МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Проєктування карти безпечних прогулянок в місті Київ»

Студент 4 курсу групи КН-20	Керівник дипломної роботи
	_____
	(посада керівника)
Морозовський Владислав Сергійович	Чернозубкін Ігор Олександрович
(прізвище, ім'я та по батькові студента)	(прізвище, ім'я та по батькові керівника)
	До захисту
	(резолюція «До захисту»)
_____	10.06.24 _____
(підпис студента)	(дата)
	_____
	(підпис викладача)

Київ, 2024 рік

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 3

ВСТУП 4

РОЗДІЛ 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ 5

1.1 Історія створення веб-карт, їх види та особливості 5

1.2 Огляд та аналіз наявних аналогів 11

ВИСНОВОК ДО РОЗДІЛУ 1 16

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ 18

2.1 Ідеї до сервісу 18

2.2 Вибір мови програмування 18

2.3 Вибір фреймворку 21

2.4 Вибір БД та споспобу взаємодії з платформою 23

2.5 Спосіб зниження кількості коду 26

2.6 Вибір картографічного сервісу 29

2.7 Вибір фреймворку для шаблонної стилізації 31

2.8 Спосіб збереження паролю користувача 32

2.9 Сесія 37

ВИСНОВОК ДО РОЗДІЛУ 2 38

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СИСТЕМИ 39

3.1 Огляд структури проекту 39

3.2 Файли проекту 41

3.3 Способи використання системи 45

ВИСНОВОК ДО РОЗДІЛУ 3 46

РОЗДІЛ 4. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ 47

4.1 Вимоги для запуску системи 47

4.2 Тестування сервісу 47

ВИСНОВОК ДО РОЗДІЛУ 4 61

ПЕРЕЛІК СКОРОЧЕНЬ

API	Application Programming Interface
БД	База даних
JS	Java Script
JSON	Java Script Object Notation
EJS	Embedded JavaScript templates
HTML	Hypertext Markup Language
ПК	Персональний комп'ютер
GIS	Geographic Information System
ПЗ	Програмне забезпечення
ОС	Операційна система
OGC	Open Geospatial Consortium
XML	eXtensible Markup Language
SLD	Styled Layer Descriptor
WMS	Web Map Service

ВСТУП

У наш час майже в кожного є смартфон, на якому встановлено величезнакількість додатків. Одним із них є веб-карти (наприклад додатки Google Maps, Apple Maps та інші). Вони стали невід'ємною частиною нашого життя тавикористовуються майже у всіх сферах. А саме: для замовлення таксі через додаток, або відслідковування поточного місцезнаходження громадського транспорту, замовлення їжі, прокладання оптимального маршруту від точки А до Б і т.д. Тобто способи використання веб-карт величезні, від побутових запитів - до планування логістики компаніями.

Але якщо потрібно зробити щось одне - наприклад обрати місце для прогулянки у місті, наприклад, Києві. То перед нами з'являється невеличка проблема - велика кількість інформації, яку важко обрати, або надмірний інтерфейс додатку, через який складно сфокусуватися на основному, або реклама, яка усіма силами намагається переманити нашу увагу. Також, не меншим завданням стає питання безпеки для прогулянок. Через це постає питання – як вирішити ці проблеми.

Рішення є – створити простіший сервіс, який буде базуватися на вирішенні цих цілей, а саме пошуку визначних місць, котрі являються самі по собі безпечними, з мінімально потрібним інтерфейсом та інтуїтивно простим дизайном.

Отже, у цій дипломній роботі виконано проект, який присвячений швидкому та зручному пошуку визначних місць на інтерактивній карті, де описані переваги цих місць над іншими. Звісно, дана робота вирішує ті проблеми, які описані вище.

РОЗДІЛ 1
ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Історія створення веб-карт, їх види та особливості

По- перше, веб- карта- це інтерактивне відображення географічної інформації у вигляді веб- сторінки. Раніше протягом декількох десятків років більша частина цифрової географічної інформації була обмежена для використання тільки на настільних (стаціонарних) персональних комп'ютерах або внутрішніх мейнфреймах (це комп'ютер, який, частіше, використовується великими організаціями для масової обробки дуже великого об'єму даних) і не могла бути легко поширеною між іншими організаціями. GIS (системи збору, зберігання, аналізу та графічної візуалізації просторових даних тапов'язаної з ними інформації про необхідніоб'єкти)- аналітики отримували доступ до даних тільки зі своїх власних комп'ютерів на робочому місці, які частіше за все були підключені до центрального файлового сервера, який знаходився десь в офісі. Для того щоб переглядати або щось робити з цими даними потрібне було спеціалізоване ПЗ, що звісно знижувала таргетовану аудиторію, тобто ту кількість людей, які могли б отримати користь від цієї GIS- інформації.

З появою Інтернету та поширенню його масової популярності в середині 1990- х років людство почало замислюватися про те, яким чином карти та іншу географічну інформацію можна використовувати на комп'ютерах, не тільки в середині організацій, так і поміж широкої громадськості. Першим кроком булорозміщення статичних зображень карт (це перший вид (усього їх два))[1]. Приклад статичної карти зображено на рис. 1.1.

Рисунок 1.1 – Приклад використання статичної карти

Переваги:

- Прості та дешеві у створенні.
- Легкі у використанні - не потрібні технічні навички.
- Розробник має повний контроль над тим, що користувач побачить(адже це просто фото).

Недоліки:

- Користувачі не можуть налаштувати карти відповідно до своїх потреб (наприклад поставити мітку), (юзер може тільки завантажити цю карту собі на пристрій та редагувати як завгодно, але загрузити знову на той сайт, з якого взяв ту мапу вже не можна бути (звичайно в тому випадку якщо людина не має відповідного доступу(типу адміністратор)).
- Щоб оновити або змінити карти, розробник зазвичай повинен створити нову карту.
- Може бути важко переглянути детальну інформацію, якщо користувачі не можуть збільшити масштаб[3].

Далі задумалися про те як їх зробити інтерактивними. Спочатку з'явилися динамічні карти(це був другий тип карт), тобто ті, які малюють карту не у веб- браузері, а на сервері в момент запиту і потім відправлять браузеру вже готове зображення потрібної карти. Приклад роботи динамічної веб- карти зображено нище, на рис.. 1.2.

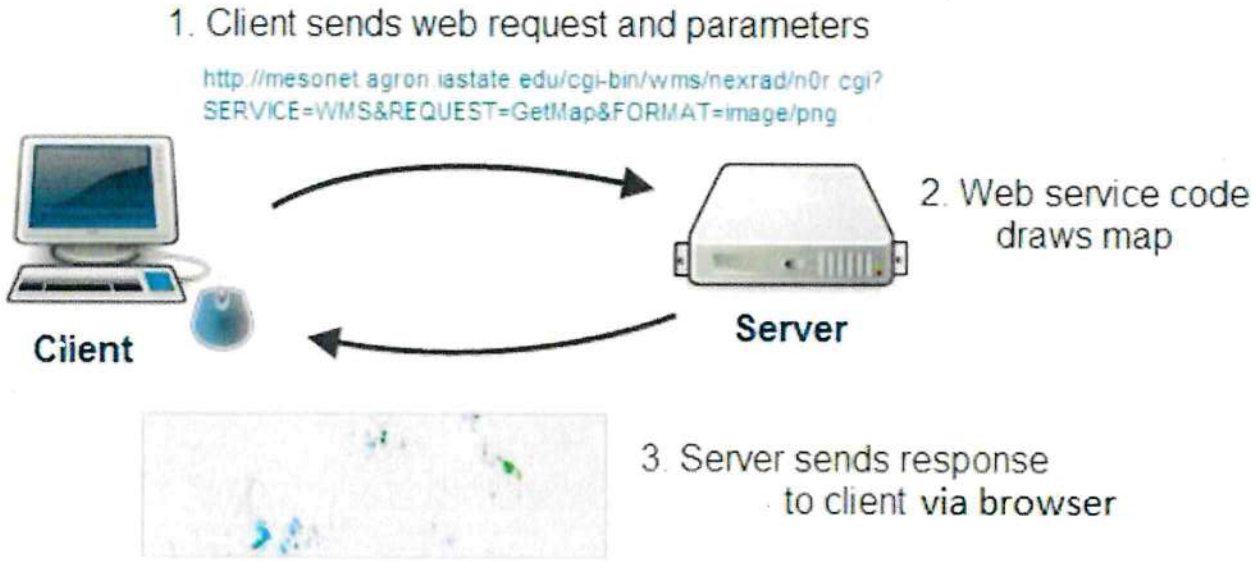


Рисунок 1.2 - Схема роботи динамічної веб-карти

Переваги динамічно намальованих веб-карт:

- Огляд об'єктів, що постійно змінюються. Через те, що динамічно намальовані картографічні служби отримують дані та малюють їх під час запиту, вони корисні для отримання даних, які постійно змінюються. Такі карти ,в основному, краще використовувати коли потрібно відобразити багато функцій, які змінюються одночасно (наприклад, положення кожного транспортного засобу у великому автопарку, або поточне місцезнаходження літака).
- Створення власних, складних, легко налаштовуваних карт. Які,динамічно намальовані за допомогою WMS (стандартний протоколдля обслуговування через Інтернет географічно прив'язаних зображень), дозволяють застосовувати широкий спектр символів і стилів за допомогою концепції, яка називається SLD (це схема

XML(мова розмітки подібна до HTML), задана OGC(міжнародна некомерційна організація, що веде діяльність з розробки стандартіву сфері геопросторових даних та сервісів)) для опису зовнішнього вигляду шарів карти.

- Широкий спектр взаємодії. Користувачі можуть працювати з даними карти за допомогою масштабування, спливаючих вікон принаведенні курсора тощо.

Недоліки :

- Тривале очікування. Поки сервер намалює карту, може пройтибільше часу ніж ми звикли, особливо якщо для візуалізації потрібнобагато шарів. Час очікування в 2-3 секунди, який може вважатися прийнятним для людей, обізнаних з цією технологією, але може бути неприпустимим для користувачів веб-карт, які цього не очікують. Зараз усім кортить, щоб кожна карта працювала так самошвидко, як карти Google.
- Ціна. Більш дорожчі і складніші у розробці ніж статичні карти.
- Користувачам можуть знадобитися технічні навички, щоб повністю використовувати усі можливі функції.

Також вони не є "гарно" масштабованим типом послуг (масштабованістьозначає, скільки клієнтів можна обслуговувати одночасно). Для доступу до даних і малювання зображення на карті сервер повинен виконувати роботу, і якщо велика кількість людей одночасно робить запит на проглядання карт, сервер може бути перевантажений, та навіть, у найгіршому випадку, зупинити свою роботу.

До недоліків можна віднести те, що розробник має менше контролю над тим, як користувачі переглядають та інтерпретують дані (наприклад одна людина може створити мітку та прикріпити фото нецензурного формату, або зробити допис, який заставить юзерів панікувати)[2].

Перші динамічні карти, які обслуговувалися початковими версіями програмного забезпечення, такими як Map Server і Esri ArcIMS, були не такимияк зараз, мали погану роздільну здатність, та повільну швидкість завантаження. Через це картографи масово ще не вийшли на ринок карт. Однак ці ранні мапибули дуже революційними на той час існування. Також, кожного разу як на динамічній карті щось змінювалось (наприклад літак починав своє переміщення) то сервер малював карту повністю заново і відсилав її браузеру замість того щоб змінити тільки ту зону, де літак змінював свої координати (уявіть яке навантаження було на сервер при масових запитів на карту).

Але все сильно змінилося з розповсюдженням масового використання кешів (особливої швидкісної пам'яті, де знаходяться копії часто використовуваних даних). Веб-карти значно зросли за вище згаданими двома показниками (швидкістю та масштабованістю), коли веб-сайти почали видавати зображення карт із попередньо згенерованих кешів. Навіщо робити запит серверу динамічно малювати карту, якщо можна попередньо намалювати всі можливі протяжності карти в розумному наборі масштабів і потім дуже швидконадіслати їх користувачу Після того, як зображення карти були намальовані та кешовані, можна було побачити зображення як мозаїку з плиткою.

З'явився плитковий (або "кешований") картографічний сервіс. У цихтипах служб сервер просто надсилає маленькі квадратні зображення (плиточки)карти, які він зберігає у кеші на диску. Приклад роботи плиткової карти зображено на рис. 1.3. Ці зображення зазвичай генеруються адміністратором сервера в різних масштабах, перш ніж сервіс стане загальнодоступним. Тоді, коли веб-користувачі роблять запит на те, щоб переглянути карти, сервер не повинен виконувати роботу з малювання усієї карти; він може просто надіслатиназад ті плитки, які потрібні для заповнення запиту карти. Звісно набагато швидше відіслати частинку зображення аніж повне (повторюючи приклад з відображенням літака - коли він змінює свої координати серверу не потрібно

заново малювати усю карту - достатньо згенерувати тільки той шматок карти, де знаходиться повітряний транспорт).

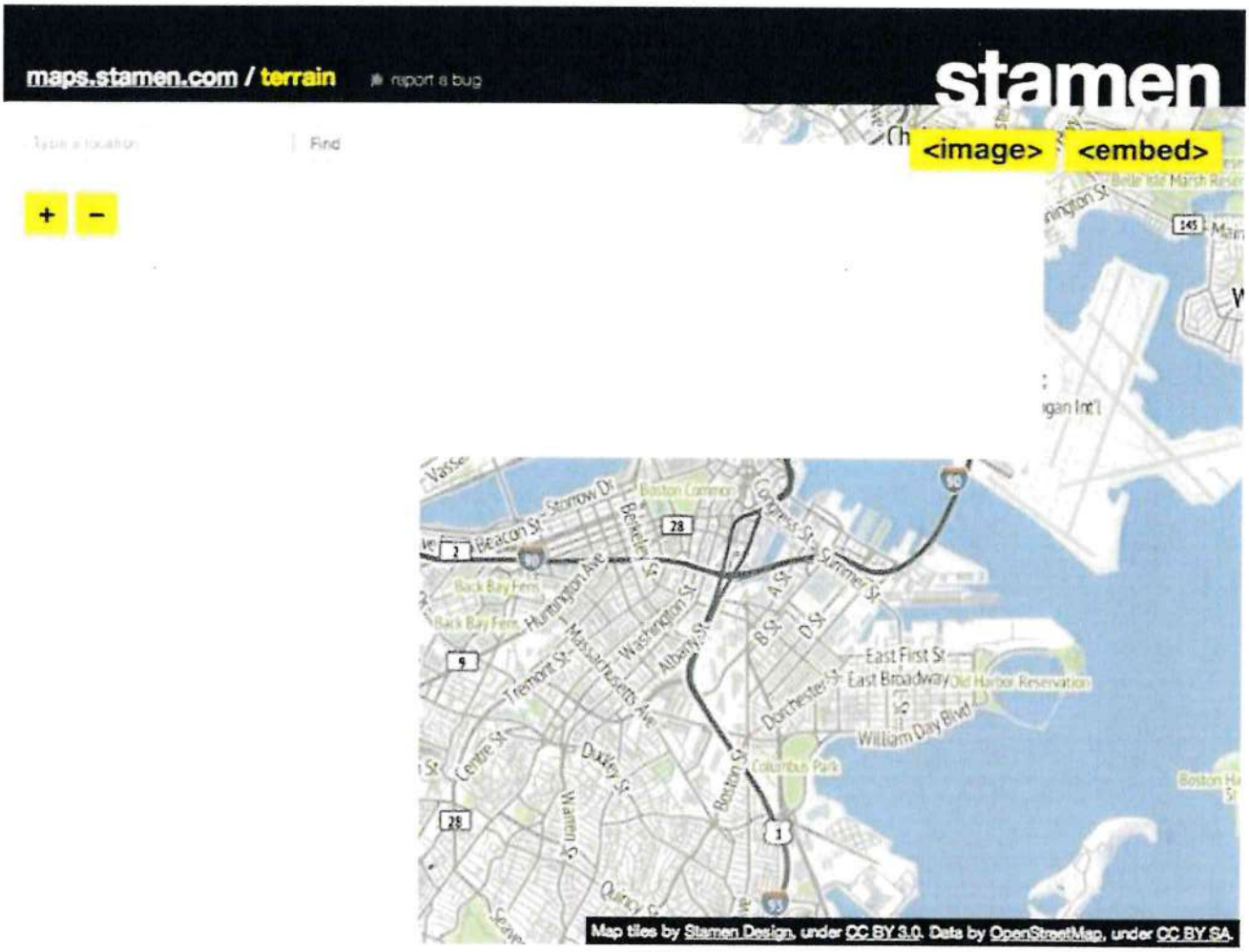


Рисунок 1.3 - Схема роботи плиткової веб-карти[5]

Першим сервісом, що використовує цю технологію, став Google Maps. Наслідуючи приклад цієї компанії, багато сайтів також почали надавати" заздалегідь підготовлені " зображення фрагментів і карт, використовуючи креативну техніку, звану асинхронним JavaScript і XML, усуваючи всюдисущі і дратівливі спалахи, які виникали після будь-яких навігаційних операцій на попередніх веб-картах. Тепер серверу працювати набагато простіше, а затримка практично повністю зникла. Технологію плиточних карт можна назвати революцією «веб 2.0".

Звісно плиткові карти також мають свої унікальні недоліки. Найбільшийз них – це витрати часу та потужності сервера, необхідні для того щоб створювати кеш, а також дисковий простір, який необхідний для його зберігання.

Плиткові карти також можна розділити на 2 категорії: растеризовані та векторні. Спочатку про растеризовану плиткову карту. Веб-карти, засновані на цій технології, застаріли, але все ще широко використовуються. У комп'ютерній графіці растрова графіка-це матричні структури даних, які зазвичай є прямокутною сіткою пікселів (кольорових крапок). Растрові зображення зберігаються у файлах зображень різних форматів. Фрагменти растрової карти насправді є лише зображеннями. Растрові карти з можливістю масштабування складаються з багатьох фрагментів растрових карт (у форматі .png або .jpg), розміщених поруч один з одним, упорядкованих за схемою, яка являє собою піраміду, що зображено на рис. 1.4. Цей надзвичайно розумний трюк дозволяє переглядати лише невелику частину карти, не завантажуючи її цілком, зберігаючи відчуття, що ви досліджуєте один великий документ[6].

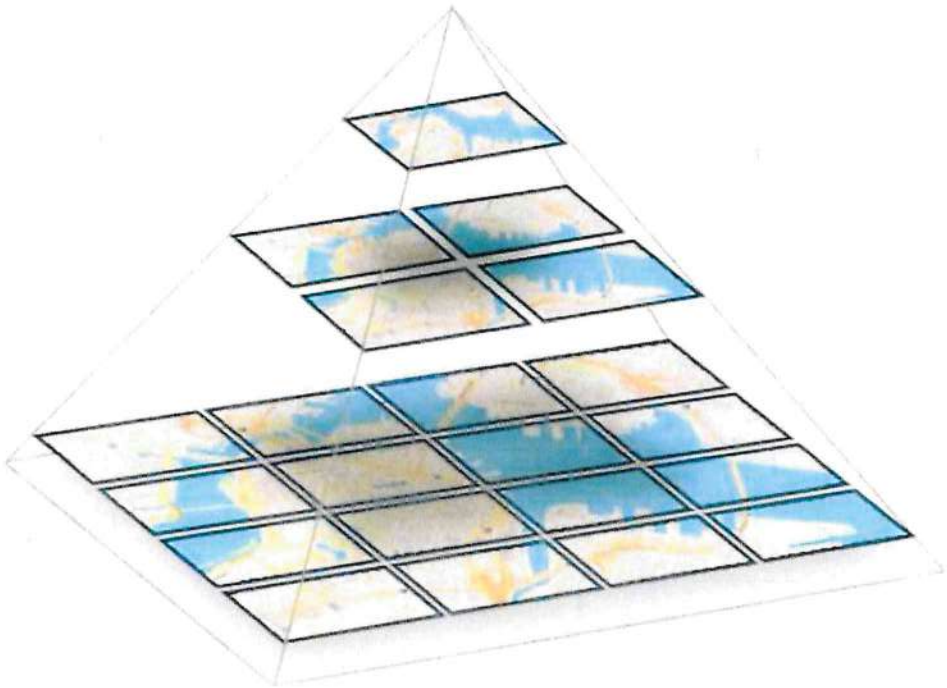


Рисунок 1.4 – Растрована веб-карта у вигляді піраміди

Оскільки растрові плитки є зображеннями, з'єднаними разом, їх можна масштабувати та панорамувати, але це не надає можливості стилізації з боку користувача (бо він отримує фото). Також растрові плитки мають досить великий розмір, і тому час завантаження під час панорамування та

масштабування на карті може бути довшим, залежно від швидкостіпідключення до мережі.

Переваги:

- Відтворення на сервері – через це вимоги до обладнання кінцевогокористувача є нижчими (адже відображення зображення не вимагаєзначних ресурсів).

Недоліки:

- Більший розмір – потрібно більше місця на сервері.
- Більше споживання пропускної здатності та відносно повільний часзавантаження (під час панорамування та масштабування на карті).
- Не плавне покрокове масштабування.

Векторна карта знаходиться поруч із картою растрових фрагментів. Потім була впроваджена ця технологія, яка також дозволяє передавати дані, розділені на квадратні фрагменти. Однак ці фрагменти не є растровими зображеннями; вони складаються з математичних інтерпретацій геометричних об'єктів, таких як точки, криві та багатокутники. [4]. Приклад векторної карти зображено на рис. 1.5.

Рисунок 1.5 – Приклад векторної карти[7]

Головною особливістю цієї карти є те, що зображення малюється на стороні клієнта, а не на сервері, оскільки сервер надсилає дані, які браузер

користувача використовує для інтерпретації карти. Векторні фрагменти відображаються у стилі, який є невеликим текстовим файлом, який визначає зовнішній вигляд та відображення певного елемента карти (наприклад, усіх доріг), а векторні фрагменти зазвичай надсилаються як пакети географічних даних, що складаються з окремих фрагментів у форматі .pbf (protocolbuffer binary format).

Переваги:

- Менший розмір це менша вимога до місця на сервері (векторніплитки становлять приблизно 20–50% розміру растрових плиток).
- Швидший час завантаження та нижче споживання пропускноїздатності (через менший розмір).
- Плавне масштабування.
- Відображення з високою роздільною здатністю на всіх рівнях масштабування без збільшення розміру файлу.
- Легке налаштування у режимі реального часу (векторні плитки дозволяють легко змінювати вигляд карти на льоту з використанням мінімальних ресурсів (бо клієнту передається не готове зображення а пакет географічних даних).

Недоліки:

- Для візуалізації на клієнтському пристрої потрібне не "слабке" обладнання (у порівнянні з відображенням просто зображення (у випадку використання растрової плиточної технології)) .

Звичайно можна змішувати растрові плиткові карти з векторними та отримуват найкращий результат з обох, наприклад, супутникова карта (растрові плитки) з накладенням вулиць з мітками, доступними різними мовами (векторні плитки). У табл. 1.1 якраз наведіні такі приклади сервісів.

1.2 Огляд та аналіз наявних аналогів

Таблиця 1.1 - Порівняння наявних сервісів з інтерактивними картами

Назва	Google Maps	Apple Maps	Bing Maps
Платформи	Усі	IOS, macOS, iPadOS	Усі
Прокладання маршруту	+	+	+
Створення зупинок на маршруті	+	+	+
Створення міток	+	+	+
Персональні данні	збираються	Не збираються	збираються
3d перегляд	+	+ (найкращий на ринку)	+
Можливість залишати відгук через дану систему(а не через іншу)	+	-	-
Реклама	+	-	+
Затримка під час швидкого масштабування	-	-	+

Google Maps — це веб-картковий сервіс, розроблений компанією Google у 2005 році. Має 154.4 мільйони користувачів за місяць. З рис. 1.6 видно величезний недолік – реклама при пошуку якогось місця. Також усі дії що робляться користувачем на даному сервісі зберігаються. Тобто ніякої приватності інформації.

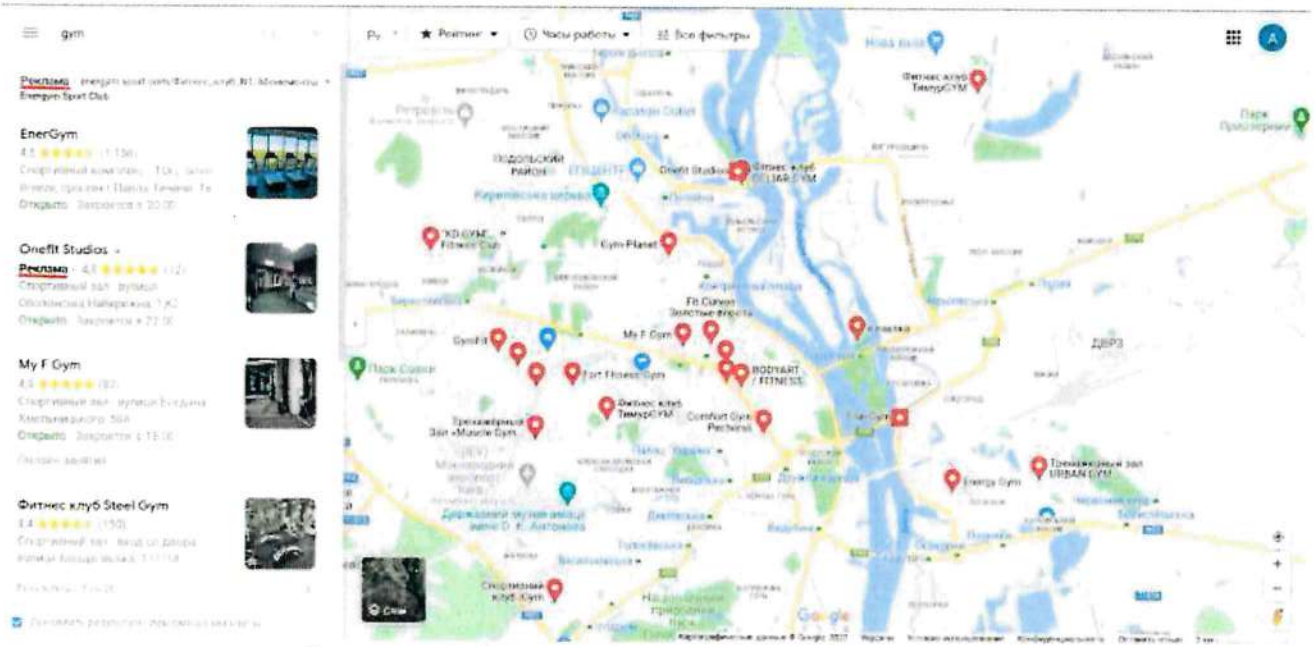


Рисунок 1.6 – Інтерфейс Google Maps на комп'ютері

Apple Maps — веб-картковий сервіс, вигляд якої можна побачити на рис. 1.7, розроблений компанією Apple у 2012 році. Має, на мою думку, привабливіший інтерфейс за Google Maps. Відсутня реклама, данні користувача приватні – не зберігаються. Шукати потрібні речі дуже зручно.

Рисунок 1.7 – Інтерфейс Apple Maps на комп'ютері

Bing Maps-це веб-картографічний сервіс, зовнішній вигляд якого був розроблений Microsoft у 2005 році, як показано на рис. 1.8. Він має дуже зручний і надлишковий інтерфейс, а саме: є кнопка для роздрукування маршруту і залишення відкликання. Крім того, при швидкому збільшенні зображення карта не встигає правильно відобразитися - виникає затримка.

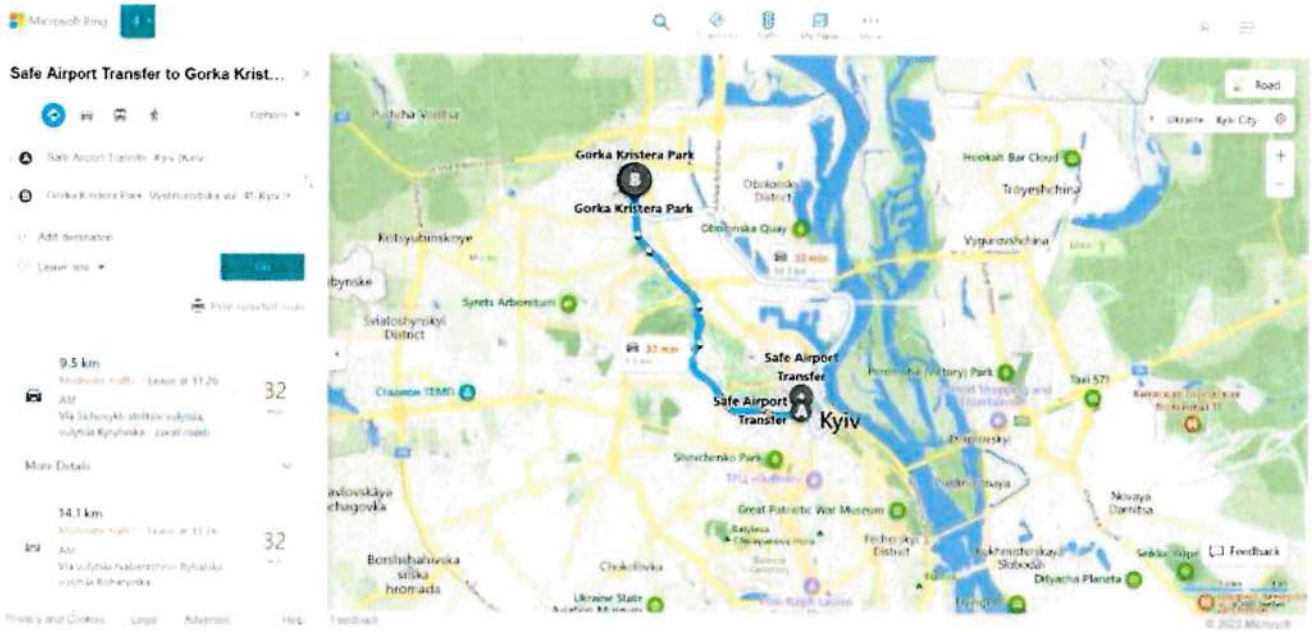


Рисунок 1.8 – Інтерфейс Bing Maps на комп'ютері

Ці програми можуть виконувати багато різних функцій. Але якщо вам потрібно вирішити 1 завдання – знайти пам'ятки або вирішити, куди піти погуляти в саду, зайвий інтерфейс може викликати дискомфорт і уповільнити час вибору. Тому ми вирішили створити сервіс, щоб вирішити 1 завдання-знайти місце для проведення вільного часу в Україні з мінімально необхідним інтерфейсом і без реклами, а також мати можливість залишити відгук про місце не через цей сервіс, а через інший сервіс.

ВИСНОВОК ДО РОЗДІЛУ 1

Отже, в цьому розділі були проведені дослідження з аналізу історії створення і появи веб-карт, розглянуті їх основні типи. Вони можуть бути статичними і динамічними (їх ще називають інтерактивними), а також їх різновиди, а саме Плиткові, растеризовані і векторизовані. Далі показані їх особливості і проведено порівняльний аналіз.

Також були показані основні області застосування інтерактивних карт, проаналізовані і зіставлені один з одним сучасні методи їх реалізації.

Грунтуючись на отриманих даних, можна сказати, що картографічні сервіси сьогодні дуже популярні і кожен день використовуються мільйонами людей для виконання роботи. Але вони в основному покладаються на моделі для вирішення широкого кола завдань – через це у них надлишковий інтерфейс, а також вони збирають і обробляють інформацію в своїх інтересах, наприклад, для реклами. Тому я вирішив реалізувати цю актуальну тему, в якій є модель для вирішення 1 завдання - створення міток на карті та дій з ними для визначення безпечності. Завдяки цьому зникає проблема надлишкових інтерфейсів і звичності призначених для користувача даних.

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СИСТЕМИ

2.1 Ідеї до сервісу

Перед тим як починати розробляти сервіс потрібно знати, що можливо створити для нього, для того щоб обрати технології для його реалізації. Ідеї:

- створення нового користувача та авторизація
- хешування паролів (шифрування)
- створення дописів, в яких є можливість додавати зображення.
- проглядання дописів
- відображення створених дописів у вигляді міток на інтерактивнійвеб-карті
- редагування та оцінювання дописів
- пошук визначних місць за назвою або місцезнаходженням
- коректна логіка для URL - запитів
- сеанс

2.2 Вибір мови програмування

Необхідно було вибрати мову, який був би досить популярним (якщо у вас виникне проблема, ви легко зможете знайти рішення в Інтернеті), серверним (повернемося до того, якою мовою був обраний JavaScript.

JavaScript — це скриптова мова програмування, яка є однією з трьох основних мов, що використовуються для розробки веб-сайтів. У той час як HTML і CSS створюють структуру і стиль веб-сайту, JavaScript дозволяє

додавати функціональність і поведінку на веб-сайт, дозволяючи відвідувачам взаємодіяти з вмістом багатьма різними способами. Я гарно запам'ятав основнуфункцію цієї мови за допомогою виразу «The purple dino dance», де dino (динозавр – це HTML - розмітка), purple (фіолетовий – це прикметник, тобто оформлення - CSS), a dance (танцює – тобто дія - JavaScript). Також вона є переважно клієнтською мовою, тобто вона інтерпретується у браузері. Але також може виконуватися на сервері за допомогою фрейвморка Node.js.

Історія його виникнення. У 1995/99 році програміст Netscape Брендан Ейх всього за 10 днів розробив нову скриптову мову. Спочатку він називався Mocha, але незабаром став відомий як LiveScript, а згодом і як JavaScript. 2005 рік став знаменним для цієї мови. Було представлено ajax та набір інноваційних технологій, включаючи JavaScript. Значно покращено користувальницький досвід, завдяки чому веб-сторінки виглядають більш автентично. Тоді JavaScript дійсно привернув увагу як професійну мову програмування. [8].

Переваги:

- Однією з головних переваг JavaScript є те, що він не вимагає дорогих засобів розробки. Оскільки це мова, яка інтерпретується в контексті веб-браузера, ви можете почати з простого текстового редактора, такого як Notepad. Наприклад, для програмування на Python вам потрібно буде встановити середовище розробки, звичайно, є безкоштовна версія Community Edition, але вона повністю відповідає атмосфері цієї мови.
- Популярність (величезне коло людей, які використовують дану мову).

- Менше взаємодії з сервером - можна перевірити те , що користувачввів перед відправкою сторінки на сервер. Це економить трафік сервера, та менше навантажує його.
- Миттєвий зворотній зв'язок – не потрібно чекати перезавантаженнясторінки, щоб побачити, чи було правильно введені данні в усіх полях, наприклад.
- Інтерактивність-дозволяє створити інтерфейс, який реагує, коли користувач наводить на нього курсор миші або активує за допомогою клавіатури.

Недоліки:

- Відсутність засобу налагодження. Хоча деякі редактори HTML підтримують налагодження, вони не настільки ефективні, як інші редактори, наприклад C/C++.
- Не многопоточна.
- У різних браузерах інтерпретація коду може виглядати по-різному.
- Через різних алгоритмів роботи браузера можуть виникнути проблеми із захистом інформації (якщо з цього приводу не було прийнято ніякого рішення).

На рис. 2.1 можна побачити, що мова програмування JavaScript найпопулярніша для веб розробки.

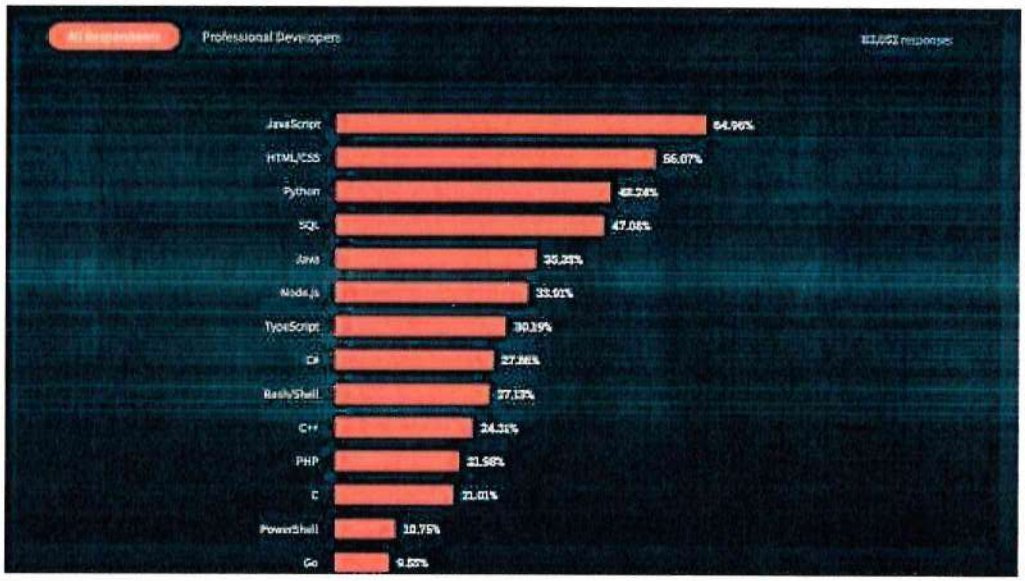


Рисунок 2.1 - Список популярності використання інструментів та мовпрограмування для веб розробки

2.3 Вибір фреймворку

Вам потрібно реалізувати внутрішню логіку, наприклад, запити URL-адрес, і захистити їх. Ви не можете зробити це в JavaScript, вам потрібно вибрати фреймворк (це програмне забезпечення, яке розробники розробляють і використовують для створення додатків). Найбільш популярними з них є::

- Django (Python).
- Laravel (PHP).
- Ruby on Rails (Ruby).
- ExpressJS (NodeJS - JavaScript).
- CakePHP (PHP).
- Flask (Python).
- Spring Boot (Java).

Щоб вибрати ExpressJS, вам потрібне середовище розробки під назвою Node (неможливо використовувати цей фреймворк, використовуючи лише JavaScript).js - це платформа розробки з відкритим кодом для запуску коду JavaScript на стороні сервера, а також величезна кількість безкоштовних бібліотек та модулів, розроблених та опублікованих розробниками. На малюнку 2.2 показано кількість модулі та бібліотеки, а також кількість модулів та бібліотек у node.js.

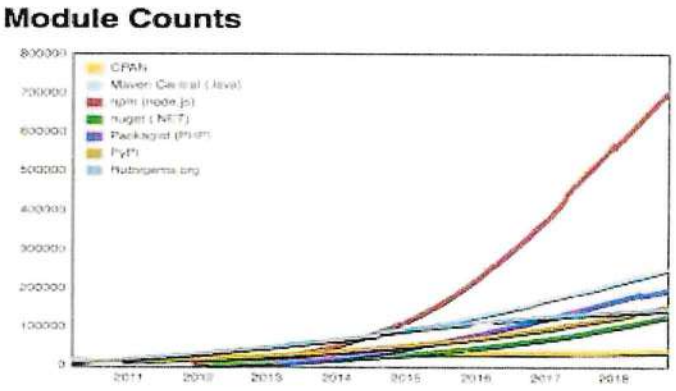


Рисунок 2.2 – Порівняльний список кількості модулів для різних мов програмування

Історія його походження. Node.js пройшло вже близько тринадцяти років. У 2009 році платформа була написана Райаном Далем. На початку цього року node.js доступний лише в Linux та Mac OS. Райан сам підтримував і розвивав продукт, а пізніше йому почав допомагати спонсор Joyent. Які умови сприяли розробці цього програмного забезпечення? У 2009 році найпопулярнішою веб-службою був HTTP-сервер Apache, який міг працювати з величезною кількістю з'єднань, понад 10 000 за одиницю часу. Але коли якийсь код був заблокований (скільки користувачів підключено до цього сервера), виникла проблема, яку потрібно було вирішити. У 2009/11/8 на Jsconf (Європейської конференції) була проведена демонстрація Node.js, який може бути використаний для вирішення проблем із сервером Apache HTTP. Також node.js це комбінація Chrome V8JavaScript (назва механізму) і низькорівневого інтерфейсу прикладного програмування для введення-виведення і циклів обробки подій.

Через те, що браузер намагався заманити користувача до себе, виникла велика конкуренція, була підвищена продуктивність і вдосконалений механізм JavaScript. Основні браузери старанно працюють над пошуком способів прискорити роботу скриптової мови і поліпшити його підтримку. Отже, nodes.js був створений у потрібному місці та в потрібний час. Він представив різні підходи до розробки серверної частини на JavaScript та інноваційне мислення, які допомогли багатьом розробникам.

1. Один з головних факторів node popularity.js, як уже згадувалося, підтримує велику кількість бібліотек з відкритим кодом. Навколо node існує дуже активна спільнота node.js, що проводить багато заходів та конференцій, таких як Node.js Summit, node.JS interactive і NodeConf [9].

Платформою розробки є node.Що стосується js, то зараз був обраний найпопулярніший фреймворк ExpressJS. Він встановлюється за допомогою менеджера пакетів Node. Він має ряд методів та плагінів, які ви можете використовувати для створення серверної частини вашої програми.

Основні функції expressJS:

- Запуск сервера та очікування запитів.
- Обробка вхідних запитів (перетворення запитів на об'єкти).
- Робити відповідності до запитів (тобто, виконувати код тількиякщо, наприклад запит був на адресу /home).
- Змінювати status code (код сервера), змінювати headers (код, якийпередає дані між веб-сервером і клієнтом) ітд.

Компанії, які використовують ExpressJS: Netflix, IBM, ebay, Uber.

2.4 Вибір БД та способу взаємодії з платформою

Бази даних — це більше, ніж просто спосіб зберігання. Зазвичай вона може обробляти щонайменше 1 тонну даних, ефективно зберігати їх, отримувати доступ до них, стискати до менших розмірів і робити їх зручними для управління. Вони також надають багато інструментів розробки, що, мабуть, є найважливішою частиною. І такі речі існують, особливо функції безпеки, адміністративний доступ, контроль над тим, у кого він є, хто що може робити. Це дуже важливо для великих додатків і великих команд з великою кількістю розробників, а також для забезпечення безпеки даних.

Була обрана БД MongoDB. В ній є такі популярні стеки (набори), для роботи з даною БД:

- MEAN (M-MongoDB, E-ExpressJS, A-AngularJS, N-NodeJS).
- MERN (M-MongoDB, E-ExpressJS, R-ReactJS, N-NodeJS).

З рис. 2.3 можна побачити, що в обох випадках використовуються вже обраний фреймворк ExpressJs та платформа NodeJS. Також БД MongoDB використовує JavaScript, як мову для запитів.

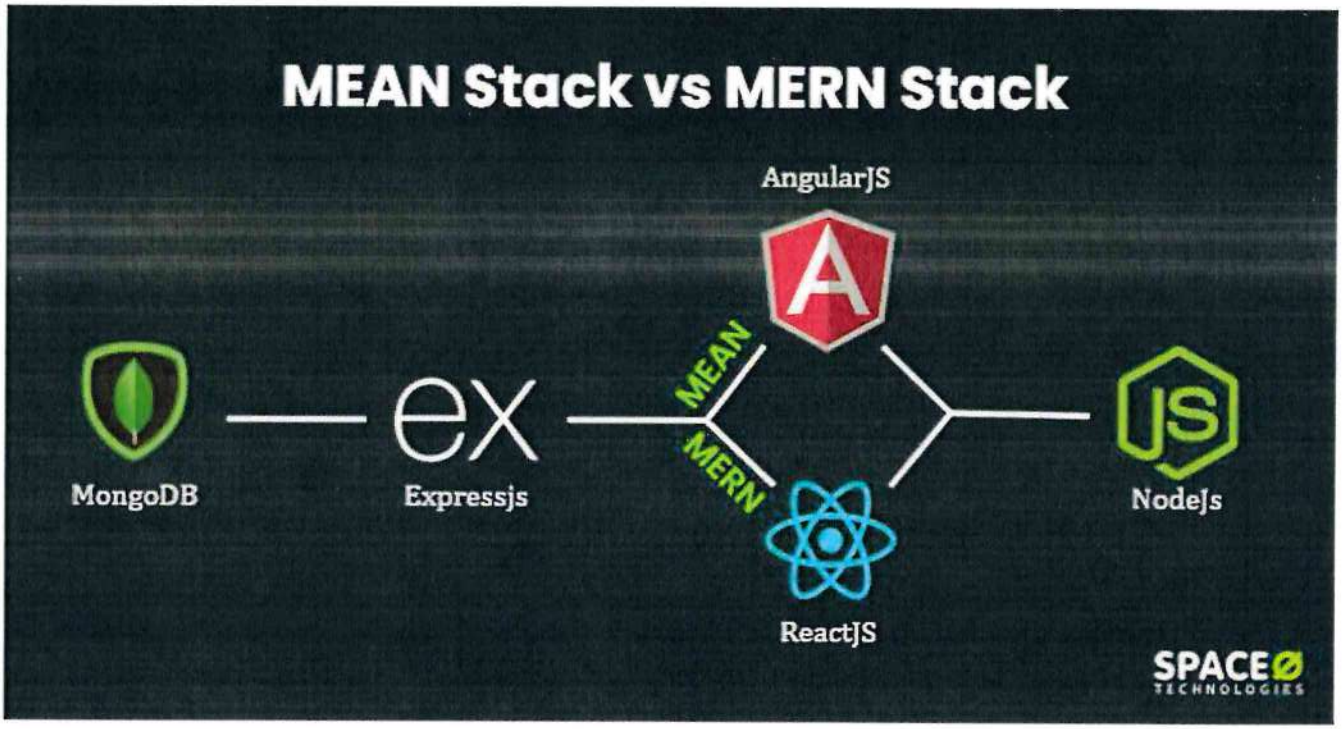


Рисунок 2.3 – Різниця між MEAN та MERN наборами

MongoDB-це тип сервера баз даних, орієнтований на документи, розроблений мовою програмування C++. Слово Mongo походить від huge (величезний). MongoDB-це сервер баз даних з відкритим кодом. Ця база даних має чудову продуктивність, оскільки вона використовує динамічні схеми баз даних, такі як JSON, замість традиційних таблиць та реляційних систем баз даних структурованою мовою запитів (SQL), що робить зберігання даних швидким та відносно простим (тобто структура збережених документів може відрізнятися один від одного). Таким чином, MongoDB відноситься до категорії серверів баз даних NoSQL (мова неструктурованих запитів). Динамічна Схема бази даних, що використовується MongoDB, називається BSON (двійкове позначення Об'єктів Javascript).

Історія. MongoDB був заснований у 2007 році командою DoubleClick у складі Дуайта Меррімана, Кевіна Райана та Еліота Горовіца. Рекламна компанія DoubleClick (зараз належить Google) розробила та використовувала різні сховища даних користувачів, щоб обійти недоліки існуючих баз даних. Компанія показала 400 000 оголошень в секунду, але в багатьох випадках вона мала проблеми як з масштабованістю, так і з гнучкістю. Розчарована,

команда була хотіла створити базу даних, яка б вирішувала проблеми, з якими вона зіткнулася в DoubleClick. Тоді і була розроблена MongoDB[10].

Як MongoDB взаємодіє з nodes.js? Використовуйте mongoose. Це бібліотека моделювання об'єктних даних (ODM (object Data Mapper)) для MongoDB та Node.js... Він використовується для управління взаємозв'язками між даними, забезпечення перевірки схеми, перетворення між об'єктами в коді та представлення цих об'єктів у MongoDB. На малюнку 2.4 ви можете побачити, що дані з бази даних надходять у node.js як Об'єкт JavaScript.

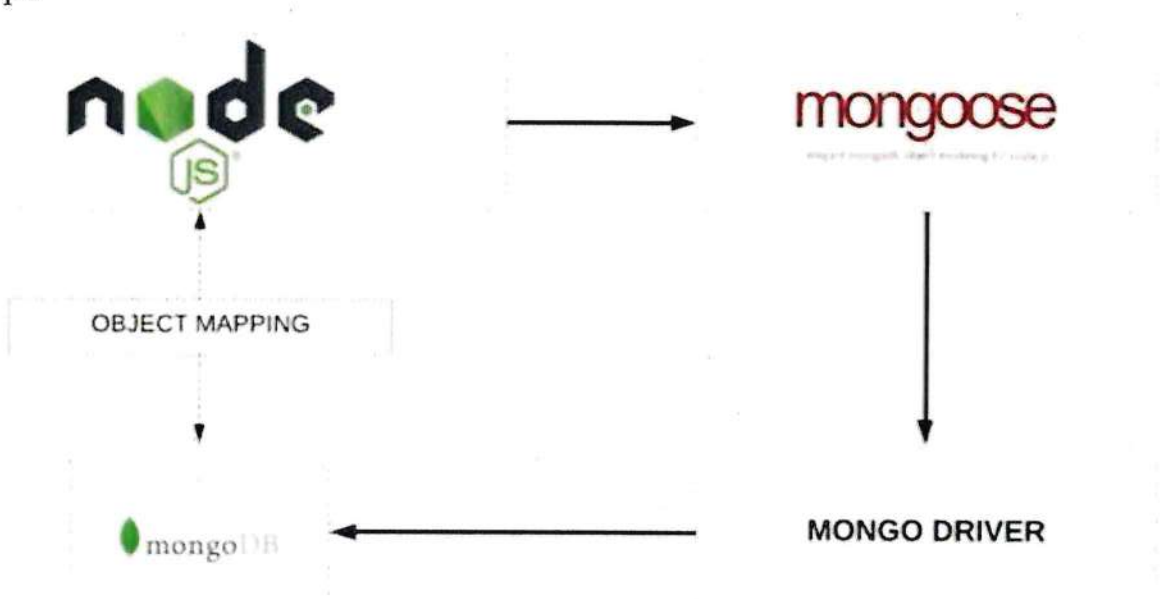


Рисунок 2.4 – Схема взаємодії БД MongoDB з платформою Node.js за допомогою бібліотеки mongoose

Застосунок також має можливість додавати зображення при створенні орієнтира. Однак база даних MongoDB має обмеження в 16 МБ, що дуже мало для зображень, тому вони не зберігаються в базі даних MongoDB.

Звичайно, MongoDB Atlas-це хмарна база даних, яка пропонує лише 512 МБ безкоштовного використання на місяць, що занадто мало. Тарифний план показаний на малюнку 2.5.

Cluster Tier	Storage	RAM	vCPUs	Base Price
M0	512 MB	Shared	Shared	Free forever
M2	2 GB	Shared	Shared	$9/mo
M5	5 GB	Shared	Shared	$25/mo

Рисунок 2.5 – Кількість ресурсів у безкоштовному пакеті хмарного сховища MongoDB Atlas

Тому було вирішено обрати альтернативу – cloudinary (хмарне сховище).Воно дає 25 безкоштовних Гб на місяць, що зображено на рис. 2.6, та має дужебагато корисних інструментів для маніпулювання зображеннями.

Рисунок 2.6 – Кількість ресурсів у безкоштовному пакеті хмарного сховища cloudinary

2.5 Спосіб зниження кількості коду

У даному сервісі є багато шаблонних сторінок, в яких однаковий початокheader (шапка (верхня частина сторінки)) та footer (нижня частина), які можнаб було постійно прописувати і тоді б код був би громіздкий.

Які є найпопулярніші інструменті для вирішення даного питання:

1. Handlebars-це мова шаблонів, приклад його синтаксису показаний на малюнку 2.7, який використовується для створення веб-сторінок. Створення шаблонів-це зручний підхід до програмування, при якому дані та макет, що використовуються для представлення даних, відокремлюються один від одного. Розділення цієї логіки означає, що дані та макет можуть бути змінені, не впливаючи один на одного. Це дозволяє зробити ваш код більш гнучким і придатним для

повторного використання. Handlebar макет-це структура HTML, яка визначає спосіб відображення розділів веб-сторінки. Макети створюються за допомогою заповнювачів даних. Дані та макети об'єднуються за допомогою шаблонізатора, який відображає інформацію в даних із заповнювачами, призначеними для відображення інформації.

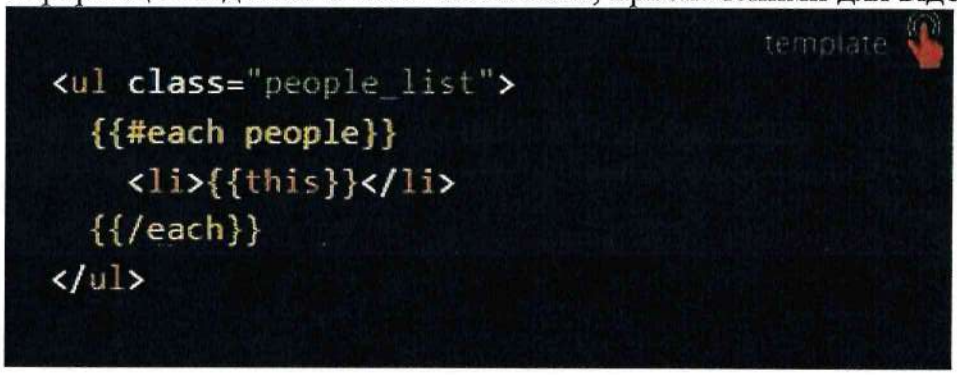


Рисунок 2.7 – Приклад синтаксису Handlebars

2. Jade – також мова шаблонів як і handlebars, проте з різним синтаксисом. Перша версія була розроблена у 2010 році. Для того щоб, наприклад написати тег … (як у HTML) достатньо всього ul, що видно на рис. 2.8.

```
ul#books
  li
    a(href="#book-a") Book A
  li
    a(href="#book-b") Book B
```

Рисунок 2.8 – Приклад синтаксису Jade

3. Pug – це перейменована МШ Jade, яка має той самий синтаксис, у 2015 році була випущена перша версія. Хоча Jade старіша ніж Pug, проте все ще популярна (хоча вони схожі).

```
mixin list(id, ...items)
  ul(id=id)
    each item in items
      li= item

+list('my-list', 1, 2, 3, 4)
```

Рисунок 2.9 – Приклад синтаксису Pug

4. Також існує Nunjucks. Це МШ створена у 2012 році. Синтаксис якої наведено на рис. 2.9, що віддалено схожий на JavaScript. Для написання коду у HTML розмітку потрібно використовувати тег "{%" - що схоже на теги у EJS.

```
<h1>Posts</h1>
<ul>
{% for item in items %}
  <li>{{ item.title }}</li>
{% else %}
  <li>This would display if the 'item' collection were empty</li>
{% endfor %}
</ul>
```

Рисунок 2.9 – Приклад синтаксису Nunjucks

Усі ці мови шаблонів використовують свій власний синтаксис, окрім EJS (Embedded JavaScript), котрий використовує JavaScript, що наведено на рис. 2.10. Через це було вибрано дану МШ. Також має 10,483,514 скачуваня за місяць.

```
<% if (query) {%>
<h1 class="d-flex justify-content-center">Search results for "<%=query%> "</h1>
<%} else {  %>
<h1 class="d-flex justify-content-center"> All Landmarks</h1>
<% } %>
```

Рисунок 2.10 – Приклад синтаксису EJS

Основні теги EJS, за допомогою яких можна писати код JavaScript в HTML розмітці:

- <% (тег для керування потоком коду, але без виводу (без відображання у HTML розмітки)).
- <%_ (прибирає усі пробіли перед ним).
- <%= (те саме що й тег <% але з виводом усього що знаходиться всередині даного тегу, але, якщо є HTML код , то виводиться як текст).
- <%- (аналог <%= але може виводити також HTML розмітку (тобто усі теги сприймаються)).
- <%# (для коментування).
- <%% (для виводу '<%').
- %> (кінцевий тег для <%).
- -%> (кінцевий тег для <%-).
- _%> (кінцевий тег для <%_).

2.6 Вибір картографічного сервісу

Звичайно можна обрати google maps як картографічний сервіс, але вньому при створенні та налаштовуванні міток данні зберігаються саме на серверах google. А нам потрібно було хранити інформацію в БД MongoDB.

Було обрано картографічний сервіс mapbox, через можливість зберігати данні у своїй БД та через більшу кількість безкоштовних запитів на карту (mapbox – 50000, google maps – 28000).

Monthly active users	Cost per 1000
up to 28 000 loads/ month	Free
0-100 000/month	$7
100 001 -500 000	$5,6

Рисунок 2.11– Порівняльний план безкоштовної кількості запитів на карту у mapbox та google maps

Mapbox була заснована у 2010 році, і є до сих пір дуже популярною. Такі великі компанії як: Adobe, IBM, Porsche, CNN, The New York Times, BMW, Xiaomi використовують даний сервіс.

2.7 Вибір фрейморвку для шаблонної стилізації

У цій статті ми вирішили використовувати фреймворк bootstrap, щоб заощадити час при створенні дизайну сервісів.

Bootstrap-це величезна колекція зручних для повторного використання фрагментів коду, написаних на HTML, CSS та JavaScript. Це також фреймворк для розробки інтерфейсу, який дозволяє розробникам та дизайнерам швидко створювати повністю адаптивні веб-сайти.

Насправді, bootstrap приділяє багато часу розробці веб-сторінок, що позбавляє від необхідності писати багато CSS-коду. Він також безкоштовний. В даний час він розміщений на GitHub і його можна легко завантажити з офіційного веб-сайту.

Він був створений у 2011 році колишніми співробітниками Twitter Марком Отто та Джейкобом Торнтоном.

Його переваги:

- Адаптивний спосіб розміщення контенту. Більше не потрібно витрачати години на кодування сітки - Bootstrap має власну попередньо визначену систему сіток. Зразу можна приступити безпосередньо до наповнення контейнерів вмістом. Також для адаптивності для різних величин екрану не потрібно в css файлі писати код під кожний розмір, можна просто у потрібних об'єктах прописати такі класи як: extra small (для екранів менше 576px(пікселів)), small (>=576px), medium (>=768px), large

(>=992px), extra large (>=1200px), extra extra large (>=1400px). Для цих класів вже прописана логіка.

- Адаптивне зображення. Bootstrap має власний код для автоматичного зміни розміру зображення залежно від поточного розміру екрана. Ви можете просто додати клас до зображення.img-відповідь, яка автоматично змінює ширину та довжину фотографії. Крім того, використовуючи такі класи, як img-circle та img-rounded, Bootstrap може швидко змінити форму зображення.
- Різні великі компоненти прописуються заздалегідь. Bootstrap постачається з написаним кодом, який ви можете легко прикріпити до веб-сторінки, наприклад, панель навігації, спадне меню, панель прогресу та ескіз. Ви також можете переконатися, що кожен компонент виглядає краще, незалежно від розміру екрана чи пристрою, на якому він відображається. Багато готових функцій у вас під рукою.
- Готова анаміція (Java Script). Bootstrap також дозволяє розробникамскористатися перевагами понад дюжини користувацьких плагінів JQuery (набір функцій JS). Ця бібліотека надає більше можливостей для маніпулювання з інтерактивністю, пропонуючи прості рішеннядля спливаючих вікон, переходів, каруселей зображень та інше.

Також до переваг відноситься проста і зрозуміла документація. Кожна частина коду детально описана та пояснена на їх веб-сайті. Опис також включає приклади коду для базової реалізації, що спрощує процес навіть для початківців. Все, що вам потрібно зробити, це вибрати компонент, скопіювати код, вставити його на сторінку та встановити звідти. Обладнання. 1. Одним з головних недоліків таких фреймворків, як Bootstrap, є те, що для правильної роботи всіх класів необхідно імпортувати файл або залишити посилання.

Завантаження цих файлів на першій сторінці може бути не таким швидким, як хотілося б. Наприклад, поточна версія файлу Bootstrap css становить 119 КБАЙТ. Це може здатися не дуже великим розміром у порівнянні з зображенням або відеофайлом, але для CSS-файлів це величезний обсяг. Однак у Bootstrap є рішення цієї проблеми. Просто перейдіть на сторінку "налаштування та завантаження", щоб позначити функції, які не потрібні програмі, зменшити вагу файлу та заощадити користувачеві додатковий час на завантаження.

Його спільнота. Як і багато інших проектів з відкритим кодом, за Bootstrap стоїть ряд дизайнерів та розробників. Хостинг на GitHub дозволяє розробникам легко додавати що-небудь до своєї кодової бази. Це також полегшує спільну роботу користувачів. У Bootstrap є ще одне місце для активної сторінки Twitter, блогу Bootstrap і навіть Slack (корпоративного месенджера). Крім того, це не стосується розробників, які хочуть допомогти з технічними проблемами Stack Overflow, коли всі питання знаходяться під тегом bootstrap-5.

Зовнішній шаблон. У міру зростання популярності фреймворку люди все частіше використовують шаблони на основі bootstrap для подальшого прискорення процесу веб-розробки.Існує також багато веб-сайтів, які спеціалізуються на заміні та придбанні користувацьких шаблонів на основі цього фреймворку[11].

2.8 Спосіб збереження паролю користувача

По-перше, не зберігайте в базі даних своє ім'я користувача (або адресу електронної пошти) та пароль у текстовому форматі. Оскільки, коли хтось отримує доступ до бази даних, він може використовувати цю інформацію на свою користь. У додатках люди часто використовують один і той же пароль для більшості програм і веб-сайтів (включаючи банки). Отже, це найгірший спосіб зберігання таких даних користувачів.

Є краще рішення-хешування. Наприклад, ви можете не тільки зберігати паролі в текстовому форматі, але і змішувати їх, обробляти за допомогою хеш-функцій і зберігати в базі даних. Хеш-функція-це функція, яка приймає дані будь-якої довжини для введення та виведення хеш-даних (змінених якимось чином) заданої довжини.

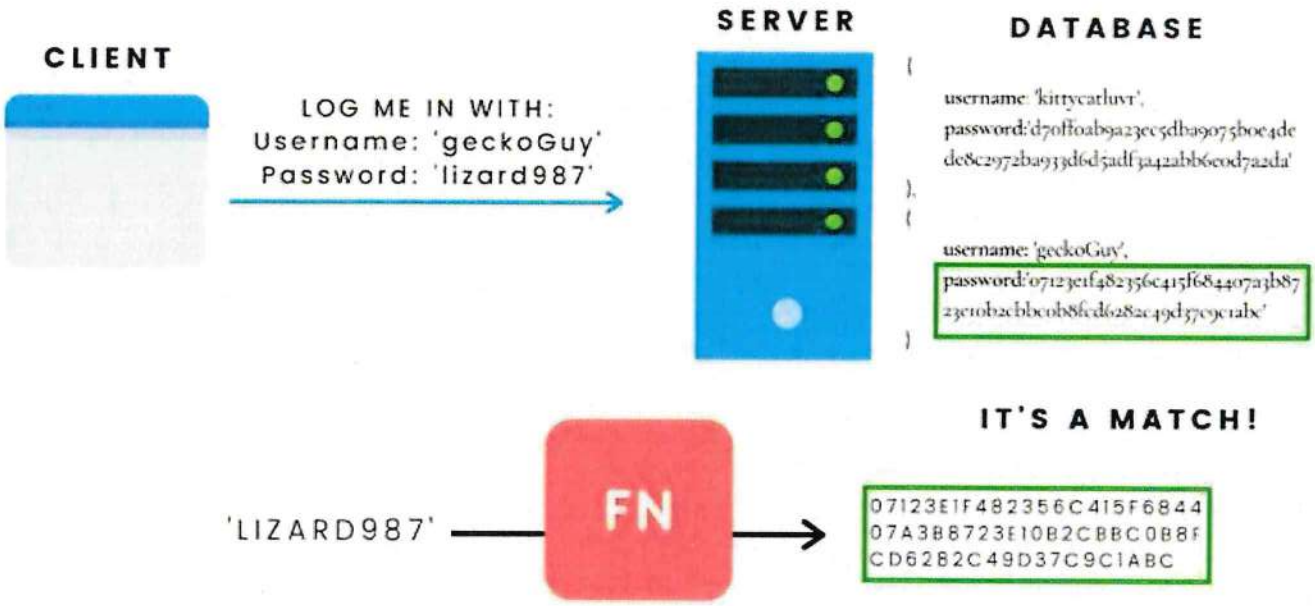


Рисунок 2.12 – Приклад роботи хеш функції

Суть хешу полягає в тому, що коли хакер отримує доступ до бази даних і бачить хешований пароль, він не розуміє, що це таке, і не може використовувати його в своїх цілях.На малюнку 2.12 видно, що "07123e1f482356c4..."- це не те саме, що LIZARD987.

Але з цими хеш–функціями є одна проблема-єдиною вимогою є перетворення вхідних даних будь-якої довжини в дані фіксованої довжини. Якщо ця функція проста, ви можете легко знайти оригінальну інформацію.

Наприклад, на малюнку 2.13 показана проста хеш-функція лише з 16 значеннями хешу, тому хакеру, який знає хешований пароль (наприклад, 03), не важко отримати найпоширеніший пароль і хешувати його, поки він не отримає хеш-значення 03.

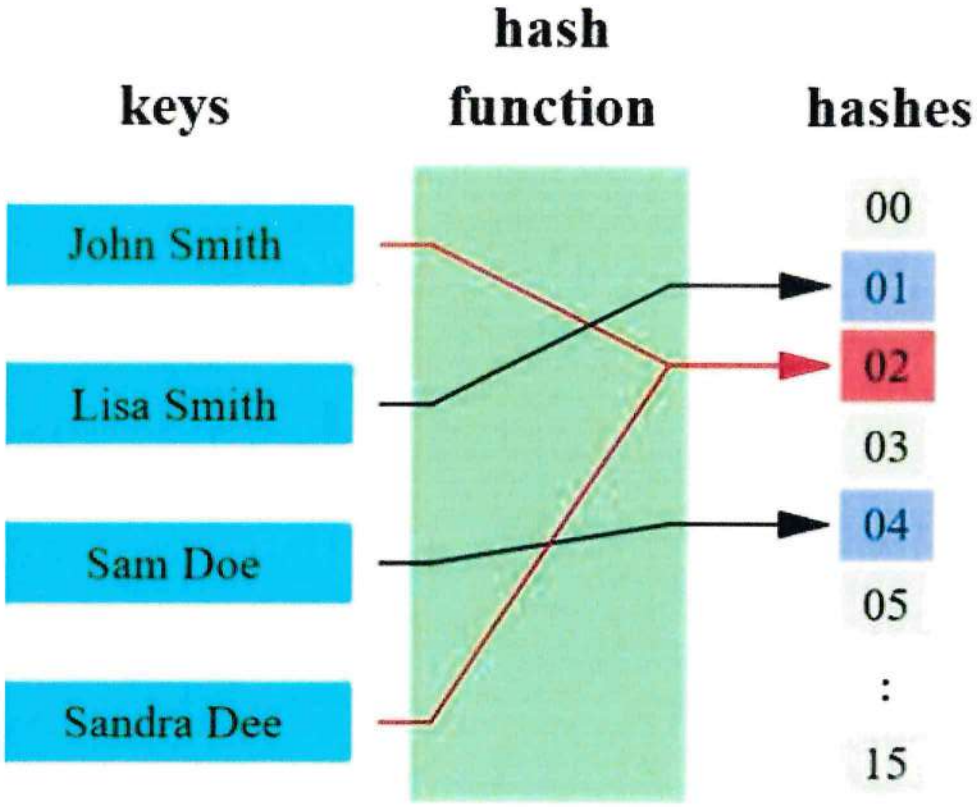


Рисунок 2.13 – Приклад дуже простої хеш-функції

Є криптографічні функції хешування (які мають більше вимог порівняно з не криптографічними функціями). Вони повинні:

1. Бути необоротними та односторонніми – тобто щоб не можна булоніяким чином подивившись на хешовану інформацію перетворити її на початкову.
2. Бути такими, щоб маленька зміна у початкових даних призводила до величезних змін у хешованих. З рис. 2.14 видно, що зміна всьогоодного символу у початкових даних приводить до повністю іншогозначення після хешування.
3. Детермінізм-незалежно від того, скільки разів ви виконували цю операцію після обробки вхідних даних за допомогою зашифрованої хеш-функції, вихідні дані повинні бути однаковими..
4. Імовірність колізій даних дуже мала, тобто хешований висновок різних даних може бути різним (звичайно, цього не можна зробити, але такі шанси повинні бути мінімальними).

5. Це може здатися дивним, незалежно від того, наскільки повільно працює функція. Чому це так? Тому що, якщо функція працює швидко, хакери можуть підмінити мільярди паролів і швидко отримати бажане значення хешу. Однак, якщо функція працює повільно, пошук потрібного значення займе набагато більше часу.

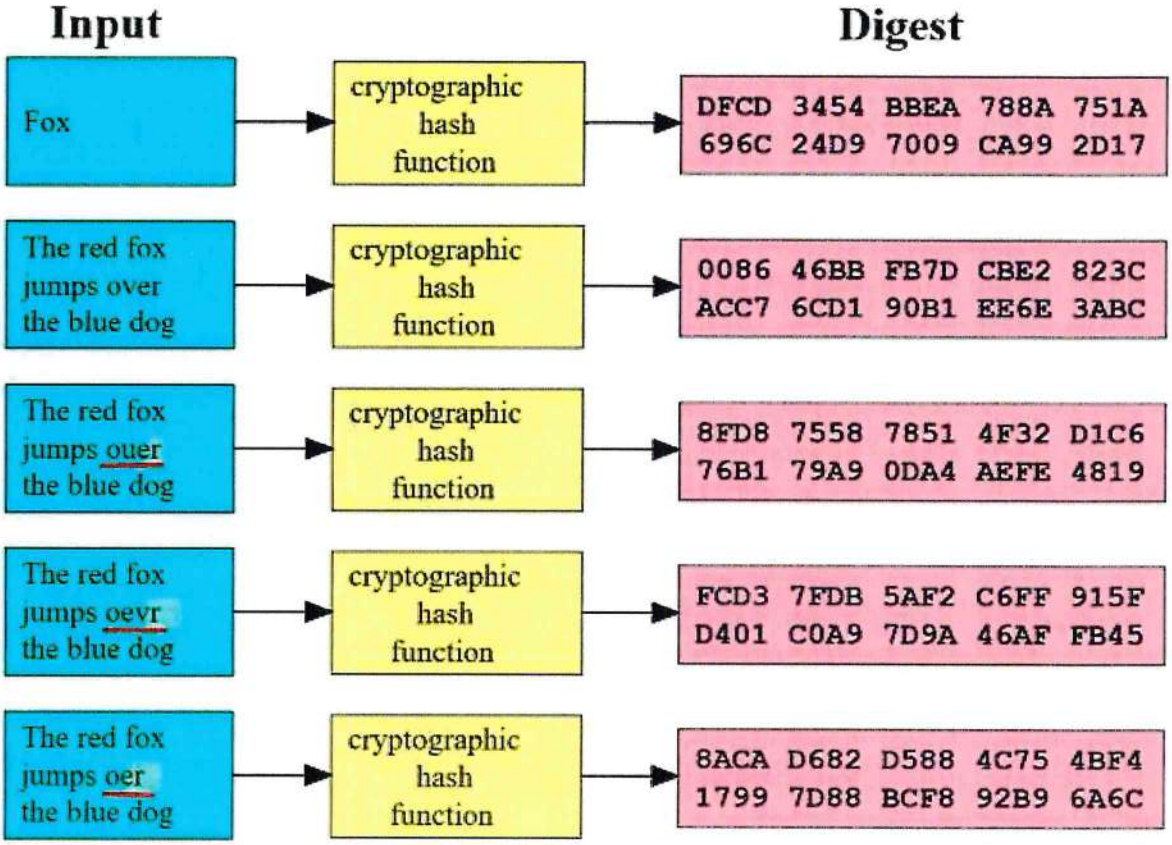


Рисунок 2.14 – Приклад зміни тільки одного символу у вхідних даних

Найкращий спосіб зберегти паролі для останніх користувачів - це Використовувати зашифровану хеш-функцію з додаванням солі (для цього завдання є такий метод). Сіль у цьому випадку-це, наприклад, випадковий набір символів, який додається до пароля перед його хешуванням. І оскільки навіть 1 символ вхідних даних повністю змінює вихідні дані (після використання зашифрованої хеш-функції), хакерам потрібно набагато більше часу та ресурсів, щоб знайти та використовувати вихідні дані.

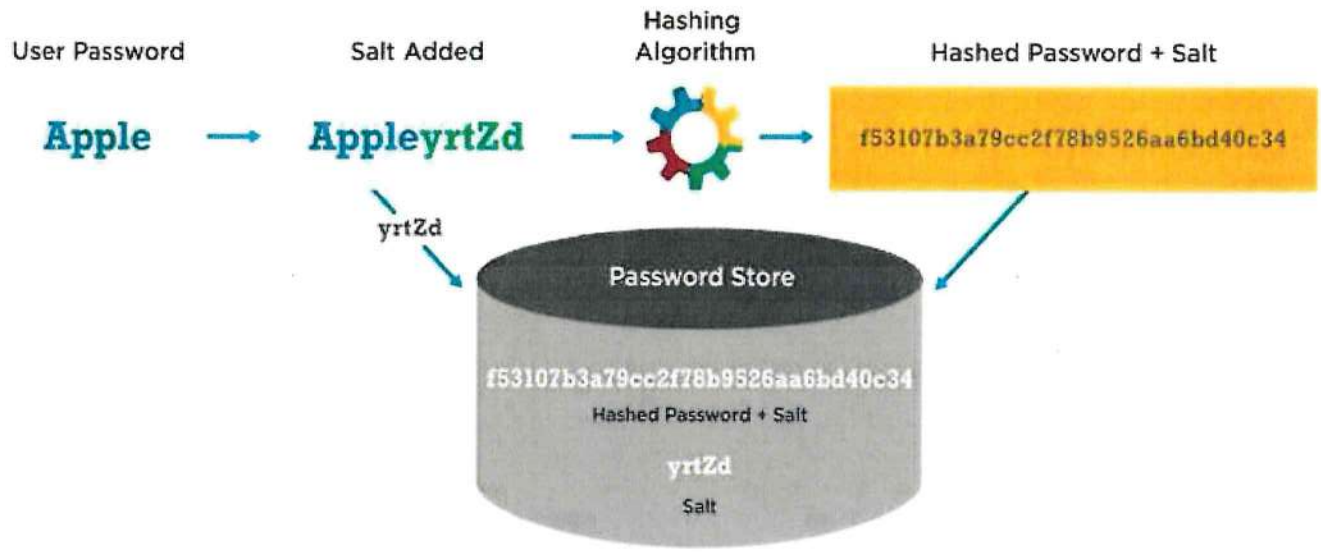


Рисунок 2.15 – Приклад додавання "солі" у вхідні данні перед їх хешуванням

Навіть користувачі, що мають однаковий пароль, наприклад "p4s5w3rdz" будуть мати зовсім інші хешовані данні на сервері через додавання "солі". Це видно на рис. 2.16.

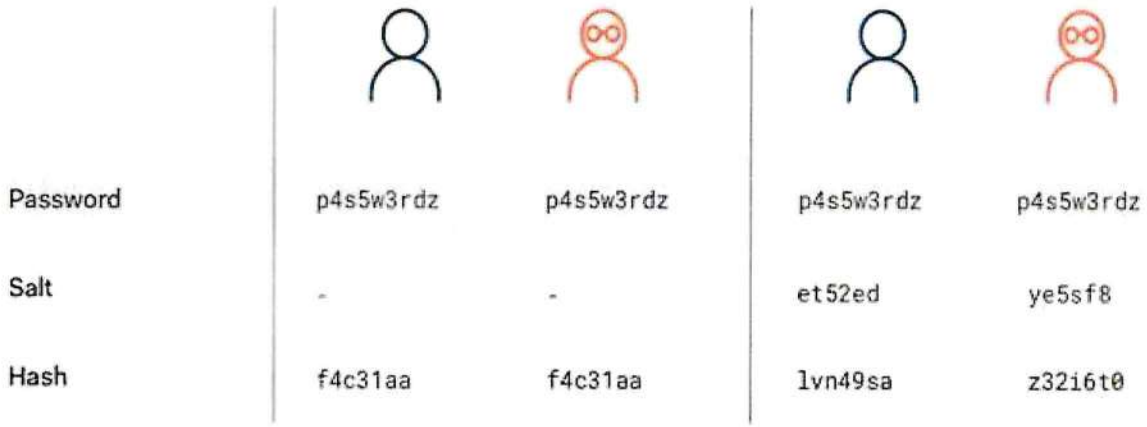


Рисунок 2.16 – Приклад використання однакового паролю різними користувачами

У даній роботі використовується проміжне програмне забезпечення Passport, що має 1,496,503 скачування за неділю, яке звичайно використовує криптографічну хеш функцію Pbkdf2.

2.9 Сесія

Як можливо перейти за різними посиланнями в одній веб-службі і дозволити браузеру зрозуміти, що робить той самий користувач, тобто як залишатися в системі (використовується в цій роботі)? Сесія-це група взаємодій Користувача з веб-сайтом, які відбуваються протягом певного періоду часу.

Особливістю веб-сервера є те, що протокол HTTP не здатний відстежувати ці стани і підтримувати зв'язок з клієнтом, тому кожен новий URL-запит, що надходить з іншого браузера, обробляється окремо. Для вирішення цієї проблеми існує browser session (сесія браузера) – механізм, який дозволяє відстежувати запити з 1 браузера і зберігати деякі змінні при переході по різних сторінках сайту.

На початку сеансу на стороні сервера створюється файл, що містить інформацію про Користувача, події, що відбулися в сеансі, і діях (1). До таких подій відносяться твердження, відвідування сайту, різні взаємодії користувача з елементами сторінки і додавання товарів в корзину. Тож використовуйте цей файл.Також, поки попередня сесія активна, нова не може бути розпочатою.

Старий сеанс може завершитися при дотриманні однієї з умов:

- при виходу з браузера
- після закінчення певного часу бездіяльності (timeout)
- в певний час доби (опівночі, наприклад)

ВИСНОВОК ДО РОЗДІЛУ 2

Отже в цьому розділі був зроблений огляд вимог, яким повинен відповідати застосунок. Також було проведено порівняльний аналіз та проаналізовано технології, інструменти та бібліотеки для розробки цього дослідження.

Розвиваючи тему дипломної роботи, після розгляду відповідних інструментів для вирішення завдання, на основі прийнятих рішень в якості мови програмування був обраний JavaScipt. Платформа розробки-node.js. Фреймворк-ExpressJS. В якості картографічного сервісу обраний Mapbox, який не зберігає персональні дані. Для зберігання зображень вибрано хмарне сховище Cloudinary, а решта інформації знаходиться в базі даних неструктурованих типів мови запитів MongoDB. Node.js та MongoDB взаємодіють між собою за допомогою бібліотеки Mongoose.

Для мов шаблонів вибрано EJS, щоб зменшити обсяг коду. Платформа Bootstrap також була обрана в якості стилю шаблону для оформлення сайту.

Також аналізується метод зберігання паролів, найкращим з яких є використання криптографічних хеш-функцій (доступних у проміжному програмному забезпеченні Passport).

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Огляд структур проекту

Система розроблялася у зручному середовищі розробки Visual Studio Code, розробленому компанією Microsoft. На рис. 3.1 представлена архітектурапроекту.

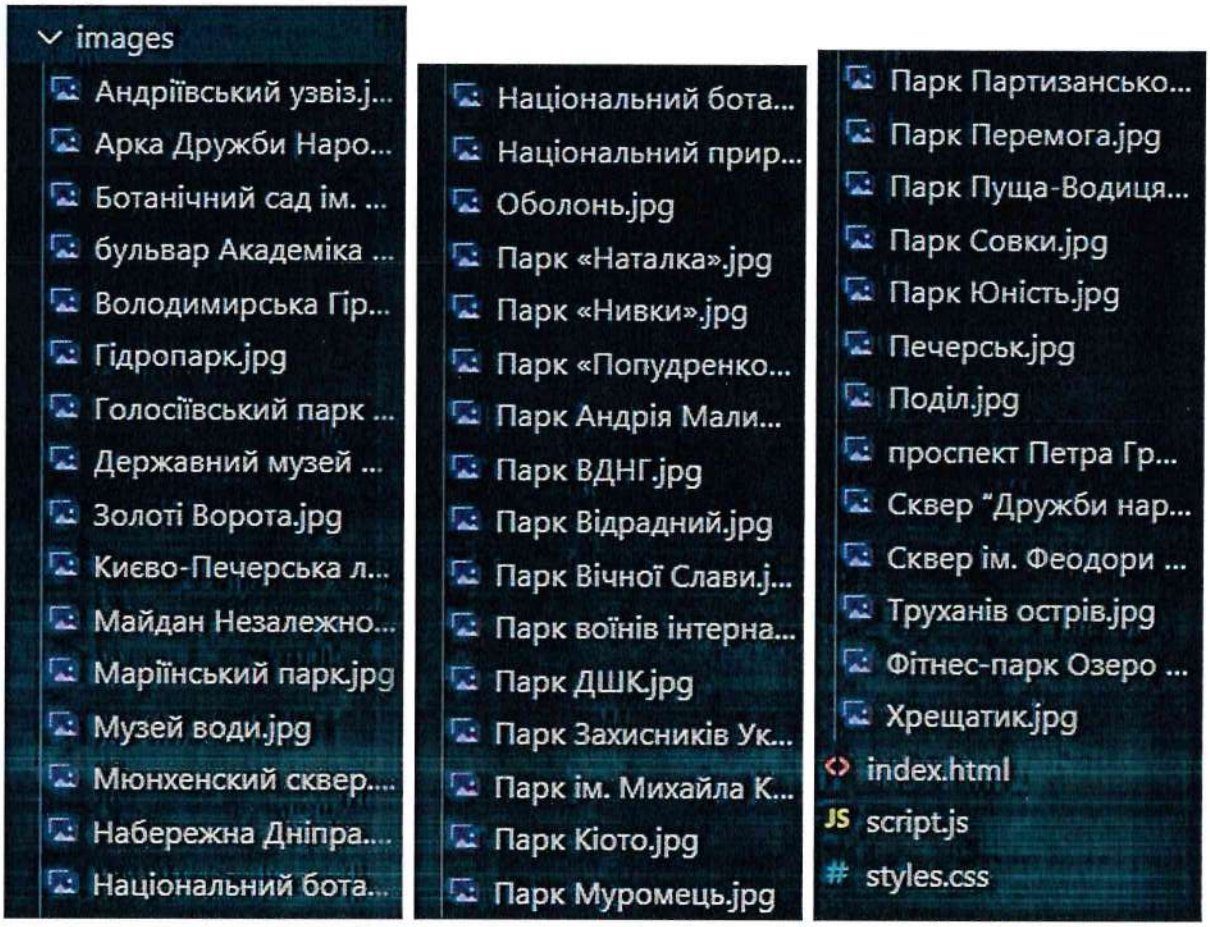


Рисунок 3.1 – Архітектура проекту

- Каталоги з файлами: cloudinary, controllers, models, node_modules,public, routes, utils, landmarks, layouts, partials, users описані нижчеу таблицях 3.1, 3.2 та 3.3.
- error.ejs – EJS сторінка, в якій описується структура виводувиникаючої помилки, за наявності.

- home.ejs - EJS сторінка, де є опис першої сторінки сервісу, на якійє привітання, кнопки для реєстрації, авторизації, переходу до сторінки з відображенням усіх визначних місць з великою інтерактивною картою.
- .env – файл, в якому містяться токен картографічного сервісу Mapbox,що був автоматично згенерований під час реєстрації, також є ім'я, ключ так секрет від аккаунта БД Cloudinary (усі ціданні використовуються для з'єднання сервісів).
- app.js – основний скрипт для взаємодії з проектом, який запускається командою >node app.js. або >nodemon app.js. В ньомусервіс підключається до порту 3000 (стандартний порт), а БД MongoDB до порту 27017. Імпортуються усі основні бібліотеки тапакети. Підключається фреймворк express.js для роботи з сервісною частиною сайту. Описується конфігурація сесії.
- middleware.js – скрипт для опрацювання проміжних дій, а саме – перевірка того, чи користувач авторизувався, чи є автором допису,або коментарю, чи правильно було створено визначне місце, або коментар.
- package-lock.json – файл, в якому міститься детальна інформаціяпро усі встановлені пакети.
- package.json – файл який було створено з самого початку розробкисистеми, має інформацію про проект та версії усіх використовуваних модулів, фреймворків та баз даних.
- schemas.js – скрипт, в якому прописані перевірки схем БД, такихяк створення визначного місця та відгука для нього.

Використано в архітектурі:

- Index.html - дозволяє добавити елементи катри та футер на сайт
- Script.js – скрипти для відкриття міток та допису до них
- styles.css – задає стиль тексту, колір, шрифт та т.д.
- images – папка з фото котрі використовуються в карті

3.2 Файли JS

В табл. 3.1 наведені скриптові Java Script файли, в яких розписана логіка системи. А саме: створення, змінення, видалення та пошук дописів, створення та видалення коментарів, реєстрація, авторизація та вихід з облікового запису користувача, моделі бази даних для дописів, комєнтарів та користувачів, коректні url-запити, а також імпорт БД cloudiary, обробка можливих помилок та опис характеристик карт.

Таблиця 3.1 – JS-файли

№ з/п	Назва файлу	Призначення та функції
1	cloudinary/index.js	скрипт, в якому імпортуються данні для підключення та роботи з БД Cloudinary
2	controllers/landmarks.js	скрипт, де розписана логіка пошуку, створення, відображення, редагування, відновлення та видалення дописів та нумерування їх можливої кількості на одній сторінці
3	controllers/reviews.js	скрипт, в якому описана логіка створення та видалення коментарів під дописами
4	controllers/users.js	скрипт, де описується логіка для реєстрування нового користувача, авторизація, та вихід зі свого аккаунту
5	models/landmark.js	скрипт, в якому є опис схеми зображення, які імпортують користувачі та схема відображення дописів на головній сторінці та на інтерактивній карті

Продовження таблиці 3.1

№ з/п	Назва файлу	Призначення та функції
6	models/review.js	скрипт, де описано схему коментарів та оцінки дописів користувачами
7	models/user.js	скрипт, де описано схему даних про користувача
8	public/javascripts/ clusterMap.js	скрипт, в якому описані характеристики карти, такі як: стиль, масштаб, центр (відображається на головній сторінці), а також логіка створення міток, опис їх зовнішнього виду, дії при наведенні, натисненні, відведення курсором.
9	public/javascripts/ showPageMap.js	скрипт, де прописані характеристики карти та міток, що відображаються на сторінці, проглядання повної інформації про допис
10	public/javascripts/ validateForms.js	скрипт для з'єднання стилів валідації Bootstrap з сервісом
11	routes/landmark.js	скрипт, в якому описано різні типи запитів, такі як 'get', 'post, 'put', 'delete' для відповідно для відображання, створення, редагування та видалення дописів про визначні місця
12	routes/reviews.js	скрипт, де прописано різні типи запитів для створення та видалення коментарів під дописами
13	routes/users.js	скрипт, в якому прописані різні типи запитів для реєстрування, авторизації та виходу з аккаунту для користувачів
14	utils/cathAsync.js	скрипт для обробки помилок асинхронних функцій

Продовження таблиці 3.1

№ з/п	Назва файлу	Призначення та функції
15	utils/ExpressError.js	скрипт для розширення можливостей виводу помилки, за наявності

В табл. 3.2 наведені Embedded JavaScript файли, в яких розписані сторінки проекту, для: створення та редагування дописів, перегляду карт, помилкових дій, опису навігаційної панелі, нижньої частини та нумерації сторінки, імпорту фреймворку Bootstra, а також реєстрації та авторизації користувача. Тобто це файли з HTML розміткою та Java Script кодом.

Таблиця 3.2 – EJS-файли

№ з/п	Назва файлу	Призначення та функції
1	views/landmarks/ edit.ejs	сторінка для редагування дописів про визначне місце
2	views/landmarks/ index.ejs	головна сторінка для перегляду великої карти, та неповної інформації про дописи, що на ній розміщені у якості міток
3	views/landmarks/ new.ejs	сторінка для створення нових дописів
4	views/landmarks/ show.ejs	сторінка для перегляду малої карти та повної інформації про дописи, а також коментарів та оцінки користувачів
5	views/layouts/ boilerplate.ejs	шаблонна сторінка, у якій є імпорт даних для коректного відображання фрейморку Bootstrap

Продовження таблиці 3.2

№ з/п	Назва файлу	Призначення та функції
6	views/partials/flash.ejs	сторінка для опису вигляду відображення коректної, або помилкової дії.
7	views/partials/footer.ejs	сторінка, в якій описується зовнішній вигляд footer
8	views/partials/ navbar.ejs	сторінка, в якій описується зовнішній вигляд header
9	views/partials/ pagination.ejs	сторінка з описом нумерації
10	views/users/ login.ejs	сторінка для авторизації
11	views/users/ register.ejs	сторінка для реєстрації

В табл. 3.3 наведені Cascading Style Sheets файли для опису стилей сторінок проекту.

Таблиця 3.3 – CSS-файли

№ з/п	Назва файлу	Призначення та функції
1	public/stylesheets/ app.css	сторінка, в якій описується використовувані стилі для головної сторінки, карт, посилань, зображень
2	public/stylesheets/ home.css	сторінка для опису стилей для файлу home.ejs
3	public/stylesheets/ stars.css	сторінка, де наведені стилі для рейтингу коментарів

3.3 Способи використання системи

Можливості для користувача:

- Переглядати сторінку-карту.
- Відкривати мітки для перегляду інформації щодо місць.
- Переглядати головну сторінку з дописами, що мають короткий опис та характеристику.
- Робити пошук дописів або місцезнаходженням по карті власноруч.
- Маніпулювати інтерактивною картою міста Київ.
- Виходити з облікового запису.
- Переглядати головну сторінку з дописами, що мають короткий описта характеристику.
- Робити пошук дописів за назвою або місцезнаходженням.
- Створювати новий допис, за зворотрім зв'язком, який залишений знизу на футер
- Переглядати головну сторінку з дописами, що мають короткий описта характеристику.
- Переглядати останній відкритий опис в низу сторінки.
- Маніпулювати великою інтерактивною картою.
- Змінювати перегляд мітки з одної на іншу

ВИСНОВОК ДО РОЗДІЛУ 3

Отже, в цьому розділі представлена архітектура системи, і на основі цих файлів можна легко пояснити всі файли, використані для розробки цього проекту, і повністю проаналізувати, як працює ця система. Також були передбачені всі можливі способи використання системи неавторизованими користувачами, які є / не є автором.

Використовуючи інформацію в цьому розділі, можна зробити висновок, що наданих функцій достатньо для вирішення завдань, тобто методів, що використовуються в інтерактивній карті.

РОЗДІЛ 4
ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Вимоги для запуску

Перед запуском потрібна наявність програмного середовища Visual Studio Code останньої версії та браузер Opera GX 84.0.4316.36 версії. Для запуску треба відкрити папку з роботою та запустити Html файл, після чого у браузері відкриється сайт, на рис. 4.1 зображено потрібний файл для запуску

Рисунок 4.1 – Файл для запуску

4.2 Тестування

Тестування буде проводитися в браузері. При відкритті сайту бачимо сторінку з мапою, що зображена на рис.4.2. На ній є мітки для перегляду інформації, а також зворотній звязок

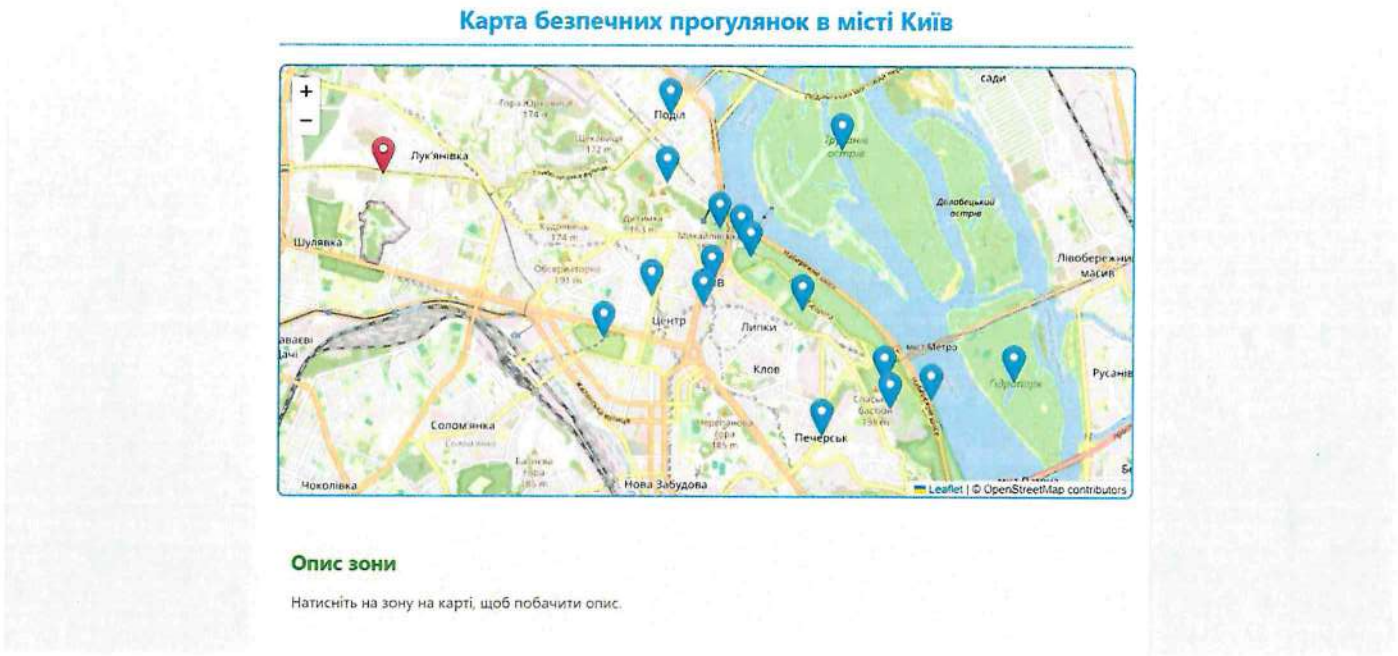


Рисунок 4.2 – Перша сторінка

Спочатку, перевіримо доступні функції для користувача, перейшовши на сторінку, що зображена на рис.4.3, та находиться за посиланням file:///C:/Users/Vlad/Desktop/Інтерактивна%20карта%20Києва/index.html за допомогою міток На даній сторінці відображається інтерактивна карта, на якій є близько 30 дописів. Під картою є ті самі дописи, які мають короткий опис, зображені на рис.4.6.

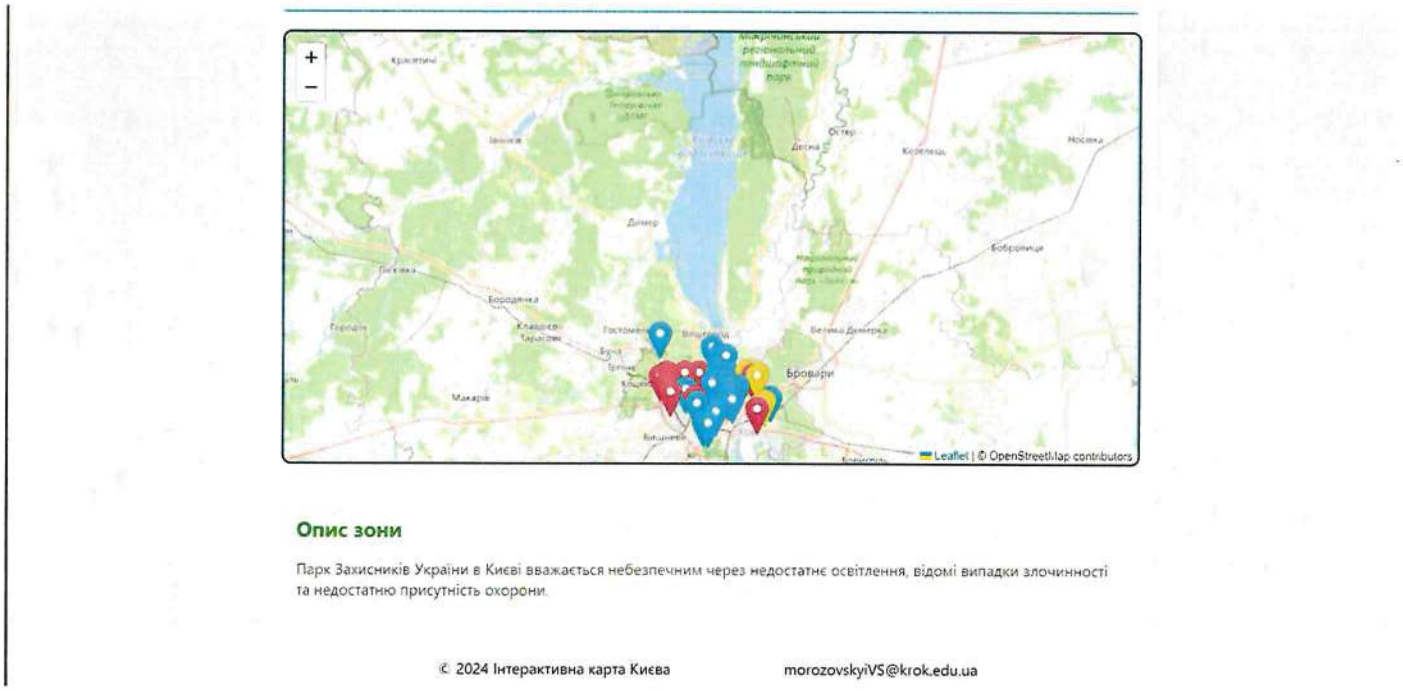


Рисунок 4.3 – Інтерфейс головної сторінки

При масштабуванні мітки будуть залишатися на своєму місці розташування. Також буде змінюватися колір відносно рівня безпеки на них(щозображено на рис. 4.4):

- червоний, при небезпечності = unsafe
- синій, коли безпечно >= safe
- жовтий, коли безпека на середньому рівні відносно інших міток>= less-safe

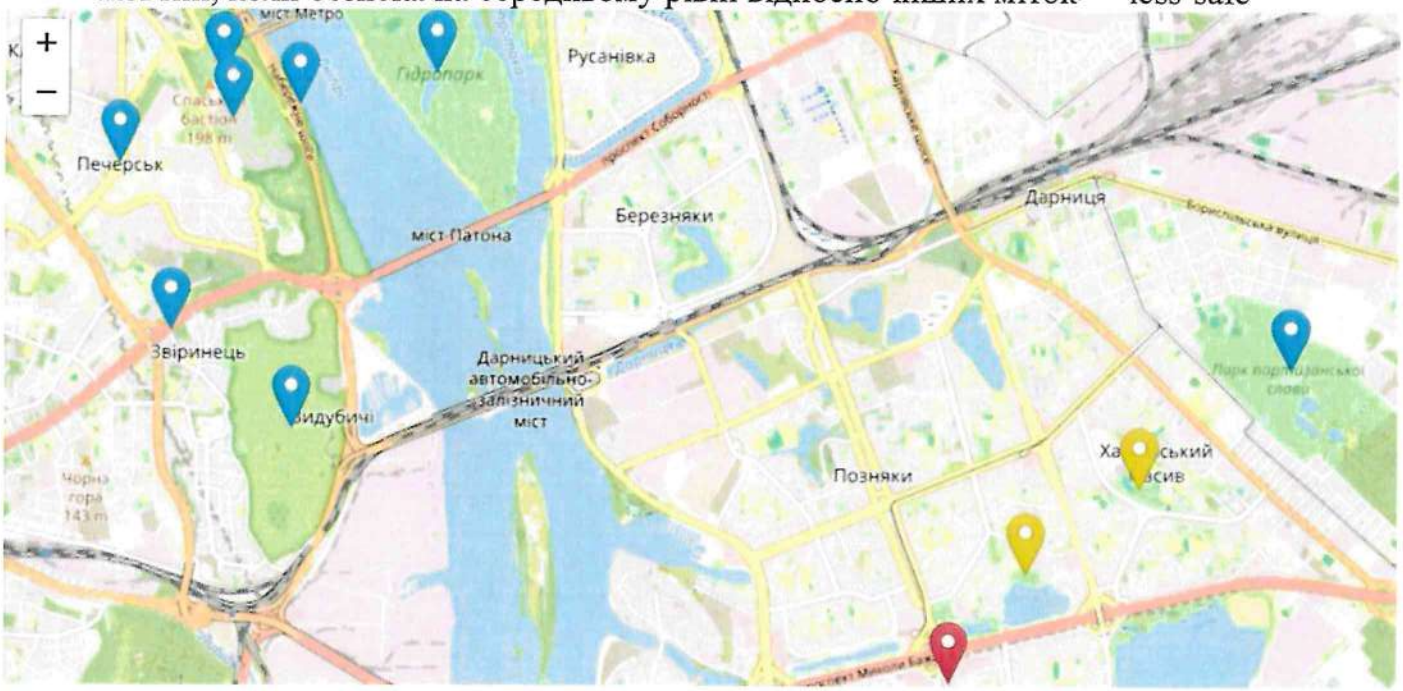


Рисунок 4.4 – Змінений масштаб карти

Тепер спробуємо натиснути на мітку. На рис.4.5 відображається контейнер з інформацією, в якому є назва та короткий опис місця.

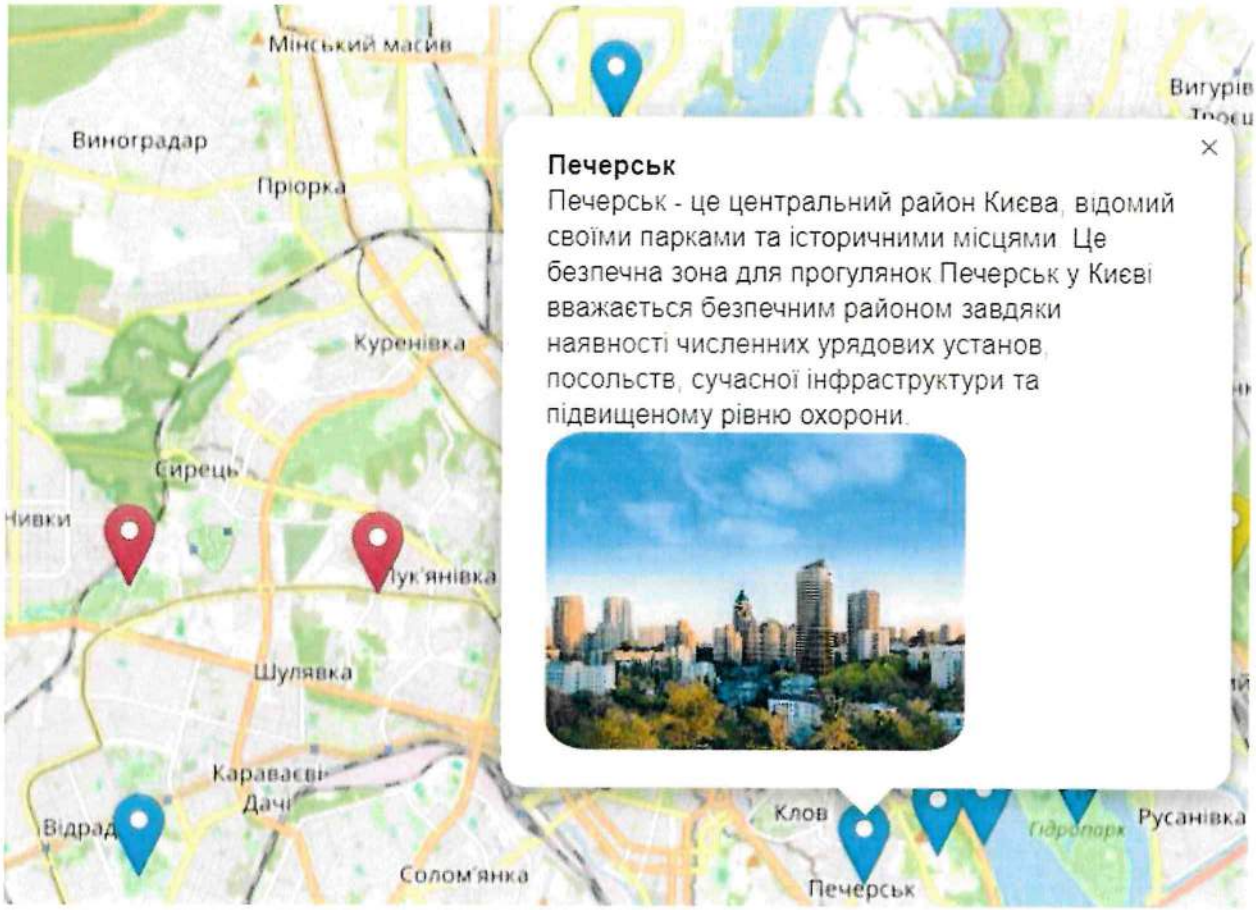


Рисунок 4.5 – Натиснена мітка

На сторінці, що зображена на рис. 4.5 є фото, назва допису, описання, місцезнаходження, та причина визначеного рівня безпеки. А також під картою є коментарі з оцінкою від інших користувачів.

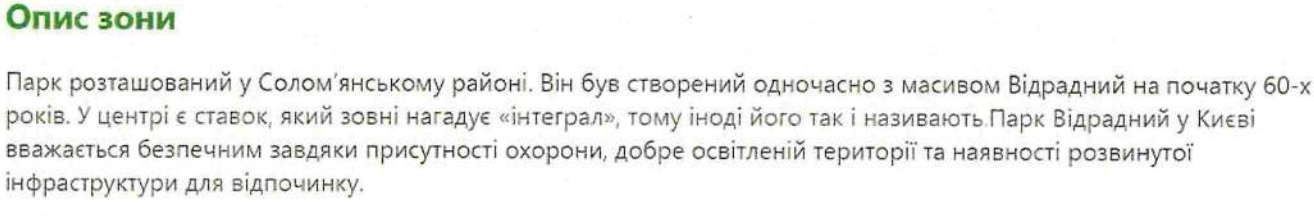


Рисунок 4.5 – Контейнер мітки з повною інформацією про допис

Тепер виконаємо огляд дописів, скролим вниз щоб переглянути. На рис. 4.6 можна побачити, що відображення спрацювало правильно та відображає опис останньої мітки..

Рисунок 4.6 – Можливі варіанти перегляду фото з різних міток

Усього було знайдено 8 дописів на рис. 4.6. Це означає, що на карті також відображається ця кількість дописів у вигляді міток, це можна побачити на рис.4.7.

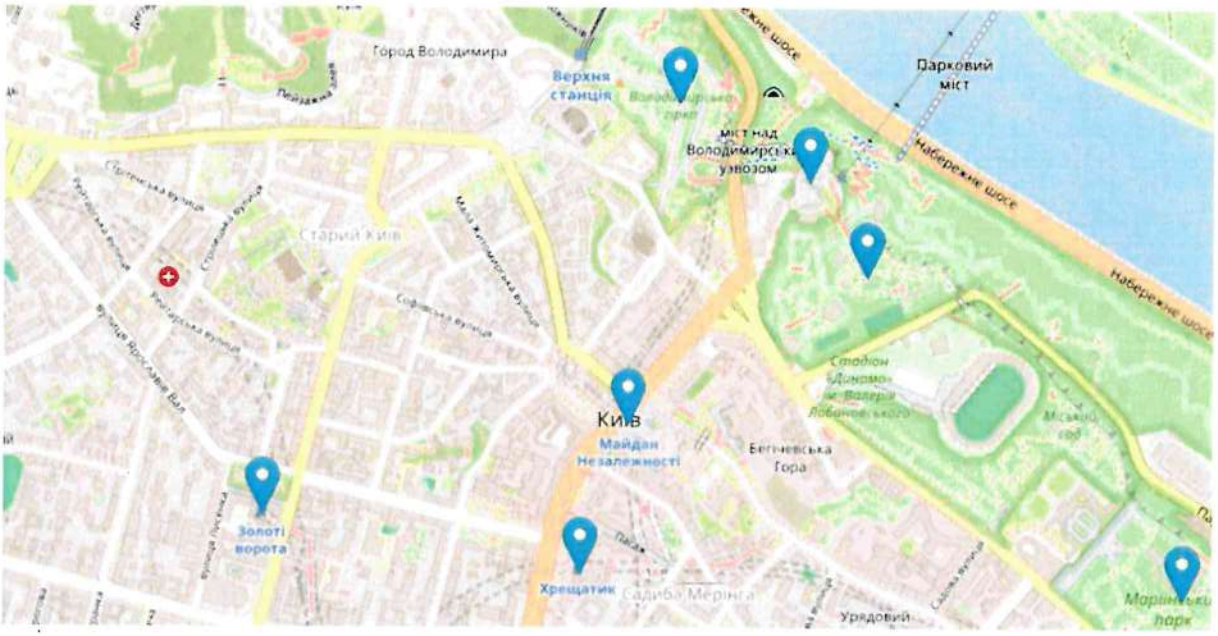


Рисунок 4.7 – Відображення міток на карті після маштабування(1)

Далі потрібно приблизити карту. Якщо це зробити, то помітно, що мітки не змінили свого розміру відносно карти, як на рис. 4.7. Зміна розміру міток відносно карти не має значення, так як площа використаної карти не велика. Користувач може віддаляти карту без обмежень, але потреби в цьому немає, тому мітки не регулюються та мають статичний розмір.

Рисунок 4.8 – Відображення футера знизу сторінки

Також потрібно переглянути можливість зворотнього звязку. З самого низу сторінки, є футер, котрий відображає назву сторінки, рік розробки, та контактну пошту рис. 4.8. Ідея зворотнього зв'язку край необхідна для цієї карти, користувачі можуть надавати розробнику актуальну інфомацію з щойно відвіданих місць, та звітувати чи дійсно місце було безпечним. Також за зворотнім зв'язком та ініціативою користувачів, можливо створювати нові мітки, на котрі може бути навіть більша потреба, ніж в базових. Також це дає можливість на розгляд нових місцевостей за проханням

```
.popup-image {
    border-radius: 10%;
}
```

Рисунок 4.9 – Зміна вигляду фото для більш привабливого вигляду

Проблема вигляду фото в popup сильно кидалась в очі, тому вона була виправлення зглажуванням кутів рис. 4.9.

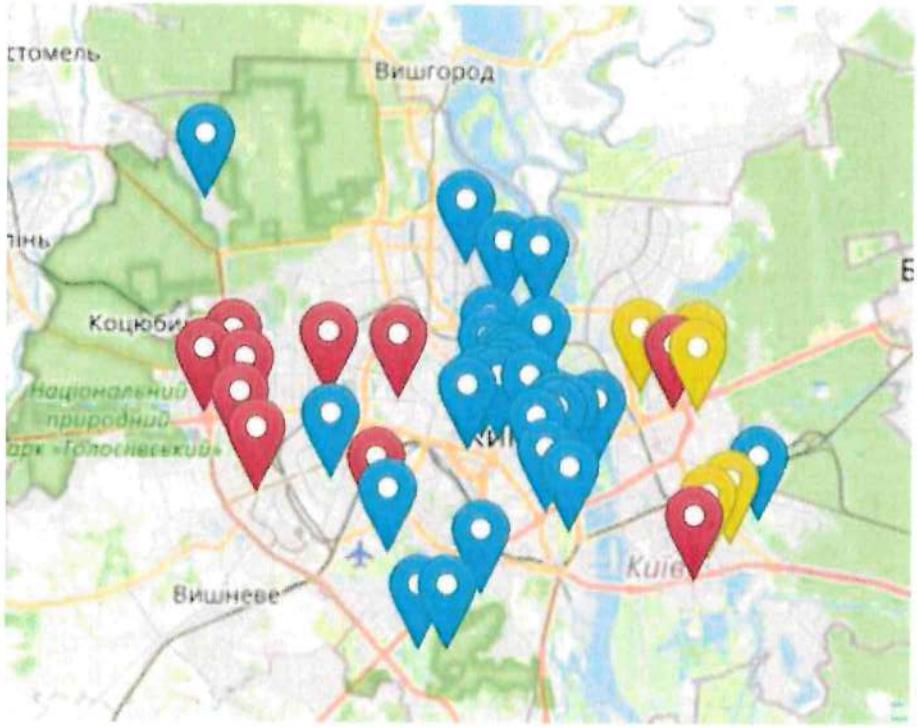


Рисунок 4.10 – Відображення міток на карті після маштабування(2)

Якщо ви віддалите карту з рис 4.7. то мітки візуально стануть більші, ніж були. Проте це дійсно якщо порівнювати з Києвом, від загального розміру карти мітки розмір не змінять. На рис 4.10 видно, що мітки заважають відкритті одна одній та вилазять на сусідні. Ця проблема не є актуальною через те, що причин віддаляти так сильно карту немає

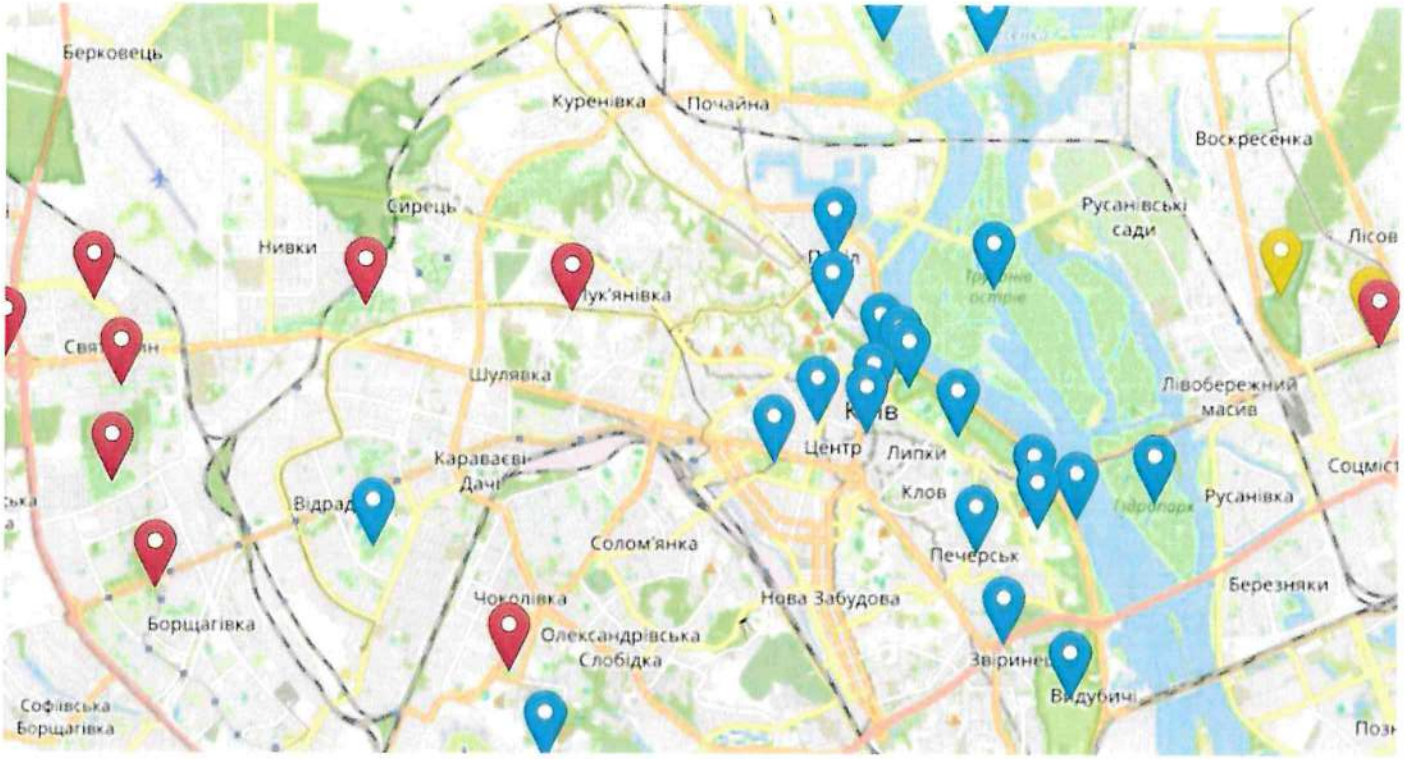


Рисунок 4.11 – Відміна PopUp

Закриємо popup з рис 4.5. Закриття відбувається досить зручно, для цього потрібно лиш нажати на хрестик в верхньому правому куті, або ж натиснути на будь-яке місце чи точку карти, де немає міток, та самого відкритого popup, див. рис. 4.11. Якщо при відкритому popup натиснути на іншу мітку, то закриється останній, а з мітки з'явиться новий. Також в контейнері з описом, опис зміниться

Рисунок 4.12 – Дані котрі виводяться в дописах

Для виведення тексту використовується JS, тому в коді з мітками вказаний і текст, котрий виводиться. Послідовність встановлення мітки:

- Задається ім’я
- Координати
- Текст
- Фото
- Рівень безпеки мітки котрий змінює колір

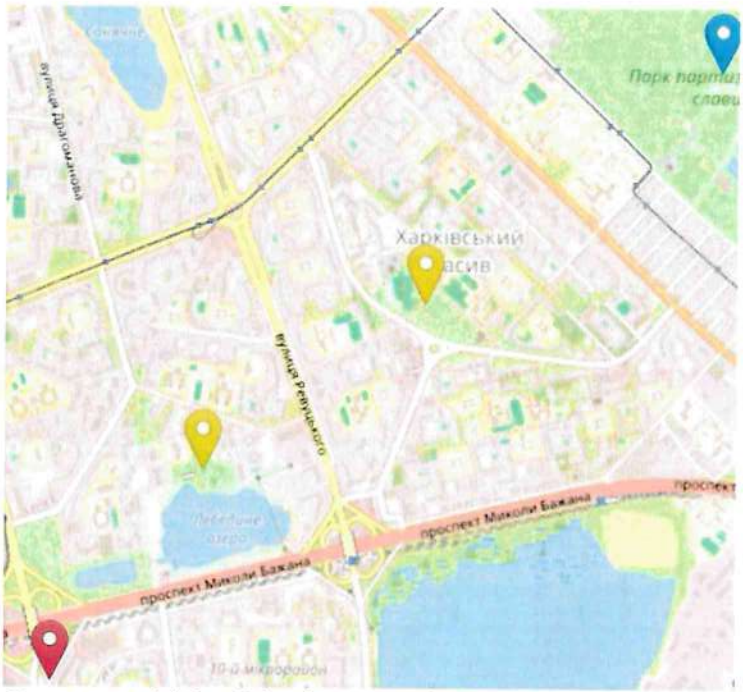


Рисунок 4.13 – Різні типи міток на карті

Зміна забарвлення міток залежить від значення безпеки. При безпечній – синя, напівбезпечній – жовта, небезпечній – червона. Значення вводиться в рядок коду мітки, причиною зміни безпеки можуть бути новини, а також зворотній зв’язок та доступна інформація з інтернету. Приклад забравлення рис 4.13

Рисунок 4.14 – Верхній контейнер з титульною назвою сторінки

В верхній частині сторінки правильно розташувалась назва. Рис. 4.14. Рядок з назвою не є статичним та при скролінгу зникає

З проблем котрі виникли при написанні html особливо відрізняється розташування та послідовність елементів, спочатку вигляділо все так ніби нічого не добавлялось, але при виправленні коду це пофіксилось, та потім залишилось лише все розставити на свої місця за допомогою css.

Також проблема була в вигляді коду. Він був ніби просто накиданий та не гармонійний. Трохи згодом приділив уваги цьому, та все стало легше для візуального сприйняття та виправлення коду за потреби. Рис. 4.15.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Карта безпечних прогулянок в місті Київ</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
</head>
<body>
    <div class="container">
        <h1>Карта безпечних прогулянок в місті Київ</h1>
        <div id="map"></div>
        <div id="info-box">
            <h2>Опис зони</h2>
            <p id="zone-description">Натисніть на зону на карті, щоб побачити опис.</p>
            <p id="zone-image"></p>
        </div>
    </div>
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script src="script.js"></script>
    <footer>
        <p>&copy; 2024 Карта безпечних прогулянок в місті Київ</p>
        <p>morozovskyiVS@krok.edu.ua</p>
    </footer>
</body>
</html>
```

Рисунок 4.15 – Створення сторінки по html (виправлення помилок)

При створенні архітектури коду, величезна кількість картинок кидалась у очі. Було не зручно знайти той самий html серед всіх фотографій. Тому для рішення цієї проблеми фото було перенесено в окрему папку.

Просто перемістити фото було не достатньо, тому що після їх переміщення, потрібно було переписати код так, щоб мітки могли використовувати зображення в popup. Код котрий став в нагоді вказаний на рис 4.16

```
const marker = L.marker(zone.coords, { icon: icon }).addTo(map)
.bindPopup(`<b>${zone.name}</b><br>${zone.description}<br><img class="popup-image" src="images/${zone.photo}" width="200px">`)
    .on('click', function () {
```

Рисунок 4.16 – Виправлення помилки з картинками

Після виправлення всіх помилок JS переглянем зміну кольору за значенням безпеки. Для прикладу візьмем мітку Золоті ворота, та зміним її значення. Safe – синій, less-safe – жовтий, unsafe – червоний. Рис 4.17.

Рисунок 4.17 – Приклад зміни безпеки та кольору мітки на мапі

```
footer {

    left: 0;
    bottom: 0;
    position: relative;
    width: 100%;
    background-color: #ffffff;
    text-align: center;
    padding: 10px 0px;
    z-index: 999;
    display: flex;
    justify-content: center;
}
footer p {
    margin: 0 60px;
}
```

Рисунок 4.18 – Код котрий виправив помилку з футером

При першому запуску сторінки з новим футером виникла проблема. Він не був на місці нижньої полоски, а відкривався як додаток для карти. Для виправлення потрібно було задати йому позицію релатив, та дисплей флекс. Зображено на рис. 4.18.

```css
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #e3f2fd;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;

    flex-direction: column;
    min-height: 100vh;

}
.container {
    margin-top: 10px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    max-width: 900px;
    height: 100%;
    width: 100%;

    flex: 1;
}
h1 {
    font-size: 26px;
    margin-bottom: 20px;
    color: #1e88e5;
    text-align: center;
    border-bottom: 2px solid #1e88e5;
    padding-bottom: 10px;
```

Рисунок 4.19 – Створення сторінки по css (виправлення помилок)

Рисунок 4.20 – Сайт з котрого взята мапа та стиль міток

При написанні коду було не зручно кожного разу зафарбовувати окрему мітку, тому кожній з них, за допомогою функції на рис. 4.21, стало можливиим задавати значення безпеки. Колір змінюється відповідно рівню вказаної небезпеки

```
zones.forEach(zone => {
    let markerColor = 'blue'; // За замовчуванням маркер буде синій
    if (zone.safety === 'less-safe') {
        markerColor = 'yellow'; // Жовтий маркер для менш безпечних зон
    } else if (zone.safety === 'unsafe') {
        markerColor = 'red'; // Червоний маркер для небезпечних зон
    }
    const icon = L.icon({
        iconUrl: `https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-${markerColor}.png`,
        iconSize: [25, 41],
        iconAnchor: [12, 41],
        popupAnchor: [1, -34],
        shadowSize: [41, 41]
    });
```

Рисунок 4.21 – javascript (усунення помилок 1 (колір міток відносно безпеки))

При введені кординат для позначок з Google Maps вони мали не правильне розташування, тому потрібно було вирівнювати кожну з них власноруч при перегляді сторінки. Введені кординати позначені на рис 4.23

```
{ name: 'Печерськ', coords: [50.4322, 30.5446], descriptio
{ name: 'Оболонь', coords: [50.5107, 30.5035], description
{ name: 'Хрещатик', coords: [50.4471, 30.5230], descriptio
{ name: 'Поділ', coords: [50.4690, 30.5167], description:
{ name: 'Золоті Ворота', coords: [50.4482, 30.5135], descr
{ name: 'Парк ВДНГ', coords: [50.3710, 30.4780], descripti
{ name: 'Сквер "Дружби народів"', coords: [50.4207, 30.550
{ name: 'Парк Муромець', coords: [50.4945, 30.5460], descr
{ name: 'Голосіївський парк імені М. Рильського', coords:
{ name: 'Національний природний парк "Голосіївський"', coo
{ name: 'Парк Відрадний', coords: [50.4328, 30.4264], desc
{ name: 'Парк Партизанської Слави', coords: [50.4180, 30.6
{ name: 'Маріїнський парк', coords: [50.4466, 30.5410], de
{ name: 'Парк «Наталка»', coords: [50.4957, 30.5260], desc
{ name: 'Арка Дружби Народів', coords: [50.45450, 30.5298]
```

Рисунок 4.23 – javascript (усунення помилок 3 (координати))

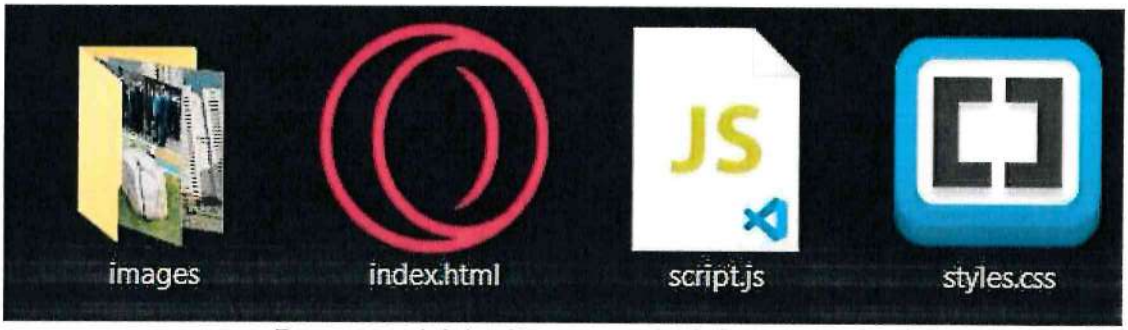


Рисунок 4.24 – Використані файли

ВИСНОВОК ДО РОЗДІЛУ 4

Отже, в цьому розділі була протестована система створення інтерактивних карт. Показано процес розгортання сторінки зі скріншотами, необхідними для розуміння, що дозволяє розібратися у всіх елементах інтерфейсу сторінки. Додаток являє собою успішний тест обробки помилок, які можуть виникнути при роботі з системою.

Додаток володіє інтуїтивно зрозумілим призначеним для користувача інтерфейсом. У ній немає нічого зайвого, що могло б відвернути від використання. Тобто існує необхідний мінімум взаємодії. Усі функції також були протестовані для несанкціонованих користувачів, яким заборонено доступ, незалежно від того, є вони авторами публікації чи ні. Розроблений додаток повністю відповідає всім вимогам.

Чітка і повна функціональність застосунку показує, що рішення, прийняті на етапі вибору інструментів, бібліотек і технологій, є правильними.

ВИСНОВКИ

Метою даної роботи є розробка інтерактивної карти для безпечних прогулянок. В ході її реалізації було теоретично проаналізовано та оброблено інструменти і матеріали для розробки завдання. Таким чином, за допомогою прийнятих рішень маємо наступні висновки:

Інтерактивні карти стали невід'ємною частиною нашого життя. Вони використовуються щодня для вирішення багатьох різних завдань. Тобто сьогодні вони дуже популярні і актуальні. Наприклад, лише служба Карт Google Maps користується понад 10 мільярдами користувачів Щомісяця, і щодня пропонується понад 2000 мільйонів оновлень. Така популярність через те, що:

- Під час подорожі до країни, мову якої не знаєте, замість того щоб запитувати прохожих про потрібні місця можна просто знайти їх удодатку на своїй рідній мові.
- Зручно будувати маршрути від точки А до Б, та обирати різні способи як туди дістатися: пішки, громадським транспортом, автомобілем, тощо.
- Можна читати відгуки, продивлятися рейтинг місць, що хочете відвідати, та обирати потрібне.

Однак, як згадувалося в розділі 1, популярність також приховує негативні аспекти, такі як збір та обробка особистої інформації для власних цілей, таких як реклама, або величезний і надмірний інтерфейс для виконання всіх функцій. В результаті з'явилися аналоги картографічних систем, які не використовують цінну особисту інформацію Користувача. 1. Однією з таких систем є розроблений проект, в якому використовуються інтерактивні карти.

Ця робота чітко пояснює мету використання застосунку, наочно вирішує проблему пошуку і створення пам'яток на карті без використання надлишкового інтерфейсу, а також зберігає конфіденційність інформації. Цей розроблений додаток має мінімальну базу, необхідну для задоволення функціональних вимог зазначеного прикладу, але реалізований проект може бути легко розширений.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The history and importance of web mapping [Електронний ресурс] – Режим доступу до ресурсу: https://www.e-education.psu.edu/geog585/node/643
2. Dynamically drawn map services [Електронний ресурс] – Режим доступудо ресурсу: https://www.e-education.psu.edu/geog585/node/697
3. Pros and Cons of Static and Interactive Maps [Електронний ресурс] – Режим доступу до ресурсу: http://clfuture.org/toolkit/pros-and-cons-of- static-and-interactive-maps
4. Vector vs raster maps [Електронний ресурс] – Режим доступу до ресурсу: https://www.e-education.psu.edu/geog865/node/95PBS
5. What is web map? [Електронний ресурс] – Режим доступу до ресурсу: https://www.axismaps.com/guide/what-is-a-web-map
6. Raster vs Vector Map Tiles: What is the Difference Between the Two DataTypes? [Електронний ресурс] – Режим доступу до ресурсу: https://documentation.maptiler.com/hc/en-us/articles/4411234458385- Raster-vs-Vector-Map-Tiles-What-Is-the-Difference-Between-the-Two- Data-Types-
7. Raster vs Vector Maps: What's the Difference & Which are Best? [Електронний ресурс] – Режим доступу до ресурсу: https://carto.com/blog/raster-vs-vector-whats-the-difference-which-is-best/
8. The History of JavaScript: Everything You Need to Know [Електронний ресурс] – Режим доступу до ресурсу: https://www.springboard.com/blog/data-science/history-of-javascript/
9. The History of Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://www.section.io/engineering-education/history-of-nodejs/

10. Our Mission [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/company

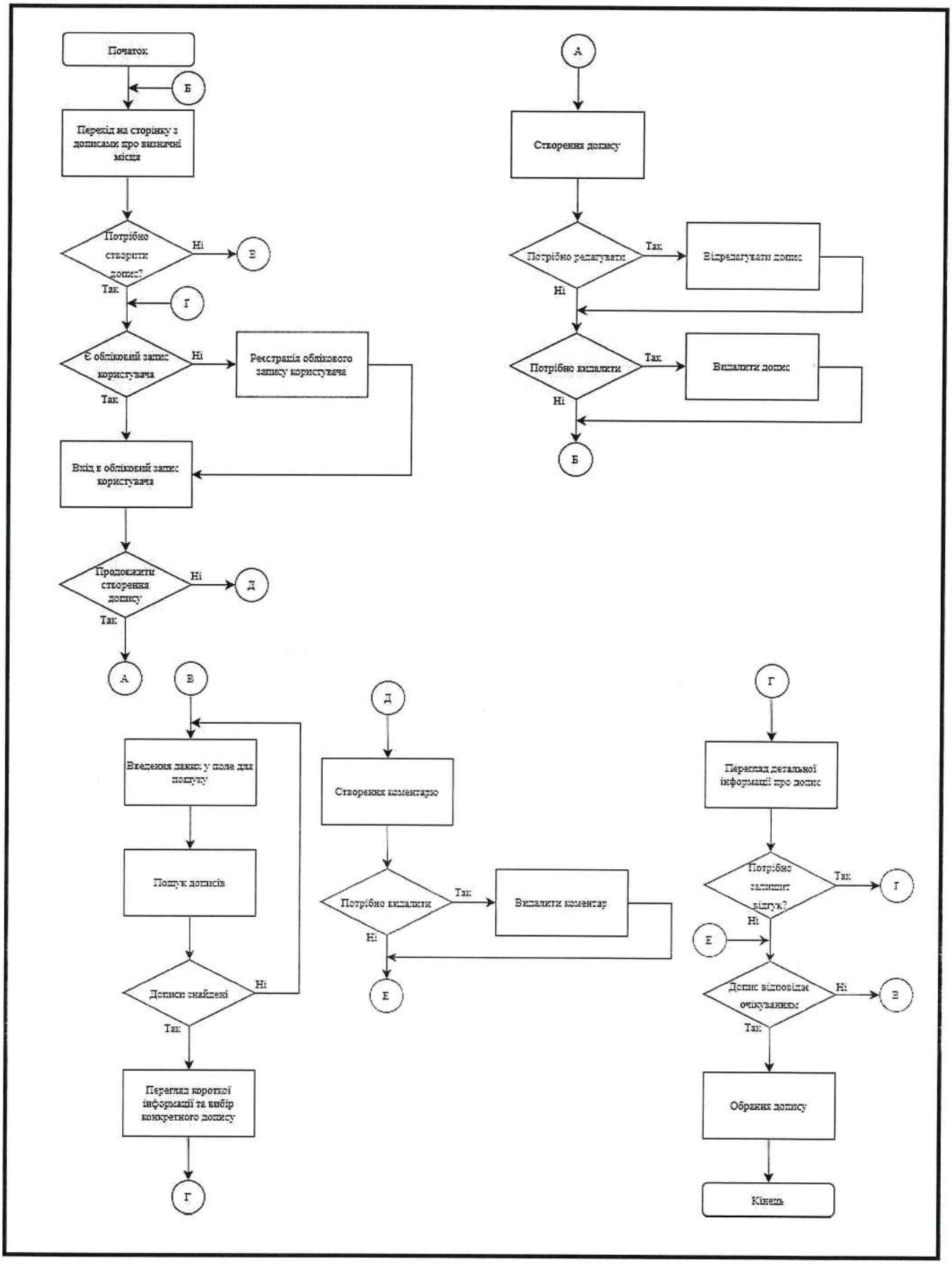
11. What is Bootstrap: A Beginner's Guide [Електронний ресурс] – Режимдоступу до ресурсу: https://careerfoundry.com/en/blog/web- development/what-is-bootstrap-a-beginners-guide/#why-is-bootstrap-the- go-to-for-web-developers

ДОДАТОК 1

Карта безпечних прогулянок в місті Київ

Схема алгоритму

Аркушів 1

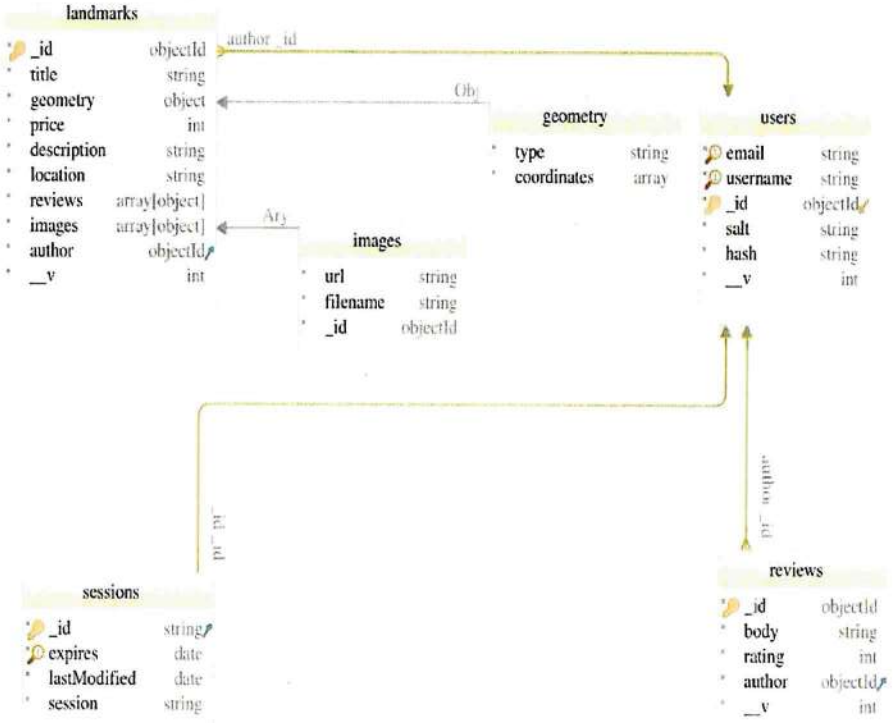

Початок
Б
Перехід на сторінку з дописами про визначні місця
Потрібно створити допис?
Ні
В
Так
Г
Є обліковий запис користувача
Ні
Реєстрація облікового запису користувача
Так
Вхід к обліковий запис користувача
Продовжити створення допису
Ні
Д
Так
А
В
Введення даних у поле для пошуку
Пошук дописів
Дописи знайдені
Ні
Так
Перегляд короткої інформації та вибір конкретного допису
Г
Д
Створення коментарю
Потрібно видалити
Так
Видалити коментар
Ні
Е
А
Створення допису
Потрібно редагувати
Так
Відредагувати допис
Ні
Потрібно видалити
Так
Видалити допис
Ні
Б
Г
Перегляд детальної інформації про допис
Потрібно залишит відгук?
Так
Г
Ні
Е
Допис відповідає очікуванням
Ні
В
Так
Обрання допису
Кінець

ДОДАТОК 2

Карта безпечних прогулянок в місті Київ

Схема бази даних

Аркушів 1

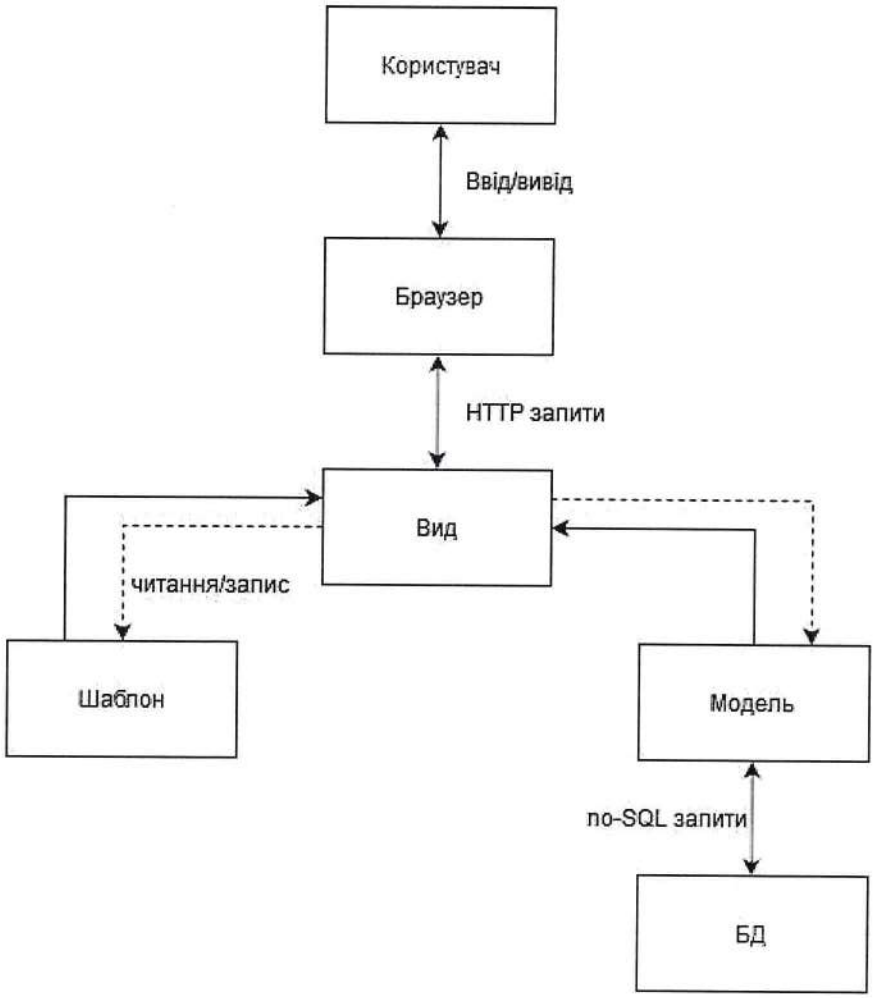

landmarks
_id objectId
title string
geometry object
price int
description string
location string
reviews array[object]
images array[object]
author objectId
__v int
author _id
Obj
Arr
geometry
type string
coordinates array
images
url string
filename string
_id objectId
users
email string
username string
_id objectId
salt string
hash string
__v int
sessions
_id string
expires date
lastModified date
session string
reviews
_id objectId
body string
rating int
author objectId
__v int

ДОДАТОК 3

Карта безпечних прогулянок в місті Київ

Функціональна схема

Аркушів 1


Користувач
Ввід/вивід
Браузер
HTTP запити
Вид
читання/запис
Шаблон
Модель
no-SQL запити
БД

ДОДАТОК 4

Карта безпечних прогулянок в місті Київ

Текст програмного коду

Аркушів 8

HTML

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Карта безпечних прогулянок в місті Київ</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
</head>
<body>
    <div class="container">
        <h1>Карта безпечних прогулянок в місті Київ</h1>
        <div id="map"></div>
        <div id="info-box">
            <h2>Опис зони</h2>
            <p id="zone-description">Натисніть на зону на карті, щоб побачити опис.</p>
            <p id="zone-image"></p>
        </div>
    </div>
    <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
    <script src="script.js"></script>
    <footer>
        <p>&copy; 2024 Карта безпечних прогулянок в місті Київ</p>
        <p>morozovskyiVS@krok.edu.ua</p>
    </footer>
</body>
</html>
```

CSS

```
body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #e3f2fd;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;

    flex-direction: column;
    min-height: 100vh;

}
.container {
    margin-top: 10px;
    background-color: #ffffff;
    border-radius: 10px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    padding: 20px;
    max-width: 900px;
    height: 100%;
    width: 100%;

    flex: 1;
}
h1 {
    font-size: 26px;
    margin-bottom: 20px;
    color: #1e88e5;
    text-align: center;
    border-bottom: 2px solid #1e88e5;
    padding-bottom: 10px;
}
#map {
    height: 450px;
    width: 100%;
    border-radius: 8px;
    margin-bottom: 20px;
    border: 2px solid #1e88e5;
}
#info-box {
    background-color: #f1f8e9;
    padding: 15px;
    border-radius: 8px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
#info-box h2 {
    font-size: 22px;
    margin-bottom: 10px;
    color: #388e3c;
}
#zone-description {
    font-size: 16px;
    color: #555;
}

.popup-image {
    border-radius: 10%;
}
```

```
footer {

    left: 0;
    bottom: 0;
    position: relative;
    width: 100%;
    background-color: #ffffff;
    text-align: center;
    padding: 10px 0px;
    z-index: 999;
    display: flex;
    justify-content: center;
}
footer p {
    margin: 0 60px;
}
```

JavaScript

```
document.addEventListener("DOMContentLoaded", function () {
    // Ініціалізація карти
    const map = L.map('map').setView([50.4501, 30.5234], 13); // Координати Києва
    // Додавання базового шару карти
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: '© OpenStreetMap contributors'
    }).addTo(map);
    // Додавання зон на карту
    const zones = [
        { name: 'Печерськ', coords: [50.4322, 30.5446], description: 'Печерськ - це центральний район Києва, відомий своїми парками та історичними місцями. Це безпечна зона для прогулянок.Печерськ у Києві вважається безпечним районом завдяки наявності численних урядових установ, посольств, сучасної інфраструктури та підвищеному рівню охорони.', photo: 'Печерськ.jpg' },
        { name: 'Оболонь', coords: [50.5107, 30.5035], description: 'Оболонь - це район на північ від центру Києва, відомий своїми набережними та озерами.Оболонь у Києві вважається безпечним районом завдяки активній роботі правоохоронних органів, сучасним житловим комплексам та розвиненій інфраструктурі.', photo: 'Оболонь.jpg' },
        { name: 'Хрещатик', coords: [50.4471, 30.5230], description: 'Хрещатик - головна вулиця Києва з безліччю магазинів, ресторанів та історичних будівель.Хрещатик у Києві вважається безпечним завдяки постійній присутності правоохоронних органів, великій кількості людей та освітленій території.', photo: 'Хрещатик.jpg' },
        { name: 'Поділ', coords: [50.4690, 30.5167], description: 'Поділ - це історичний район Києва з багатою культурною спадщиною, популярними кафе та магазинами.Поділ у Києві вважається безпечним завдяки активній роботі правоохоронних органів, наявності численних культурних та історичних обєктів, а також розвиненій інфраструктурі.', photo: 'Поділ.jpg' },
        { name: 'Золоті Ворота', coords: [50.4482, 30.5135], description: 'Золоті Ворота - одна з головних історичних пам\'яток Києва.Золоті Ворота в Києві вважаються безпечним районом завдяки розташуванню в центрі міста, постійній присутності правоохоронних органів та високій концентрації туристів і місцевих жителів.', photo: 'Золоті Ворота.jpg'},
        { name: 'Парк ВДНГ', coords: [50.3710, 30.4780], description: 'Парк ВДНГ- це Національний Експоцентр України\ може заслужити звання головного культурно-розважального центру столиці, адже багато масштабних подій Києва відбувалися саме там.Парк ВДНГ у Києві вважається безпечним завдяки регулярним патрулям охорони, організованим заходам та доглянутій території з розвиненою інфраструктурою. ', photo: 'Парк ВДНГ.jpg' },
        { name: 'Сквер “Дружби народів”', coords: [50.4207, 30.5502], description: 'Сквер "Дружби народів" у Києві є одним з найвідоміших і контроверсійних памятників міста. Він знаходиться на вулиці Парковій в Хрещатому парку, неподалік від Європейської площі.Сквер "Дружби народів" у Києві вважається безпечним завдяки його доглянутій території, регулярним патрулям охорони та активній відвідуваності місцевими жителями і туристами.', photo: 'Сквер “Дружби народів.jpg' },
        { name: 'Парк Муромець', coords: [50.4945, 30.5460], description: 'Парк Муромець, розташований у Києві, є одним із найпопулярніших місць для відпочинку на природі. Розташований між лівим та правим берегами Дніпра, він займає площу 780 гектарів, включаючи канали приток Десенки та Десни. Парк був створений ще в радянські часи, але офіційно відкритий у 1972 році як Парк Дружби Народів. У 2018 році парк був перейменований на честь героя билин Іллі Муромця.Парк Муромець у Києві вважається безпечним завдяки активній роботі правоохоронних органів, добре освітленій території та регулярному догляду за парком.', photo: 'Парк Муромець.jpg' },
        { name: 'Голосіївський парк імені М. Рильського', coords: [50.3910, 30.5133], description: 'Голосіївський парк імені Максима Рильського — один з найбільших та найгарніших парків Києва. Розташований у південній частині міста, він займає площу близько 140 гектарів. Парк засновано в 1957 році на території колишнього Голосіївського лісу, що оточував Київ з півдня.Голосіївський парк імені М. Рильського у Києві вважається безпечним завдяки наявності охорони, системі відеоспостереження та добре організованій інфраструктурі.', photo: 'Голосіївський парк імені М. Рильського.jpg' },
        { name: 'Національний природний парк "Голосіївський"', coords: [50.3705, 30.4933], description: 'Національний природний парк "Голосіївський" розташований у Києві і є одним з небагатьох національних парків у світі, що знаходяться в межах міста. Парк був створений указом Президента України 27 серпня 2007 року і займає площу 4525,52 га. У 2014 році до його
```

території було додано 6462,62 га лісів Святошинського лісопаркового господарства, що збільшило загальну площу до 10 988,14 га.Національний природний парк "Голосіївський" у Києві вважається безпечним завдяки ефективній охороні природи, регулярному контролю за відвідувачами та збереженню біорізноманіття.', photo: 'Національний природний парк Голосіївський.jpg' },

{ name: 'Парк Відрадний', coords: [50.4328, 30.4264], description: 'Парк розташований у Солом'янському районі. Він був створений одночасно з масивом Відрадний на початку 60-х років. У центрі є ставок, який зовні нагадує «інтеграл», тому іноді його так і називають.Парк Відрадний у Києві вважається безпечним завдяки присутності охорони, добре освітленій території та наявності розвинутої інфраструктури для відпочинку.', photo: 'Парк Відрадний.jpg' },

{ name: 'Парк Партизанської Слави', coords: [50.4180, 30.6727], description: 'Парк був створений на базі соснового лісу в Дарницькому районі між вулицями Тростянецька й Славгородська.На території є спортивний майданчик, велика соснова алея, літній кінотеатр, канатні дороги, прогулянкове кільце і три лісових озера. Можна орендувати альтанку з мангалом і провести затишний вечір у компанії друзів.Парк Партизанської Слави у Києві вважається безпечним завдяки постійному присутності правоохоронних органів, добре обладнаній та освітленій території та системі відеоспостереження.', photo: 'Парк Партизанської Слави.jpg' },

{ name: 'Маріїнський парк', coords: [50.4466, 30.5410], description: 'Куточок природи в центрі Києва – це Маріїнський парк. Він розташований навпроти будівлі Верховної Ради та Маріїнського палацу. Після огляду визначних пам'яток Києва краще піти у парк, щоб відпочити в тіні дерев і помилуватися чудовими краєвидами.Маріїнський парк у Києві вважається безпечним завдяки регулярним патрулям охорони, високому рівню освітлення та активній присутності відвідувачів.', photo: 'Маріїнський парк.jpg' },

{ name: 'Парк «Наталка»', coords: [50.4957, 30.5260], description: '«Наталка» — парк на березі Дніпра в Оболонському районі Києва. Розташований вздовж берега Дніпра, від гольф центру на Оболонській Набережній до Північного мосту. Загальна площа території парку становить близько 25 га. Назва походить від однойменного урочища, яке довго було дикою територією між житловим масивом «Оболонь» і Дніпром.Парк "Наталка" у Києві вважається безпечним завдяки наявності безперервної охорони, добре організованому просторі для відпочинку та активному співробітництву з місцевими владами для забезпечення безпеки відвідувачів.', photo: 'Парк «Наталка».jpg' },

{ name: 'Арка Дружби Народів', coords: [50.45450, 30.5298], description: 'Арка Дружби Народів це залишок радянських часів – його відкрили у 1982 році. Архітектурне творіння містить величезну арку у формі веселки, скульптурні композиції з бронзи та гранітні скульптури фрагмента зі сцен легендарної Переяславської ради.Арка Дружби Народів в Києві вважається безпечною завдяки стійкій охороні, ретельному технічному обслуговуванню та системі безпеки, що забезпечує безперебійне функціонування памятника та безпеку відвідувачів.', photo: 'Арка Дружби Народів.jpg' },

{ name: 'Набережна Дніпра', coords: [50.4363, 30.5643], description: 'Набережна Дніпра – ще одне чудове місце для романтичного побачення! Тут є безліч цікавинок – річковий вокзал, Поштова площа, реконструйована до недавнього Євробачення, Пішохідний міст.Набережна Дніпра в Києві вважається безпечною завдяки активній присутності патрульних поліцейських, добре освітленій і обладнаній території та постійному контролю за безпекою відвідувачів.', photo: 'Набережна Дніпра.jpg' },

{ name: 'Труханів острів', coords: [50.4650, 30.5480], description: 'Довгі пляжі вздовж Дніпра та величезна територія Труханового острова, де ви знайдете і лісові угіддя, і мальовничі озера, і швидкі річки.Труханів острів у Києві вважається безпечним завдяки регулярному патрулюванню поліції, облаштованій інфраструктурі для відпочинку та нагляду за безпекою відвідувачів.', photo: 'Труханів острів.jpg' },

{ name: 'Гідропарк', coords: [50.4385, 30.5793], description: 'Гідропарк – одне з найдушевніших місць Києва. 360 гектарів чагарників, дерев, піщаних полян, пляжів і величезної кількості розважальних закладів. Для тих, хто любить спорт, тут є ціле тренажерне містечко.Гідропарк у Києві вважається безпечним завдяки постійному присутності поліції, добре організованому руху транспорту та контролю за безпекою на воді та на суші.', photo: 'Гідропарк.jpg' },

{ name: 'Андріївський узвіз', coords: [50.4612, 30.5162], description: 'Одна з найвідоміших київських вулиць, душа міста, що вабить до себе творчих людей зі всієї України та світу. Ця вулиця – прекрасна в будь-яку пору року та дозволить вам на повну насолодитися київським антуражем. Кілька скверів, вихід до Пейзажної алеї. Звідти недалеко пішки до Володимирської гірки.Андріївський узвіз у Києві вважається безпечним завдяки активній присутності поліції, багатому культурному середовищу та великій кількості відвідувачів у

будь-який час дня.', photo: 'Андріївський узвіз.jpg' },
{ name: 'Володимирська Гірка', coords: [50.4560, 30.5259], description: 'Природній оазис посеред галасливого мегаполісу і один з найромантичніших парків Києва – все це про Володимирську гірку. Маса зручних лавок, альтанки та незрівнянний вигляд на лівий берег. Думаєте, куди сходити з коханою людиною? Звичайно сюди!Володимирська Гірка у Києві вважається безпечною завдяки постійному нагляду правоохоронних органів, наявності освітлення та популярності серед відвідувачів.', photo: 'Володимирська Гірка.jpg' },
{ name: 'Ботанічний сад ім. акад. О. В. Фоміна', coords: [50.4434, 30.5049], description: 'Старий ботсад (один з перших в Україні) тішить своїх гостей величезним розмаїттям дерев, квітів і чагарників з усіх куточків світу. Він розташований практично в центрі міста.Ботанічний сад ім. акад. О. В. Фоміна у Києві вважається безпечним завдяки постійному патрулюванню охорони, наявності відповідних сигналізаційних систем та добре облаштованим стежкам для відвідувачів.', photo: 'Ботанічний сад ім. акад. О. В. Фоміна.jpg' },
{ name: 'Майдан Незалежності', coords: [50.4499, 30.5244], description: 'Головна площа міста – місце двох революцій. Майдан Незалежності – центр неформальних "тусовок" і офіційних заходів, про який чули всі в Україні і багато хто за її межами.Майдан Незалежності у Києві вважається безпечним завдяки постійному патрулюванню поліції, системі відеоспостереження та високому рівню громадської свідомості та взаємодії між відвідувачами.', photo: 'Майдан Незалежності.jpg' },
{ name: 'Києво-Печерська лавра ', coords: [50.4353, 30.5569], description: 'Києво-Печерська Один із найвідоміших і найдавніших храмових комплексів всієї країни, заснований в далекому 1051 році. Тут можна бродити годинами, насолоджуючись сакральною архітектурою, таємничістю печер і численними музеями. У що б ви не вірили – Києво-Печерська лавра зможе дати вам трошки душевного спокою.Києво-Печерська лавра у Києві вважається безпечною завдяки постійному нагляду правоохоронних органів, наявності безперебійної охорони та великій кількості відвідувачів у будь-який час дня.', photo: 'Києво-Печерська лавра.jpg' },
{ name: 'Національний ботанічний сад імені Миколи Гришка', coords: [50.4139, 30.5634], description: 'У різні пори року ботанічний сад імені Миколи Гришка пропонує відвідувачам помилуватись різними рослинами, серед яких є й екзотичні для України. Також цей ботсад є одним з найбільших у Європі. У ньому зібрали безліч різних представників флори практично зі всієї земної кулі.Національний ботанічний сад імені Миколи Гришка в Києві вважається безпечним завдяки системі безперервного патрулювання, ретельному обслуговуванню та добре облаштованій інфраструктурі для відвідувачів.', photo: 'Національний ботанічний сад імені Миколи Гришка.jpg' },
{ name: 'Парк Пуща-Водиця', coords: [50.5345, 30.3532], description: 'Мабуть, найчарівніший лісовий парк столиці – густий сосняк і довгий ланцюг озер. Тут ви можете повністю сховатись від людських очей, відійшовши від дороги буквально на десять хвилин.Парк Пуща-Водиця у Києві вважається безпечним завдяки наявності охорони, системі відеоспостереження та добре обладнаній території для відпочинку та активного відвідування місцевими жителями та туристами.', photo: 'Парк Пуща-Водиця.jpg' },
{ name: 'Парк Вічної Слави', coords: [50.4384, 30.5559], description: 'Парк Вічної Слави розташований між Лаврською вулицею та Дніпровським узвозом. Головний вхід у парк з площі Слави. Тут багато меморіалів і памяток, які не дадуть забути історію.Парк Вічної Слави у Києві вважається безпечним завдяки постійному патрулюванню охорони, добре освітленій території та високому рівню громадської свідомості про його значення як меморіального обєкту.', photo: 'Парк Вічної Слави.jpg' },
{ name: 'Музей води', coords: [50.4527, 30.5315], description: 'Якщо ж вам набридло ходити вулицями столиці, у центрі міста існує чимало цікавих для відвідування закладів, куди вас безперешкодно пустять з сертифікатом або тестом. Зокрема, музей води, який називають найбільш неординарним в Україні. Він символічно розташовується у підземеллі старовинної водонапірної вежі, яка була головною у місті понад 140 років тому.Музей води в Києві вважається безпечним завдяки наявності необхідних заходів безпеки, які забезпечують охорону відвідувачів та відповідний нагляд за експонатами.Державний музей авіації в Києві вважається безпечним завдяки системі безперебійного нагляду, дотриманню відповідних безпекових стандартів та організації безпечного пересування відвідувачів.', photo: 'Музей води.jpg' },
{ name: 'Державний музей авіації', coords: [50.4063, 30.4602], description: 'Найбільший історіко-технічний музей України, створений у 2003 році, працює у безпосередній близькості від аеропорту "Київ". Експозиція музею нараховує понад 80 літаків та гвинтокрилів. Військові та цивільні борти, до салону більшості з яких можна безперешкодно зайти, здатні зацікавити будь-кого.', photo: 'Державний музей авіації.jpg', },
{ name: 'проспект Петра Григоренка, 36', coords: [50.3963, 30.6353], description: 'Проспект Петра Григоренка, 36 в Києві вважається небезпечним через великий потік

```
автотранспорту, відсутність безпеки для пішоходів та недостатню розвиненість
інфраструктури.', photo: 'проспект Петра Григоренка, 36.jpg', safety: 'unsafe' },
        { name: 'Парк «Попудренко»', coords: [50.4580, 30.6235], description: 'Парк
"Попудренко" в Києві вважається небезпечним через недостатнє освітлення, відсутність охорони
та випадки злочинності.', photo: 'Парк «Попудренко».jpg', safety: 'unsafe' },
        { name: 'Мюнхенский сквер', coords: [50.4622, 30.4650], description: 'Мюнхенский
сквер в Києві вважається небезпечним через високий рівень криміальності, недостатнє
освітлення та недостатню присутність охорони.', photo: 'Мюнхенский сквер.jpg', safety:
'unsafe' },
        { name: 'Парк «Нивки»', coords: [50.4628, 30.4246], description: 'Парк "Нивки" в
Києві вважається небезпечним через відомі випадки злочинності, недостатнє освітлення та
відсутність систематичного патрулювання охорони.', photo: 'Парк «Нивки».jpg', safety:
'unsafe' },
        { name: 'Парк Захисників України', coords: [50.4173, 30.4530], description: 'Парк
Захисників України в Києві вважається небезпечним через недостатнє освітлення, відомі випадки
злочинності та недостатню присутність охорони.', photo: 'Парк Захисників України.jpg',
safety: 'unsafe' },
        { name: 'Парк Юність', coords: [50.4275, 30.3834], description: 'Парк "Юність" в
Києві вважається небезпечним через недостатнє освітлення, відомі випадки злочинності та
недостатню присутність охорони.', photo: 'Парк Юність.jpg', safety: 'unsafe' },
        { name: 'Сквер ім. Феодори Пушиної', coords: [50.4564, 30.3534], description: 'Сквер
ім. Феодори Пушиної в Києві вважається небезпечним через недостатнє освітлення, відомі
випадки злочинності та відсутність систематичного контролю з боку правоохоронних органів.',
photo: 'Сквер ім. Феодори Пушиної.jpg', safety: 'unsafe' },
        { name: 'бульвар Академіка Вернадського', coords: [50.4635, 30.3709], description:
'Бульвар Академіка Вернадського в Києві вважається небезпечним через великий потік
автомобільного транспорту, недостатню безпеку для пішоходів та недосконаліть
інфраструктури.', photo: 'бульвар Академіка Вернадського.jpg', safety: 'unsafe' },
        { name: 'Парк Совки', coords: [50.4409, 30.3749], description: 'Парк Совки в Києві
вважається небезпечним через відомі випадки злочинності, недостатнє освітлення та відсутність
систематичного патрулювання охорони.', photo: 'Парк Совки.jpg', safety: 'unsafe' },
        { name: 'Парк ім. Михайла Котельникова', coords: [50.4527, 30.3766], description:
'Парк ім. Михайла Котельникова в Києві вважається небезпечним через недостатнє освітлення,
випадки злочинності та відсутність регулярного патрулювання охорони.', photo: 'Парк ім.
Михайла Котельникова.jpg', safety: 'unsafe' },
        { name: 'Парк воїнів інтернаціоналістів', coords: [50.4097, 30.6562], description:
'Парк воїнів-інтернаціоналістів у Києві не дуже безпечний через випадки злочинності,
недостатнє освітлення та рідкісні патрулі охорони.', photo: 'Парк воїнів
інтернаціоналістів.jpg', safety: 'less-safe' },
        { name: 'Фітнес-парк Озеро Лебедине', coords: [50.4038, 30.6438], description:
'Фітнес-парк Озеро Лебедине в Києві не дуже безпечний через недостатнє освітлення, випадки
вандалізму та обмежену присутність охорони.', photo: 'Фітнес-парк Озеро Лебедине.jpg',
safety: 'less-safe' },
        { name: 'Парк Перемога', coords: [50.4643, 30.6040], description: 'Парк Перемога в
Києві не дуже безпечний через недостатнє освітлення, випадки злочинності та обмежену
присутність патрульної охорони.', photo: 'Парк Перемога.jpg', safety: 'less-safe' },
        { name: 'Парк Андрія Малишка', coords: [50.4594, 30.6217], description: 'Парк Андрія
Малишка в Києві не дуже безпечний через недостатнє освітлення, відомі випадки злочинності та
недостатню присутність охорони.', photo: 'Парк Андрія Малишка.jpg', safety: 'less-safe' },
        { name: 'Парк Кіото', coords: [50.4632, 30.6366], description: 'Парк Кіото в Києві не
дуже безпечний через недостатнє освітлення, випадки злочинності та рідкісні патрулі
охорони.', photo: 'Парк Кіото.jpg', safety: 'less-safe' },
        { name: 'Парк ДШК', coords: [50.4575, 30.6364], description: 'Парк ДШК у Києві не
дуже безпечний через недостатнє освітлення, випадки злочинності та обмежену присутність
охорони.', photo: 'Парк ДШК.jpg',safety: 'less-safe' },
    ];
  zones.forEach(zone => {
        let markerColor = 'blue'; // За замовчуванням маркер буде синій
        if (zone.safety === 'less-safe') {
            markerColor = 'yellow'; // Жовтий маркер для менш безпечних зон
        } else if (zone.safety === 'unsafe') {
            markerColor = 'red'; // Червоний маркер для небезпечних зон
        }
```

```
        const icon = L.icon({
            iconUrl: `https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-
icon-2x-${markerColor}.png`,
            iconSize: [25, 41],
            iconAnchor: [12, 41],
            popupAnchor: [1, -34],
            shadowSize: [41, 41]
        });
        const marker = L.marker(zone.coords, { icon: icon }).addTo(map)
        .bindPopup(`<b>${zone.name}</b><br>${zone.description}<br><img class="popup-image"
src="images/${zone.photo}" width="200px">`)
            .on('click', function () {

                document.getElementById('zone-description').textContent = zone.description;
                document.getElementById('zone-image').src = zone.photo;

            });
    });
});
```