросервісна архітектура вимагає ретельного планування та налаштування комунікації між сервісами, а також додаткових інструментів та інфраструктури для розгортання та керування мікросервісами (див.рис.2).

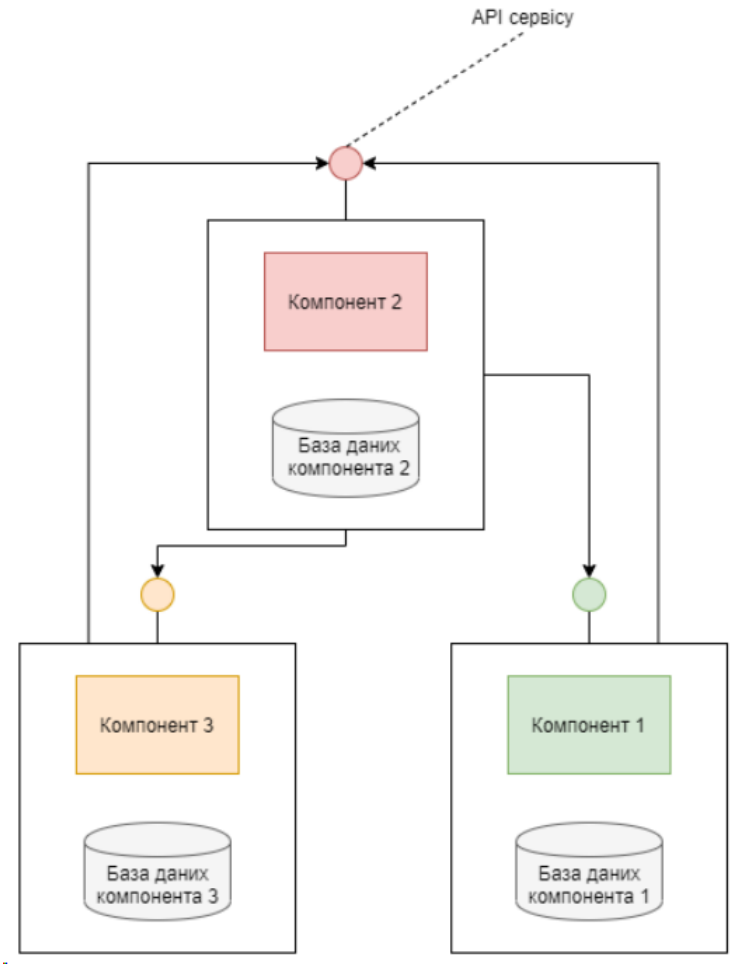

Рисунок 2 – Приклад мікросервісної архітектури

Підхід “архітектура з мікрофронтендами” поєднує переваги монолітної та мікросервісної архітектур. Зберігаючи централізовану веб-програму, він дозволяє розбити інтерфейс користувача (frontend) на незалежні мікрофронтенди. Кожен мікрофронтенд розробляється та розгортається окремо, а потім інтегрується в загальний інтерфейс за допомогою спеціальних технологій. Масштабованість, незалежність розвитку компонентів інтерфейсу, централізована серверна частина, – можна віднести до переваг. А, підвищена складність інтеграції мікрофронтендів, необхідність застосування спеціальних технологій та інструментів – до недоліків [5].

Архітектура з мікрофронтендами поєднує переваги монолітної та мікросервісної архітектур. Замість поділу на окремі сервіси для всіх частин за-

стосунку, цей підхід зосереджується на фронтенді (інтерфейсі користувача) та розбиває його на незалежні мікрофронтенди.

У контексті веб-застосунку з картографічним функціоналом архітектура з мікрофронтендами може виглядати наступним чином:

– ядро застосунку (загальна частина застосунку, що об'єднує інші компоненти та забезпечує базову функціональність);
– мікрофронтенд карти (компонент, який відповідає за відображення карти, маршрутів, маркерів та керування картографічними даними);
– мікрофронтенд профілю користувача (компонент для відображення та редагування інформації про користувача);
– мікрофронтенд коментарів (компонент для перегляду та додавання коментарів до географічних об'єктів);
– мікрофронтенд додаткових функцій (компонент, що включає інші функції застосунку, такі як бронювання, оплата, налаштування тощо).

Кожен мікрофронтенд розробляється та розгортається незалежно, а потім інтегрується в загальний інтерфейс за допомогою технологій, таких як модулі JavaScript, веб-компоненти або фреймворки мікрофронтендів (наприклад, Piral, Single-Spa або Module Federation).

Архітектура з мікрофронтендами забезпечує масштабованість, незалежність розробки компонентів інтерфейсу та можливість оновлювати та розгортати кожен компонент окремо. Проте, вона вимагає використання спеціальних технологій та інструментів для інтеграції мікрофронтендів, що може ускладнити розробку та потребувати додаткових зусиль.

Мікрофронтенд (рис.3) – це існування кількох вбудованих віджетів – різних джерел даних, сервісів і навіть продуктів різних провайдерів – у межах однієї вебсторінки. Таким чином ми можемо мати на одній сторінці програми з різними фронтенд-фреймворками, наприклад React.js і Angular. Звичайно, це не проблема, якщо використовувати це як перехідний стан, коли рухаємося від одного фреймворку до іншого, або ж якщо застосовуємо багато

зовнішніх сервісів, на чию поведінку не можемо вплинути. Однак в інших ситуаціях, на думку ThoughtWorks, така поведінка з мікрофронтендами спричиняє хаос і замість того, щоб допомагати в роботі, все заплутує та ускладнює [6].

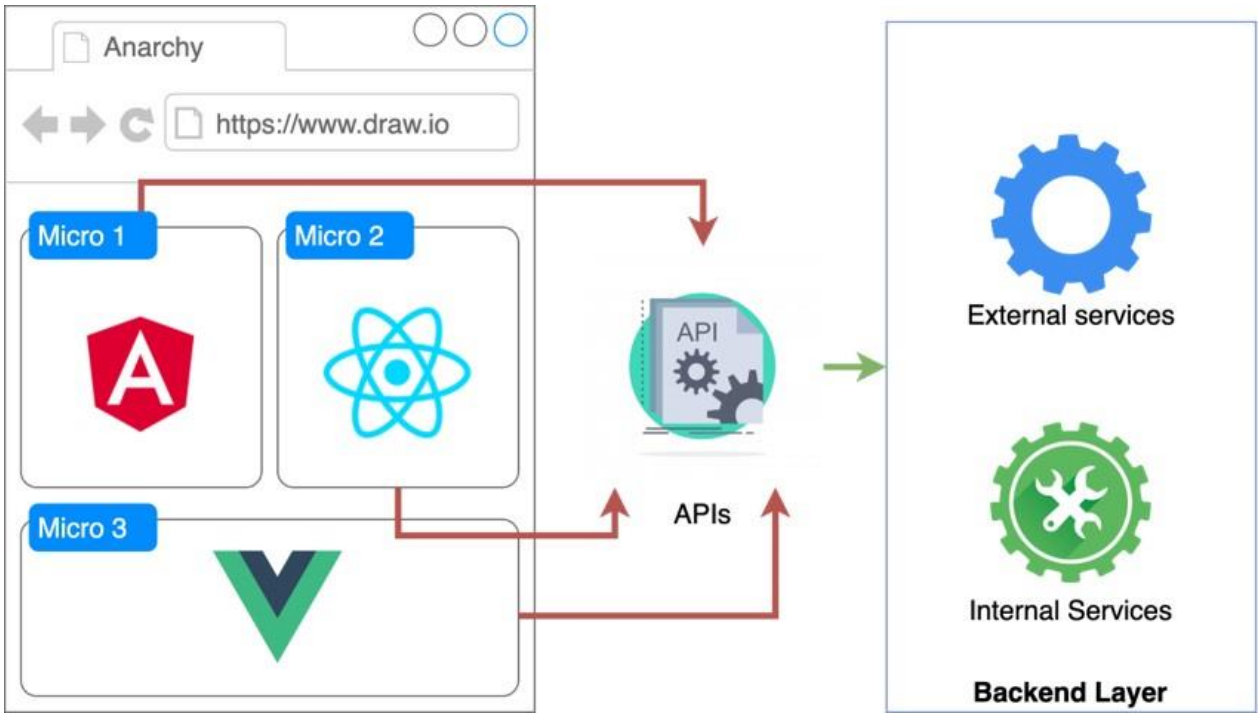

Рисунок 3 – Приклад архітектури з мікрофронтендами

Для реалізації інтерактивної туристичної карти найбільш доцільною видається використання “мікросервісна архітектура”. Така архітектура дозволить створити масштабовану систему, де окремі сервіси відповідатимуть за різні функціональні блоки: управління картою, робота з базою даних туристичних об'єктів, обробка коментарів та оцінок користувачів, інтеграція з додатковими сервісами тощо. Така модульність забезпечить гнучкість у розвитку та підтримці застосунку.

2.2 Вибір веб-технологій для реалізації інтерактивної туристичної карти

Вибір веб-технологій для реалізації інтерактивної туристичної карти залежить від обраної архітектури та функціональних вимог. Фронтенд (клієнтська частина) відповідає за представлення інформації користувачеві та забезпечення інтерактивності. Тут доцільно використовувати сучасні JavaScript-фреймворки та бібліотеки, такі як React, Angular або Vue.js. Вони забезпечують високу продуктивність, зручність розробки, реактивність та багатий функціонал для створення інтерфейсів.

React – один з найпопулярніших фреймворків, який пропонує гнучкий та декларативний підхід до розробки інтерфейсів. Він забезпечує високу продуктивність завдяки віртуальному DOM та підходу реактивного програмування [7]. Крім того, React має величезну екосистему готових компонентів та бібліотек, що значно прискорює розробку (рис.4).

Рисунок 4 – Фреймворк React

Фреймворк React – це відкрита бібліотека JavaScript для створення інтерфейсів користувача, розроблена та підтримувана компанією Meta (раніше Facebook). Вона набула величезної популярності завдяки своїй продуктивності, гнучкості та потужним можливостям. Ось основні характеристики та особливості React:

– компоненти (React базується на підході компонентної архітектури, де інтерфейс користувача розбивається на невеликі, повторно використовувані компоненти. Кожен компонент є незалежною та самодостатньою одиницею, що містить власний стан та логіку відображення);
– віртуальний DOM (React використовує віртуальне представлення DOM, що значно підвищує продуктивність. Замість безпосередньої маніпуляції реальним DOM, React створює легковагу копію DOM в пам'яті, порівнює її з попереднім станом та оновлює лише ті елементи, які змінилися. Це допомагає уникнути непотрібних операцій з DOM та підвищує швидкість роботи застосунку);
– JSX (React використовує синтаксис JSX, який є розширенням JavaScript, що дозволяє писати HTML-подібні структури безпосередньо в коді. JSX компілюється в JavaScript, що дає розробникам можливість створювати компоненти з комбінацією HTML та JavaScript);
– односпрямований потік даних (React застосовує односпрямований потік даних, за яким дані передаються від батьківських компонентів до дочірніх через властивості (props). Це забезпечує чіткий контроль над потоком даних та полегшує відстеження змін);
– менеджмент стану (React надає можливість керувати станом компонентів через розширення бібліотеки, такі як React Hooks (useState, useEffect тощо), які дозволяють додавати локальний стан компонентам. Для управління глобальним станом застосунку можна використовувати сторонні бібліотеки, такі як Redux, MobX або Context API);
– екосистема та спільнота (React має величезну та активну спільноту розробників, що сприяє створенню та розвитку численних бібліотек, інструментів та ресурсів для фреймворку. Це забезпечує широкі можливості для розширення функціональності та полегшує вирішення проблем);
– інструменти розробки (React пропонує потужний набір інструментів для розробки, таких як React Developer Tools для відлагодження ком-

понентів в браузері, Create React App для швидкого створення проектів та React Native для розробки мобільних застосунків);

– гнучкість та масштабованість (React є достатньо гнучким, щоб підтримувати як невеликі, так і великі проекти. Застосунки можна легко масштабувати за допомогою модульної структури та незалежних компонентів).

React став одним з найпопулярніших фреймворків для створення інтерфейсів користувача завдяки своїй простоті, продуктивності та гнучкості. Він широко використовується як у веб-розробці, так і в розробці мобільних застосунків та інших середовищ.

Angular – це повнофункціональний фреймворк для створення веб-застосунків, розроблений і підтримуваний командою Google [8]. Він базується на TypeScript, що надає переваги статичної типізації та полегшує розробку великих та складних застосунків (рис.5).

Рисунок 5 – Фреймворк React

Основні характеристики та можливості Angular:

– модульна структура (Angular використовує модульний підхід, де застосунок розбивається на модулі, кожен з яких містить певний функціонал та залежності);

– ін'єкція залежностей (фреймворк має вбудовану систему ін'єкції залежностей, яка полегшує управління залежностями та сприяє модульності);

– шаблони та компоненти (Angular має власну систему шаблонів та компонентів, які дозволяють розбити інтерфейс на логічні блоки та забезпечити їх повторне використання.
– двостороннє зв'язування даних (підхід двостороннього зв'язування даних забезпечує синхронізацію між моделлю даних та представленням);
– роутинг (Angular має вбудовану систему роутингу, яка дозволяє легко налаштовувати маршрути застосунку та керувати навігацією);
– відгук на події та потоки даних (завдяки використанню RxJS, Angular пропонує потужну систему реактивного програмування для роботи з асинхронними подіями та потоками даних);
– інструменти та CLI (Angular має власний набір інструментів для розробки, збирання, тестування та розгортання застосунків, включаючи Angular CLI для швидкого створення та управління проектами);
– велика спільнота та екосистема (Angular має велику спільноту розробників та багату екосистему бібліотек та компонентів, що полегшує створення та розширення застосунків);

Vue.js – це прогресивний фреймворк для створення інтерфейсів користувача. Він має простий та інтуїтивний синтаксис, а також забезпечує гнучкість та легкість інтеграції [9]. Ключові особливості Vue.js:

– прогресивний (Vue.js можна вводити поступово, починаючи з простих інтерактивних елементів інтерфейсу та розширюючи до повноцінної системи для створення складних застосунків);
– віртуальний DOM (фреймворк використовує віртуальне представлення DOM для оптимізації оновлень та підвищення продуктивності);
– реактивність (Vue.js забезпечує реактивну систему відстеження змін, яка автоматично оновлює інтерфейс при змінах в моделі даних);

– компоненти з можливістю повторного використання (Vue.js дозволяє створювати окремі компоненти інтерфейсу, які можна легко складати та повторно використовувати);
– директиви та фільтри (фреймворк пропонує набір вбудованих директив та фільтрів, які полегшують роботу з даними та DOM);
– легкий та гнучкий (Vue.js є легким та гнучким фреймворком, який легко інтегрується з іншими бібліотеками та фреймворками);
– невелика крива навчання (Простий синтаксис та інтуїтивна документація роблять Vue.js легким для вивчення);
– активна спільнота та екосистема (Vue.js має активну спільноту розробників та зростаючу екосистему бібліотек та інструментів для розширення можливостей фреймворку).

Vue.js – це фреймворк, який працює на JavaScript, створений для розробки користувацьких інтерфейсів. Він працює на базі звичайного HTML, CSS та JavaScript, з можливостями декларативно програмувати користувацькі інтерфейси будь-якої складності на основі компонентів (рис.6).

Рисунок 6 – Фреймворк Vue.js

Також слід розглянути бібліотеки для роботи з картами, такі як Leaflet, OpenLayers або Google Maps API [10]. Вони надають засоби для відображення інтерактивних карт, маршрутів, маркерів та додаткових шарів даних (табл.1).

Таблиця 1 – Порівняльна таблиця найбільш популярних бібліотек для роботи з картами

Критерій	Leaflet	OpenLayers	Google Maps API
Опис	Відкрита бібліотека JavaScript для відображення карт на веб-сторінках	Відкрита бібліотека JavaScript для візуалізації карт та геопросторових даних	Широка платформа Google для відображення карт, надання API та інструментів розробки
Ліцензія	Вільна (BSD)	Вільна (MIT)	Платна для комерційного використання
Розмір	Невелика (близько 100 Кб)	Середня (близько 500 Кб)	Великий розмір з численними бібліотеками та залежностями
Підтримувані картографічні шари	Стандартні (базові карти, тайли)	Широкий спектр (базові карти, тайли, векторні дані, маркери тощо)	Широкий спектр (базові карти, тайли, векторні дані, маркери, перспективні види тощо)
Можливості відображення	Базові (переміщення карти, масштабування)	Розширені (переміщення карти, масштабування, керування видимістю шарів, стильові засоби тощо)	Розширені (переміщення карти, масштабування, керування видимістю шарів, стильові засоби, інтерактивні елементи тощо)
Підтримка маршрутів	Базова (відображення маршрутів)	Розширена (відображення маршрутів, інструменти побудови маршрутів)	Розширена (відображення маршрутів, інструменти побудови маршрутів, інформація про маршрут)
Додаткові можливості	Невеликий набір додаткових плагінів	Широкий набір додаткових бібліотек та інструментів	Численні додаткові бібліотеки, інструменти та сервіси Google (Places API, Directions API, Geocoding API тощо)
Спільнота та підтримка	Активна спільнота, багато навчальних ресурсів	Активна спільнота, багато навчальних ресурсів	Величезна спільнота, багато навчальних ресурсів та документації
Рекомендації	Рекомендована для простих карт та базових функцій	Рекомендована для складних карт та розширених можливостей	Рекомендована для проектів, що вимагають численних додаткових сервісів, але готових платити за комерційне використання

Кожна з цих бібліотек має свої переваги та недоліки. Вибір залежатиме від конкретних вимог проекту, необхідних функцій, бюджету та рівня досві-

ду команди розробників. Leaflet є гарним вибором для більш простих карт, OpenLayers підходить для складних геопросторових візуалізацій, а Google Maps API пропонує найбільший набір інструментів та сервісів, але є платною для комерційного використання. Уважний аналіз переваг та недоліків допоможе зробити правильний вибір.

Для стилізації інтерфейсу можна використовувати препроцесори CSS, такі як Sass або Less, які полегшують роботу з каскадними таблицями стилів та надають додаткові можливості.

На бекенд (серверній стороні) потрібно реалізувати логіку для управління даними, обробки запитів та взаємодії з клієнтською частиною. Популярними технологіями для розробки бекенду є Node.js, Python (Django або Flask), Java (Spring Boot) або .NET Core. Node.js – це потужна платформа, яка дозволяє використовувати JavaScript і на серверній стороні. Вона забезпечує високу продуктивність завдяки асинхронній моделі подій та потужній екосистемі бібліотек та фреймворків, таких як Express.js. Node.js добре підходить для створення RESTful API, обслуговування WebSocket-з'єднань та інтеграції з різними базами даних. Python з фреймворками Django або Flask також є гарним вибором для розробки бекенду. Ці фреймворки пропонують швидку розробку, багатий функціонал та легку інтеграцію з численними бібліотеками та базами даних.

Для зберігання даних про туристичні об'єкти, користувачів та їх коментарі можна використовувати реляційні бази даних, такі як PostgreSQL або MySQL, або NoSQL-рішення, наприклад, MongoDB. Вибір залежить від структури даних та вимог до продуктивності (табл.2).

Таблиця 2 – Порівняльна таблиця реляційних (PostgreSQL, MySQL) та NoSQL (MongoDB) баз даних

Критерій	PostgreSQL	MySQL	MongoDB (NoSQL)
Тип	Реляційна база даних	Реляційна база даних	Документоорієнтована NoSQL база даних
Модель даних	Реляційна (таблиці, рядки, стовпці)	Реляційна (таблиці, рядки, стовпці)	Документо-орієнтована (колекції, документи у форматі JSON-подібних об'єктів)
Структура даних	Жорстка схема, визначена наперед	Жорстка схема, визначена наперед	Гнучка, без фіксованої схеми
Масштабованість	Горизонтальна шардинг, реплікація	Горизонтальна шардинг, реплікація	Горизонтальна шардинг, реплікація, автоматичний розподіл даних
Консистентність даних	Забезпечує ACID-властивості (атомарність, узгодженість, ізольованість, довговічність)	Забезпечує ACID-властивості	Без жорсткої консистентності, можлива невідповідність даних
Схема даних	Заздалегідь визначена і строга	Заздалегідь визначена і строга	Динамічна, без фіксованої схеми
Запити	SQL-запити, підтримка складних запитів та з'єднань	SQL-запити, підтримка складних запитів та з'єднань	Пошук за документами, простий синтаксис запитів
Продуктивність	Висока завдяки індексуванню та оптимізації запитів	Висока завдяки індексуванню та оптимізації запитів	Висока завдяки простоті моделі даних та відсутності з'єднань
Дані	Структуровані та нормалізовані	Структуровані та нормалізовані	Неструктуровані, можуть містити вкладені документи та масиви
Екосистема та підтримка	Величезна, багато інструментів та бібліотек	Величезна, багато інструментів та бібліотек	Велика і швидко зростаюча екосистема
Рекомендації	Для структурованих даних та складних операцій з великими обсягами	Для структурованих даних та складних операцій з великими обсягами	Для неструктурованих даних, масштабованості та гнучкості схеми

Вибір між реляційними та NoSQL базами даних (БД) залежить від характеру даних та вимог до продуктивності й масштабованості. Реляційні бази даних (PostgreSQL, MySQL) краще підходять для структурованих даних із

жорсткою схемою, коли потрібно виконувати складні запити та підтримувати високий рівень консистентності даних. NoSQL бази даних, як MongoDB, підходять для неструктурованих даних з мінливою схемою, коли потрібна гнучкість та горизонтальна масштабованість.

У випадку з інтерактивною туристичною картою можна розглянути використання MongoDB для зберігання даних про туристичні об'єкти, коментарі та оцінки користувачів, оскільки ці дані можуть бути неоднорідними та вимагати гнучкої схеми. Водночас, для зберігання інформації про користувачів та їх облікових даних краще використовувати реляційну базу даних. Остаточний вибір залежить від деталей проекту, структури даних, необхідної продуктивності та досвіду команди розробників.

Для забезпечення комунікації між мікросервісами та інтеграції з зовнішніми системами доцільно використовувати RESTful API. Це стандартизований підхід, який дозволяє здійснювати обмін даними між різними компонентами системи за допомогою HTTP-протоколу та визначеного набору методів (GET, POST, PUT, DELETE).

Для розгортання мікросервісів можна використовувати контейнерні рішення, такі як Docker та Kubernetes. Контейнери забезпечують ізоляцію та портативність застосунків, а також полегшують їх розгортання та масштабування.

Для безперервної інтеграції та безперервного розгортання (CI/CD) доцільно використовувати інструменти, такі як Jenkins, GitLab CI/CD або GitHub Actions, які автоматизують процеси збирання, тестування та розгортання програмного забезпечення.

Для моніторингу та управління застосунком можна використовувати інструменти, такі як Prometheus, Grafana або ELK-стек (Elasticsearch, Logstash, Kibana), які дозволяють збирати метрики, журнали та візуалізувати стан системи.

Для підвищення продуктивності та забезпечення масштабованості веб-застосунку варто розглянути використання технологій кешування, таких як

Redis або Memcached, які дозволяють тимчасово зберігати дані в оперативній пам'яті для швидкого доступу.

Для реалізації функцій бронювання готелів, купівлі квитків та замовлення послуг необхідно інтегрувати систему з відповідними зовнішніми API, такими як сервіси онлайн-бронювання, платіжні шлюзи та системи управління заходами.

Для забезпечення безпеки важливо використовувати протокол HTTPS, SSL/TLS-сертифікати, а також застосовувати сучасні методи автентифікації та авторизації, такі як JSON Web Tokens (JWT) або OAuth 2.0.

Для полегшення розробки та співпраці в команді можна використовувати інструменти для управління версіями коду, такі як Git, та системи для спільної роботи над проектом, наприклад, JIRA, Trello або GitHub Projects.

Важливим аспектом є тестування, тому варто приділити увагу автоматизованим тестам, таким як юніт-тести, інтеграційні тести та тести користувацького інтерфейсу. Для цього можна використовувати бібліотеки та інструменти, такі як Jest, Cypress або Selenium.

3 ПРОЄКТУВАННЯ ІНТЕРАКТИВНОЇ ТУРИСТИЧНОЇ КАРТИ МІСТА ОДЕСИ

Інтерактивна туристична карта Одеси повинна мати наступні основні функції: навігація та орієнтування на місцевості, відображення туристичних об'єктів та пам'яток, побудова маршрутів, додаткова інформація та послуги, інтерактивні функції, інтерактивні функції, мультиплатформенність та багатомовність.

Перша функція “навігація та орієнтування на місцевості”:

– зручна та зрозуміла навігація по місту;
– показ основних вулиць, площ, парків та інших об'єктів інфраструктури;
– визначення поточного місцезнаходження користувача.

Друга функція “відображення туристичних об'єктів та пам'яток”:

– нанесення на карту основних туристичних пам'яток (архітектурних, історичних, культурних, релігійних тощо);
– надання стислої інформації про кожен об'єкт (назва, опис, фото);
– можливість фільтрації об'єктів за категоріями (наприклад, архітектура, музеї, парки тощо).

Третя функція “побудова маршрутів”:

– можливість створення власних туристичних маршрутів з обраних точок інтересу;
– розрахунок оптимальних шляхів між точками маршруту;
– відображення орієнтовного часу проходження маршруту.

Четверта функція “додаткова інформація та послуги”:

– відображення місць харчування (ресторани, кафе, бари);
– відображення готелів та інших місць розміщення;
– інформація про громадський транспорт (маршрути, розклад);
– відображення цікавих подій та фестивалів у місті.

П’ята функція “Інтерактивні функції:

– можливість залишати відгуки та рекомендації про туристичні об'єкти;
– обмін маршрутами між користувачами;
– інтеграція з соціальними мережами для публікації фотографій та вражень.

Шоста функція “мультиплатформенність”:

– адаптивний дизайн для використання на різних пристроях (смартфони, планшети, ПК);
– можливість офлайн-використання (завантаження карт і даних для роботи без інтернету).

Сьома функція “багатомовність”:

– підтримка декількох мовних інтерфейсів для зручності іноземних туристів.

Реалізація цих функцій допоможе створити зручний та багатофункціональний інструмент для планування туристичних маршрутів, ознайомлення з визначними пам'ятками та об'єктами інфраструктури міста Одеси.

3.1 Розробка архітектури та моделювання бази даних

Наступним важливим етапом проєктування інтерактивної туристичної карти Одеси є розробка архітектури та моделювання бази даних. Ця частина проєкту має забезпечити ефективну та масштабовану структуру для зберігання, обробки та управління даними.

Для реалізації всіх необхідних функцій інтерактивної карти доцільно використовувати клієнт-серверну архітектуру. Вона складається з нище описаних компонентів.

По-перше це клієнтська частина (фронтенд):

– розробка користувацького інтерфейсу (UI) за принципами зручності та інтуїтивності;

– реалізація функціоналу з відображення карт, туристичних об'єктів, побудови маршрутів та інших інтерактивних можливостей;
– забезпечення адаптивності та мультиплатформенності (веб-браузери, мобільні застосунки).

По-друге – серверна частина (бек-енд):

– розробка серверної логіки для обробки запитів від клієнтів;
– реалізація функціоналу з пошуку та фільтрації туристичних об'єктів, розрахунку маршрутів, зберігання та обробки відгуків користувачів;
– забезпечення безпеки та масштабованості системи.

По-третє, розробка БД:

– проєктування ефективної структури бази даних для зберігання інформації про туристичні пам'ятки, об'єкти інфраструктури, маршрути, відгуки та інші дані;
– використання реляційної або нереляційної (NoSQL) бази даних в залежності від вимог до масштабованості та швидкості обробки даних.

Четверте, робота з геоінформаційною системою (ГІС):

– інтеграція з географічними інформаційними системами для отримання даних про карти, вулиці, дороги та інші елементи географічного ландшафту;
– використання сервісів, таких як OpenStreetMap або Google Maps, для надання картографічних даних.

Для ефективного зберігання та управління даними в рамках проєкту інтерактивної туристичної карти Одеси необхідно спроєктувати логічну модель бази даних. Вона повинна включати наступні основні сутності та зв'язки між ними.

Туристичні об'єкти (пам'ятки, музеї, парки, церкви, пляжі тощо):

– унікальний ідентифікатор об'єкта;
– назва;
– опис;
– категорія (архітектура, історія, культура, релігія тощо);

– географічні координати (широта, довгота);
– фотографії;
– час роботи (для музеїв, парків та інших об'єктів з обмеженим доступом);
– ціни на вхід (якщо потрібно).

Об'єкти інфраструктури (готелі, ресторани, кафе, бари тощо):

– унікальний ідентифікатор об'єкта;
– назва;
– категорія (готель, ресторан, кафе, бар тощо);
– опис;
– географічні координати (широта, довгота);
– фотографії;
– години роботи;
– середній чек (для закладів харчування);
– ціни на розміщення (для готелів).

Маршрути:

– унікальний ідентифікатор маршруту;
– назва;
– опис;
– список туристичних об'єктів, що входять до маршруту;
– орієнтовний час проходження;
– відстань.

Відгуки користувачів:

– унікальний ідентифікатор відгуку;
– текст відгуку;
– оцінка (рейтинг туристичного об'єкта);
– дата та час публікації;
– посилання на користувача, що залишив відгук;
– посилання на туристичний об'єкт, про який залишено відгук.

Користувачі:

– унікальний ідентифікатор користувача;

– ім'я користувача;

– контактна інформація (електронна пошта, номер телефону);

– налаштування (мова інтерфейсу, одиниці вимірювання тощо).

Ці сутності та зв'язки між ними дозволять зберігати та управляти всією необхідною інформацією для реалізації функціоналу інтерактивної туристичної карти Одеси. Для забезпечення цілісності та ефективності даних можуть бути використані відповідні індекси та обмеження в базі даних.

Крім того, важливо передбачити можливість масштабування бази даних для зберігання зростаючої кількості даних та збільшення навантаження на систему в міру її популяризації. Це може включати використання сучасних технологій баз даних, таких як горизонтальне масштабування, кешування та оптимізацію запитів.

На рис. 7 зображено структуру БД для реалізації даного проєкту.

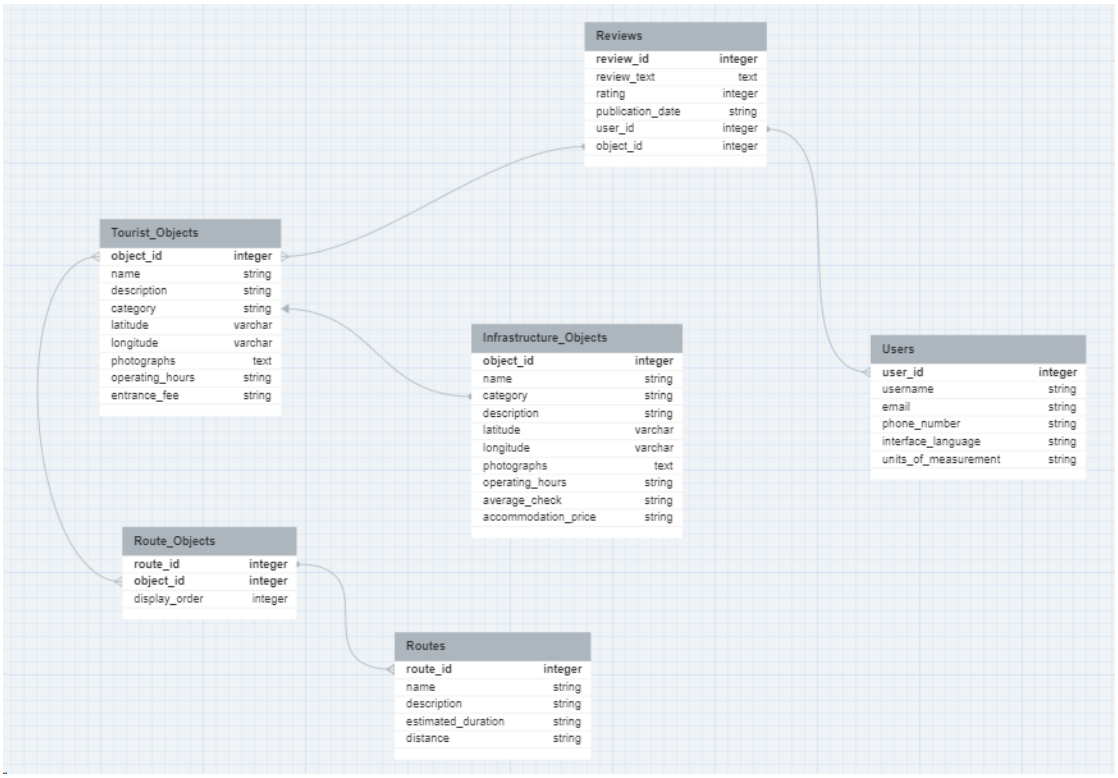

Рисунок 7 – Структура БД

На структурній схемі БД, представлено можливі зв'язки між таблицями:

– Tourist_Objects і Infrastructure_Objects (туристичні об'єкти та інфраструктурні об'єкти можуть мати спільний тип категорії (наприклад, "Музей", "Парк", "Ресторан" тощо));
– Tourist_Objects і Route_Objects (зв'язок "багато до багатьох": один туристичний об'єкт може бути включений у декілька маршрутів, і кожен маршрут може містити декілька туристичних об'єктів. Цей зв'язок визначається через таблицю Route_Objects);
– Tourist_Objects і Reviews (зв'язок "один до багатьох": один туристичний об'єкт може мати декілька відгуків, але кожен відгук належить лише одному туристичному об'єкту. Цей зв'язок визначається через зовнішній ключ object_id в таблиці Reviews);
– Users і Reviews (зв'язок "один до багатьох": один користувач може залишати декілька відгуків, але кожен відгук належить лише одному користувачеві. Цей зв'язок визначається через зовнішній ключ user_id в таблиці Reviews);
– Routes і Route_Objects (зв'язок "один до багатьох": кожен маршрут може містити декілька об'єктів маршруту, але кожен об'єкт маршруту може належати лише одному маршруту. Цей зв'язок визначається через зовнішній ключ route_id в таблиці Route_Objects).

Ці зв'язки дозволяють організувати зв'язану БД для зберігання та опрацювання інформації про туристичні об'єкти, маршрути, відгуки користувачів та інших даних, пов'язаних з туризмом та подорожами.

3.2 Проєктування Backend частини веб-застосунка

Проєктування бекенду веб-застосунка включає в себе ряд ключових етапів і функціональних складових, які необхідно врахувати під час розробки. Ці етапи включають наступне: маршрутизація та обробка запитів, робота з

базою даних, автентифікація та авторизація, бізнес-логіка, обробка відповідей та помилок, тестування та налагодження, документація, масштабованість та безпека.

Одним із перших кроків у розробці бекенду є визначення маршрутів для обробки різних типів запитів. Це може включати маршрути для отримання, оновлення, видалення та створення ресурсів.

Реалізація логіки доступу до бази даних включає створення, читання, оновлення та видалення записів у базі даних. Це також може включати виконання складних запитів для отримання необхідних даних..

Розробка механізмів аутентифікації користувачів, перевірка їхніх прав доступу та забезпечення безпеки даних є важливим аспектом розробки бекенду. Це включає в себе реалізацію механізмів ідентифікації користувачів, створення та перевірку токенів доступу, контроль доступу до ресурсів тощо.

Розробка бізнес-логіки застосунка включає в себе реалізацію логіки обробки замовлень, операцій з продуктами та інших бізнес-процесів, які потрібні для функціонування веб-застосунка.

Розробка механізмів формування відповідей на запити включає в себе роботу з форматами даних (наприклад, JSON або XML) та обробку помилок, які можуть виникнути під час виконання запитів.

Проведення тестування функціональності бекенду для забезпечення його правильної роботи, а також виправлення помилок та налагодження є важливим етапом розробки.

Написання документації, яка описує роботу різних компонентів бекенду, включаючи опис API, схеми бази даних та інші важливі аспекти, є необхідним кроком для забезпечення розуміння роботи бекенду іншими розробниками або користувачами.

Розробка бекенду з урахуванням можливостей масштабування та забезпечення оптимальної продуктивності при великих обсягах даних та трафіку. Також важливо забезпечити безпеку даних та захист від різних видів атак.

3.3 Проєктування інтерактивної картографічної частини

Проєктування інтерактивної картографічної частини веб-застосунка – це важливий етап розробки, що передбачає створення інтерфейсу та функціоналу для відображення географічної інформації на мапі та взаємодії з нею.

Першим кроком є вибір засобів (картографічної бібліотеки або API), які будуть використовуватися для реалізації інтерактивної картографічної частини. Це може бути бібліотека, така як Leaflet, OpenLayers або Mapbox, або API, надане сервісом картографії, наприклад, Google Maps або Mapbox.

Другий крок, це інтеграція з базою даних геоданих. Для відображення різних об'єктів на мапі (наприклад, туристичних об'єктів, маршрутів тощо) потрібно мати доступ до відповідних геоданих. Це може включати імпорт геоданих з бази даних, таких як PostGIS або MySQL з розширенням для геоданих, або використання зовнішніх сервісів, що надають геодані.

Третій крок – реалізація відображення об'єктів на мапі. Після отримання геоданих потрібно реалізувати їх відображення на мапі. Це включає створення маркерів, ліній, полігонів тощо для різних типів об'єктів, а також встановлення взаємозв'язків між ними (наприклад, з'єднання точок маршруту).

Четвертий етап включає в себе розробку інтерактивності на мапі. Інтерактивна картографічна частина повинна надати можливість користувачам взаємодіяти з мапою. Це може включати зум і переміщення по мапі, показ додаткової інформації при натисканні на об'єкти, можливість фільтрації об'єктів за категоріями тощо.

Оптимізація продуктивності, відбувається на п’ятому кроці. При великій кількості об'єктів на мапі важливо забезпечити оптимальну продуктивність відображення. Це може включати кластеризацію маркерів, пакетну передачу даних на клієнтську сторону, використання тайлованого підходу до відображення мапи тощо.

Шостий крок, адаптація до різних пристроїв. Важливо забезпечити адаптивність картографічної частини до різних типів пристроїв (комп'ютери,

планшети, мобільні телефони). Це включає в себе відповідність розмірів та розташування елементів на мапі для зручного використання на різних пристроях.

Сьомий крок – тестування та налагодження. Після реалізації картографічної частини важливо провести тестування її функціональності та продуктивності, а також виправлення можливих помилок.

Крайні, восьмий крок це документація та навчання користувачів. Написання документації, яка описує роботу картографічної частини веб-застосунка, можливості взаємодії з нею та інші важливі аспекти, а також навчання користувачів її використанню.

Вище наведено та описано лише загальний огляд етапів та завдань, які потрібно врахувати під час проєктування інтерактивної картографічної частини веб-застосунка.

4 РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОЇ ТУРИСТИЧНОЇ КАРТИ МІСТА ОДЕСИ

Розробка проєкту "Туристична карта міста Одеси" може бути розбита на кілька етапів, включаючи налаштування середовища розробки, створення структури проєкту, написання HTML, CSS та JavaScript коду, та оновлення стилів та функціональності.

Етапи розробки: налаштування середовища розробки, створення структури проєкту, написання HTML коду, написання CSS коду, написання JavaScript коду та оновлення стилів та функціональності.

Етап “налаштування середовища розробки” передбачає встановлення редактора коду (наприклад: PhpStorm, Visual Studio Code). Ініціалізація проєкту та установка необхідних залежностей (наприклад, Leaflet для картографічної бібліотеки). Етап “створення структури проєкту” включає в себе створення папок для різних типів файлів, таких як HTML, CSS та JavaScript. Розміщення зображень та інших ресурсів у відповідних папках. На етапі “написання HTML коду”, слід приділити увагу створення базової структури HTML файлу з визначенням основних елементів сторінки, таких як шапка/header, основний контент/main, підвал/footer. А також, додавання необхідних тегів для зображень, форм та інших елементів. Етап “написання CSS коду”, включає в себе стилізацію основних елементів сторінки, таких як заголовки, текстові блоки, форми, кнопки. Додавання стилів для карт та інших графічних елементів. Етап “написання JavaScript коду”, це етап більше описує налаштування та ініціалізації картографічної бібліотеки Leaflet. І додавання маркерів на карту з відповідними даними. А також, функції для інтерактивної взаємодії користувача з даною ІС, в тому числі реалізація функціоналу для зміни мови та іншого інтерактивного контенту. Крайній етап “оновлення стилів та функціональності”, включає в себе додавання стилів для кнопок зміни мови з відображенням активної мови. Оновлення JavaScript коду для взаємодії з

кнопками зміни мови та іншим інтерактивним контентом. На рис.8 представлено структуру проєкту в середовищі розробки.

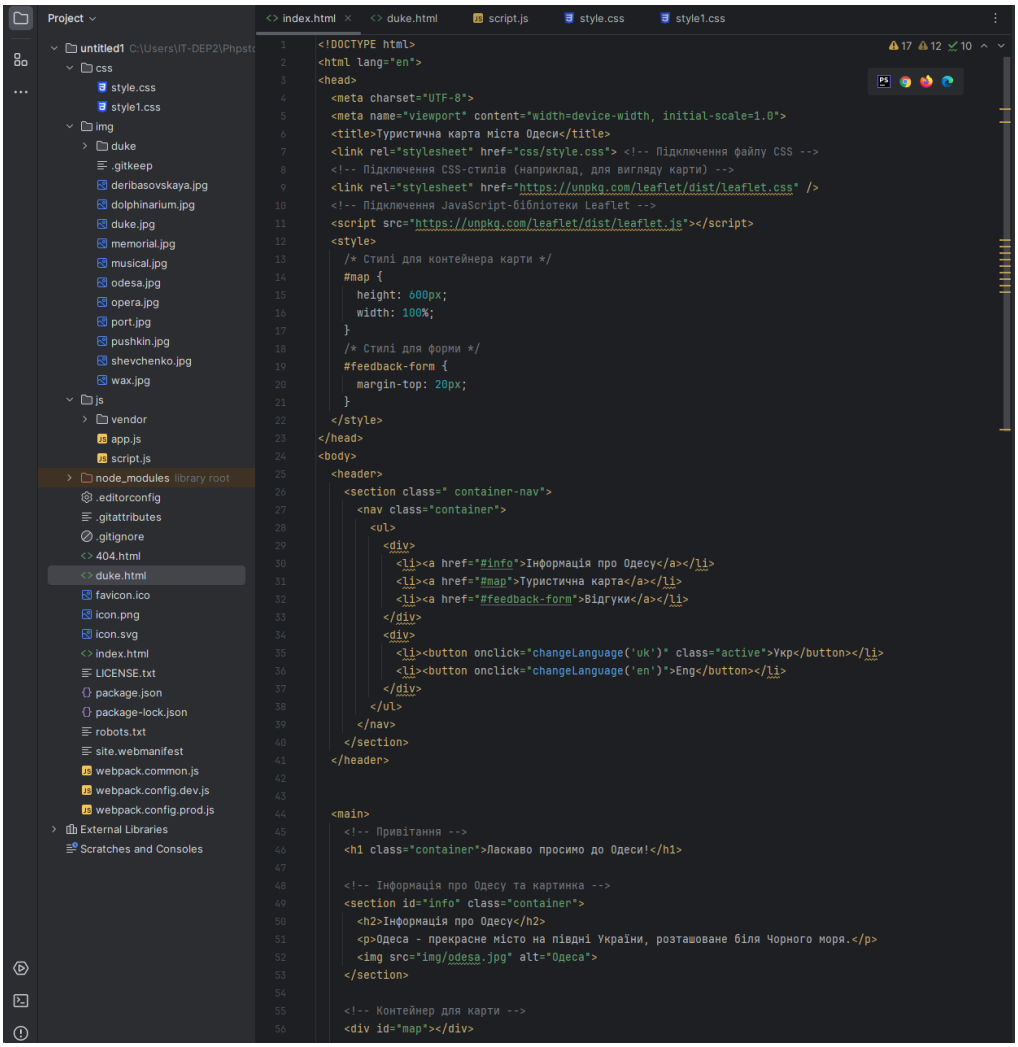

Рисунок 8 – Структура на початкових етапах реалізації проєкту в IDE PhpStorm

Основна сторінка (index.html) та внутрішня сторінка (наприклад duke.html) складаються з різних елементів, які відображають відповідну інформацію про місто Одеса та окремий об'єкт чи місце у місті.

Основна сторінка (index.html) складається з основних блоків, які в собі включать безпосередньо необхідну інформацію. Таким чином шапка або header сторінки включає в себе навігаційне меню з посиланнями на різні розділи сторінки (інформація про Одесу, туристична карта, відгуки). А також,

кнопки зміни мови (українська та англійська). Основна частина ІС (main content) складається з наступних блоків, а саме “Привітання з заголовком "Ласкаво просимо до Одеси!" ”, “Інформація про Одесу та зображення міста”, “Туристична карта міста Одеси з маркерами на популярних місцях” та “Форма для відгуків про місця” (рис.9)

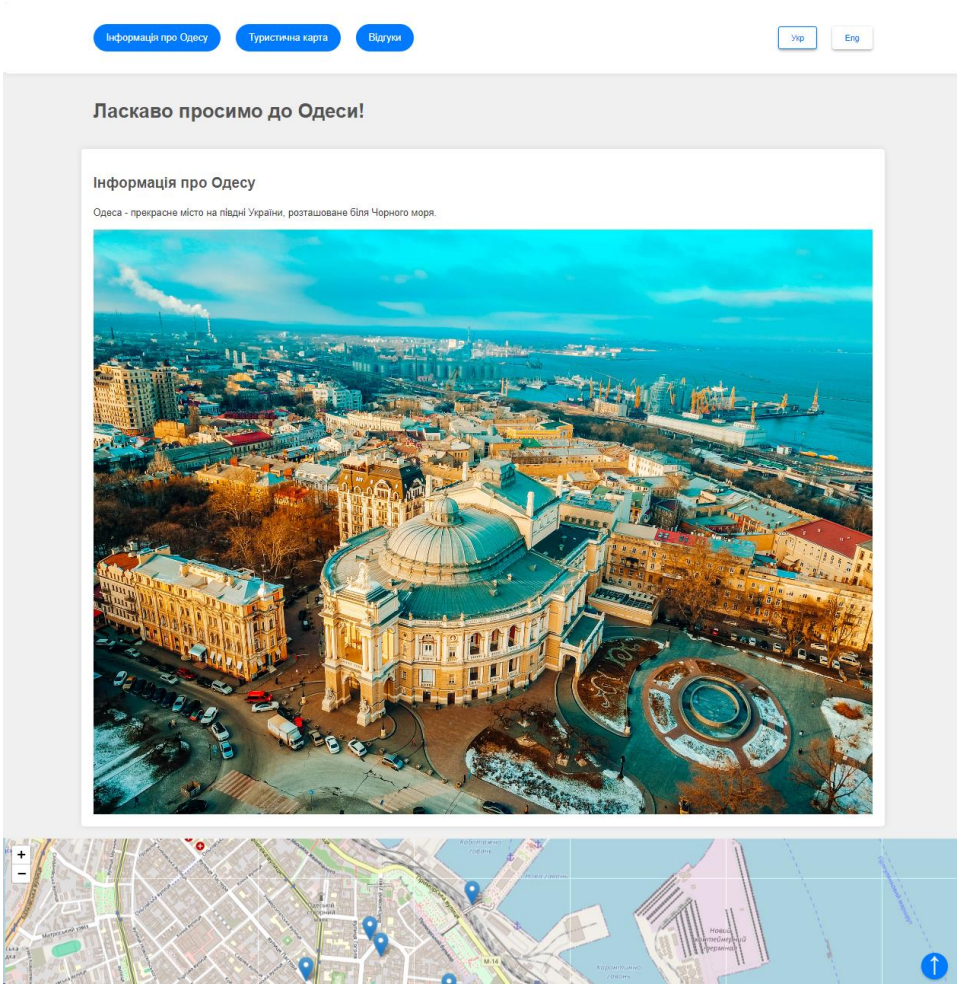

Рисунок 9 – Представлення реалізованого проєкту
(частина 1 основної сторінки)

На даному етапі, слід акцентувати увагу на блок “Туристична карта міста Одеси з маркерами на популярних місцях”. Даний блок є ключовим елементом веб-сторінки ІС що розробляється, який надає користувачам можливість візуально ознайомитися з розташуванням та інформацією про популяр-

ні місця міста Одеса на карті. Даний блок, містить карту міста Одеси з маркерами на популярних туристичних об'єктах. Може відображати основні визначні місця, важливі для туристів. При натиску на маркер, користувач може перегляну основну інформацію, яка може бути йому корисною при виборі місця відвідання. До основної інформації, наразі відноситься: назва пам’ятки (місця відвідування), фото пам’ятки (місця відвідування), адреса/місце розташування пам’ятки (місця відвідування), вартість входу, посилання (у вигляді ключового слова “Детальніше”) для переходу на внутрішню сторінку пам’ятки (місця відвідування) та рейтинг.

Блок “Форма відгуків” є важливою частиною веб-сторінки цієї ІС, яка надає користувачам можливість висловити свою думку, поділитися враженнями та залишити відгук про відвідані місця або послуги. Цей блок створений з метою взаємодії з аудиторією та збору корисної інформації, що допомагає у вдосконаленні сервісу або покращенні якості послуг.

Форма відгуків може містити різні поля для заповнення, такі як ім'я, електронна пошта, заголовок відгуку, текстове поле для коментаря, вибір рейтингу тощо. Ці елементи дозволяють користувачам зручно та швидко залишити свої враження. Далі будуть описані основні компоненти форми відгуків. Поля “Ім'я та електронна пошта” для введення особистої інформації користувача. Ці дані можуть бути використані для ідентифікації автора відгуку та зв'язку з ним у разі необхідності. Поле “заголовок відгуку” для короткого заголовку або опису відгуку, яке швидко передає основну ідею або враження. “Текстове поле для коментаря” необхідне для користувача щоб висловити свої думки, враження, досвід або зауваження щодо відвіданого місця або послуги. Поле для вибору рейтингу або оцінки за певними критеріями, такими як якість послуг, зручність розташування, відповідність очікуванням тощо. Кнопка “Надіслати” необхідна для відправлення заповненої форми. Після натискання на цю кнопку введені дані будуть надіслані адміністратору або збережені у базі даних.

Форма відгуків є важливим інструментом для спілкування з аудиторією та збору корисної інформації. Вона дозволяє створювати взаємодію з користувачами, враховувати їхні побажання та покращувати якість сервісу або послуг (рис.10).

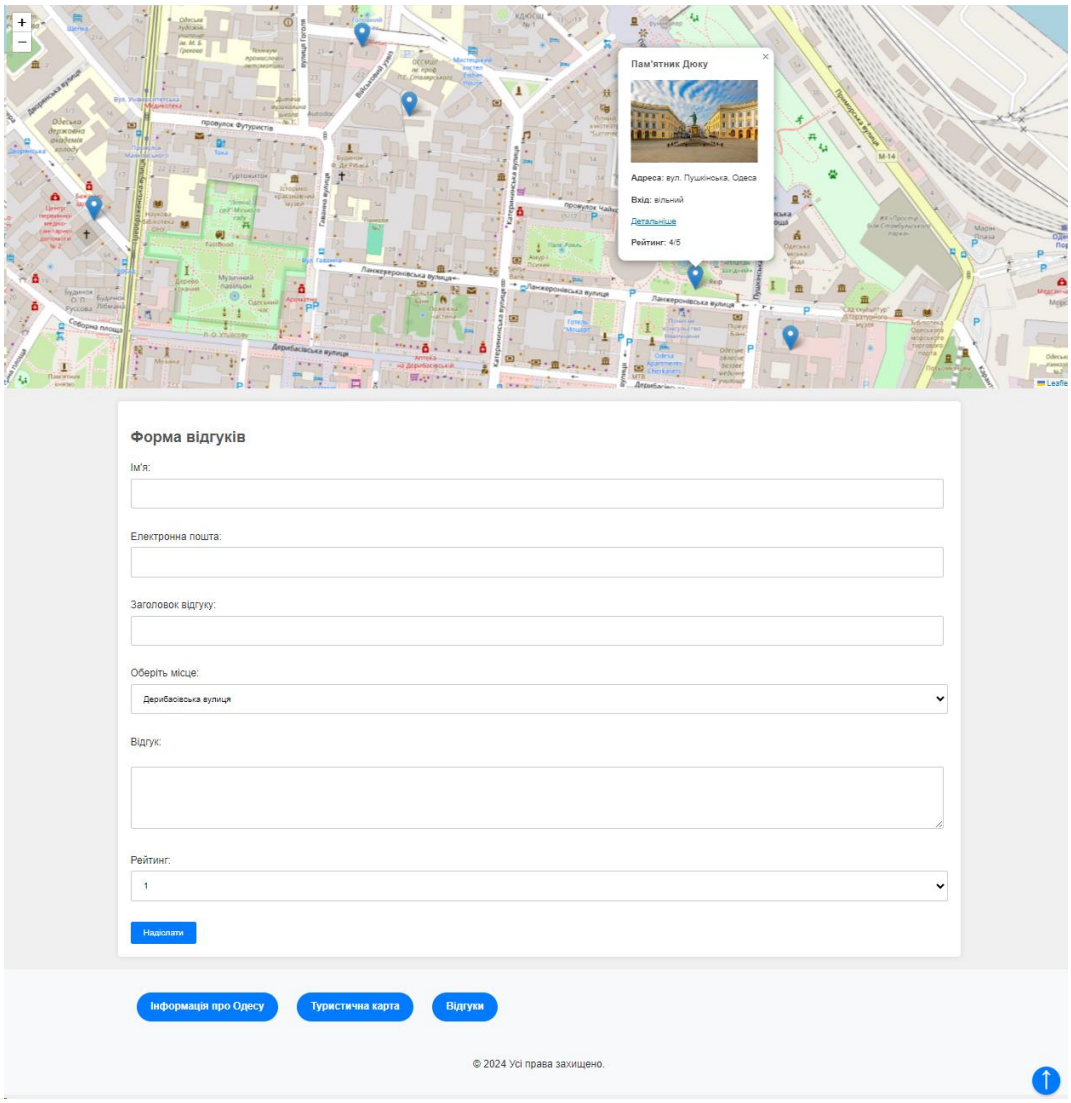

Рисунок 10 – Представлення реалізованого проєкту
(частина 2 основної сторінки)

Підвал або ж footer, складається з “Навігаційне меню з посиланнями на різні розділи сторінки (інформація про Одесу, туристична карта, відгуки)”, “Кнопка для прокрутки вгору сторінки” та “Інформація про авторські права”. У разі необхідності інформацію можна додати, наприклад розширити поля

для заповнення у блоці “форма відгуку” або ж у блоці “підвал/footer”, можна також додати інформацію щодо посилань на соціальні мережі.

Внутрішня сторінка (duke.html) включає в собі ті ж самі основні три блоки, при цьому шапка та підвал, повторюся на всіх сторінка даної ІС. Що стосується основної частини цієї сторінки слід зазначити, що вона складається з таких блоків як “Назва пам’ятки (місця відвідування)”, “Текстова інформації щодо даної пам’ятки (місця відвідування)” та “Каруселі з фотографіями” (див.рис.11).

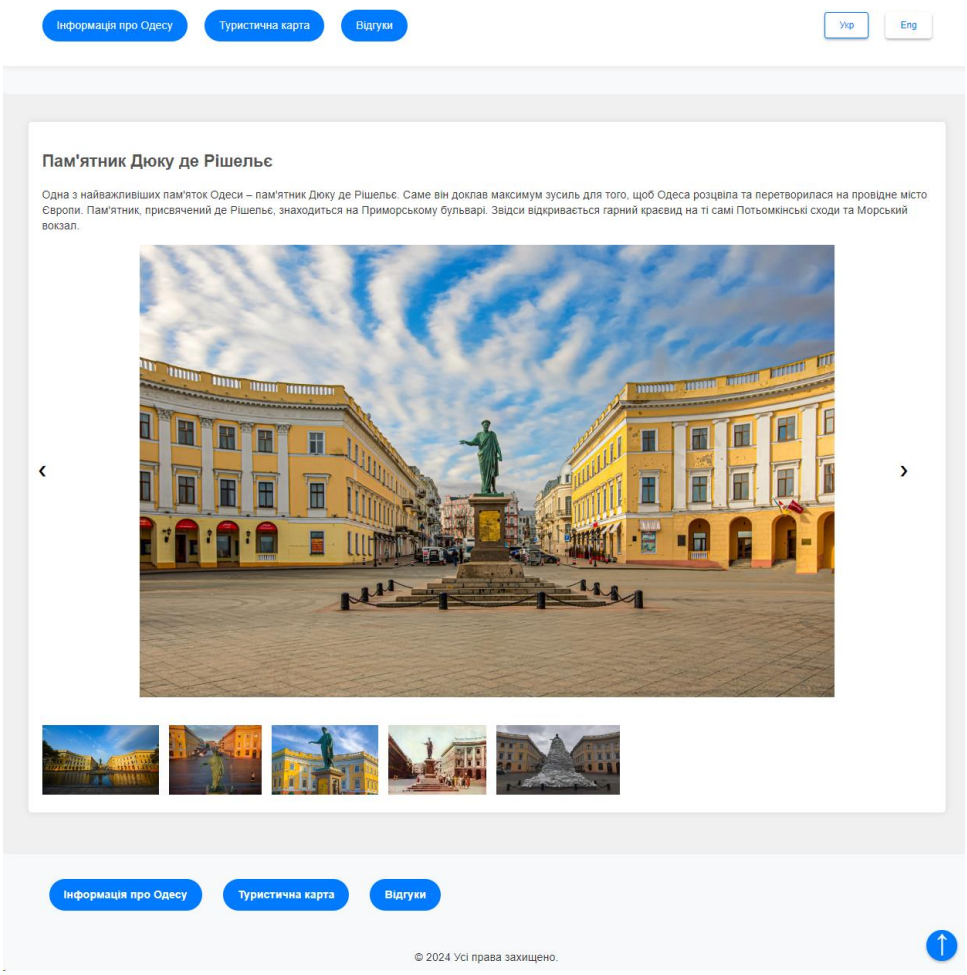

Рисунок 11 – Представлення реалізованого проєкту (внутрішня сторінка)

ВИСНОВКИ

Дана розробка має на меті створення інформаційної системи, яка дозволить користувачам ознайомитися з основними туристичними об'єктами міста Одеси через інтерактивну карту. У процесі проєктування та реалізації використовуються сучасні веб-технології, такі як HTML, CSS та JavaScript, для створення користувацького інтерфейсу, а також бібліотека Leaflet для відображення та маніпулювання картою.

Основна сторінка проєкту (index.html) містить загальну інформацію про місто Одесу, зокрема зображення та короткий опис. Також тут розміщені розділи з інформацією про місця відпочинку, туристичні об'єкти та форма для залишення відгуків. Внутрішня сторінка може містити детальну інформацію про певний туристичний об'єкт, наприклад, пам'ятник Дюку.

Головною функціональністю системи є інтерактивна карта, яка відображає розташування різних туристичних об'єктів у місті Одеса. Кожен об'єкт позначений на карті маркером, при натисканні на який відображається відповідна інформація.

Форма відгуків дозволяє користувачам залишати свої коментарі та оцінки про відвідані місця, а також вказувати контактні дані. Ця функціональність додає до системи елемент соціальної взаємодії та забезпечує зворотний зв'язок від користувачів.

Усі елементи системи оформлені за допомогою сучасних дизайнерських та інтерактивних практик, що забезпечують зручність користування та естетичний вигляд. Кнопки зміни мови та кнопки прокрутки мініатюр каруселі додані з міркуванням про зручність користування та естетику інтерфейсу.

Реалізація інтерактивної туристичної карти міста Одеси вимагає ретельного аналізу та вибору відповідної архітектури та веб-технологій. Важливо також приділити увагу безпеці, кешуванню, інтеграції з зовнішніми сервісами, тестуванню та використанню інструментів для спільної роботи над проектом. Критично важливим є ретельний аналіз та вибір відповідних технологій

та інструментів для успішної реалізації інтерактивної туристичної карти, яка буде зручною, масштабованою та надійною.

Процес розробки проєкту включав в себе декілька етапів. По-перше, проводився детальний аналіз вимог та визначення функціональності та структури застосунку. Потім відбувалася розробка користувацького інтерфейсу, включаючи створення HTML-шаблонів та CSS-стилів для оформлення сторінок. Наступним кроком була реалізація інтерактивної карти за допомогою JavaScript та бібліотеки Leaflet. Крім того, була реалізована функціональність форми відгуків, де користувачі можуть залишати свої коментарі та оцінки.

На фінальному етапі розробки важливим аспектом було тестування застосунку на різних пристроях та в різних браузерах, щоб забезпечити його коректну роботу та сумісність. Крім того, було важливо врахувати аспекти безпеки, зокрема захист від XSS атак та забезпечення конфіденційності даних користувачів.

Усі ці етапи розробки були виконані з урахуванням сучасних підходів до веб-розробки та найкращих практик, що дозволило створити функціональний та естетичний веб-застосунок (ІС) для відображення туристичної інформації про місто Одеса.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Проектування інформаційних систем. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160125164438/content-20160125164438.pdf (дата звернення 27.02.2024)
2. Теоретичні засади розробки туристичного застосунку як елемента діджиталізації у бізнес-середовищі туризму. URL: https://tourlib.net/statti_ukr/pashkevych.htm (дата звернення 01.03.2024)
3. What's the difference between monolithic and microservices architecture? URL: https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/#:~:text=%D0%9C%D0%BE%D0%BD%D0%BE%D0%BB%D0%B8%D1%82%D0%BD%D0%B0%D1%8F%20%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0%20%E2%80%94%20%D1%8D%D1%82%D0%BE%20%D1%82%D1%80%D0%B0%D0%B4%D0%B8%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C,%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%B7%D0%BC%D0%BE%D0%B2%20%D0%BE%D0%B1%D0%BC%D0%B5%D0%BD%D0%B0%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%D0%B8%20%D0%B2%D0%BD%D1%83%D1%82%D1%80%D0%B8%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B. (дата звернення 03.04.2024)
4. Microservices vs. monolithic architecture. URL: https://www.atlassian.com/ru/microservices/microservices-architecture/microservices-vs-monolith (дата звернення 26.03.2024)
5. Як мікрофронтенди впливають на розробку продукту. Власний досвід. URL: https://dou.ua/forums/topic/40912/ (дата звернення 19.03.2024)

6. Де шукати архітектурні тренди та що нас чекає у майбутньому. URL: https://dou.ua/lenta/columns/about-architectural-trends/ (дата звернення 07.04.2024)
7. React. URL: https://react.dev/ (дата звернення 13.04.2024)
8. Angular. URL: https://angular.io/ (дата звернення 12.03.2024)
9. Vue.js. URL: https://vuejs.org/ (дата звернення 01.04.2024)
10. A Comprehensive Comparison: Leaflet, OpenLayers, Mapbox GL JS, Google Maps JavaScript API, ESRI JavaScript API and D3.js for Interactive Web Maps. URL: https://norcomdentaire.medium.com/a-comprehensive-comparison-leaflet-openlayers-mapbox-gl-js-google-maps-javascript-api-esri-430c98b657fe (дата звернення 07.04.2024)