ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Дослідження використання "API" для створення веб-сайтів»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерних наук
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Олександр Солов'янчик

Виконав: здобувач(ка) вищої освіти групи КНДМ-61
Олександр Солов'янчик
(Ім'я, ПРІЗВИЩЕ)

Керівник: Сергій Прокопов
д.т.н, професор (Ім'я, ПРІЗВИЩЕ)

Рецензент: _____
(Ім'я, ПРІЗВИЩЕ)

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Комп'ютерних наук
Ступінь вищої освіти Магістр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Віктор Вишнівський
"___" _____2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Солов'янчика Олександра Андрійовича
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
«Дослідження використання "API" для створення веб-сайтів»
керівник кваліфікаційної роботи: Прокопов Сергій, д.т.н., професор,
(ПРІЗВИЩЕ, Ім'я, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «19» жовтня 2023р. №. 145

2. Строк подання студентом кваліфікаційної роботи: 20.12.2023 р.

3. Вихідні дані до кваліфікаційної роботи:
Погодний веб-додаток;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз технології «API»
2. Аналіз веб-сайтів
3. Розробка веб-додатку

5. Перелік ілюстративного матеріалу:
Презентація PowerPoint

6. Дата видачі завдання 20.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ зп	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Дослідження використання технології «API» .	20.10.2023 р.	
2.	Аналіз наукової та технічної літератури за темою кваліфікаційної роботи.	22.10.2023 р.	
3.	Аналіз веб-сайтів, їх типи та категорії.	27.10. 2023р.	
4.	Аналіз мов програмування.	03.11.2023 р.	
5.	Програмна реалізація проекту, створення веб-додатку.	15.11.2023 р.	
6.	Оформлення результатів дослідження. Проходження плагіату	26.11.2023 р.	
7.	Підготовка доповіді до захисту.	15.12.2023 р.	

Здобувач(ка) вищої освіти _____ Олександр Солов'янчик
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій Прокопов
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи і на здобуття освітнього ступеня магістра: 74 сторінок, 39 рисунків, 0 таблиці, 48 джерел.

Мета роботи – дослідження використання технології «API» в різних сферах використання.

Об`єкт дослідження – види та сфери застосування технології в веб-додатках.

Предмет дослідження – технологія «API» та веб-сайти.

Короткий зміст роботи:

Проведено аналіз технології, її використання в різних проектах. Проведено аналіз веб-сайтів та їх роботи з «API».

Розроблено веб додаток з використанням погодного API. Описано як він контактує з ним.

ТЕХНОЛОГІЯ «API», ВЕБ_ТЕХНОЛОГІЇ, МЕТОДИ ТА ЗАСОБИ, АРХІТЕКТУРА, ТИПИ ВЕБ-САЙТІВ, КАТЕГОРІЇ ВЕБ-САЙТІВ

ABSTRACT

Text part of the master's qualification work:74 pages, 39 figures, 0 tables, 48 sources.

The purpose of the work is studying the use of API technology in various fields of application.

Object of research - types and applications of the technology in web applications.

Subject of research – API technology and websites.

Summary of the work:

The technology was analyzed and used in various projects. We analyzed websites and their work with the API.

A web application using the weather API is developed. How it interacts with it is described.

API TECHNOLOGY, WEB TECHNOLOGIES, METHODS AND TOOLS, ARCHITECTURE, TYPES OF WEBSITES, CATEGORIES OF WEBSITES

ЗМІСТ

Вступ

У постійно мінливому ландшафті цифрової сфери веб-сайти слугують воротами до інформації, розваг та інтерактивності в Інтернеті. За бездоганним користувацьким досвідом і динамічною функціональністю стоять складні технології, які дозволяють веб-сайтам спілкуватися і взаємодіяти з різноманітними джерелами даних. Однією з таких ключових технологій, яка розширила спосіб створення та функціонування веб-сайтів, є Application Programming Interface, широко відомий як API. Ця робота заглиблюється у захоплюючу сферу дослідження використання API у створенні веб-сайтів, вивчаючи фундаментальні концепції, специфічні та різні типи API, які лежать в основі взаємопов'язаної інфраструктури Інтернету.

Щоб розпочати цю подорож важливо розуміти, що таке API і як він функціонує в ролі мосту між різними програмними додатками. Розбираючись у специфікаціях і тонкощах API, буде досліджено їх роль у забезпеченні безперешкодного зв'язку між різними системами, відкриваючи перед розробниками безліч можливостей для використання та інтеграції зовнішніх функцій у свої веб-проекти.

Також буде розглянуто різноманітний ландшафт веб-сайтів, класифікуючи їх на основі їхніх цілей та функціональних можливостей. Від статичних односторінкових інформаційних веб-сайтів до динамічних, керованих даними додатків, будуть описані різні типи веб-сайтів, і те, яку роль відіграє в них API.

Для веб-сайтів будуть описані різні методи для їх створення. Основна увага буде зосереджена на мовах програмування, які є основою веб-розробки. Буде розглянуто, як ці мови взаємодіють з API, виступаючи в ролі каналу через який веб-сайт отримує доступ до зовнішніх даних та маніпулювання ними, забезпечуючи динамічний контент.

1 Аналіз технології “API”

1.1 Основи технології та поняття

Application Programming Interface (API) відіграють важливу роль у сучасному цифровому ландшафті, слугуючи основою для комунікації та сумісності між різними програмними додатками та системами.

API сприяють безперешкодному обміну даними та функціональністю, дозволяючи різним додаткам працювати разом і спільно використовувати ресурси у стандартизований та ефективний спосіб.

Наприклад, інформація про погоду, яка знаходиться в системі метеослужби, використовується для оновлення щоденного прогнозу у мобільному додатку для погоди. Цей додаток взаємодіє з метеосистемою за допомогою API, що дозволяє отримувати та відображати актуальну інформацію про погоду на телефоні.

У визначенні API для цього контексту термін "додаток" застосовується до будь-якого програмного забезпечення з конкретною функціональністю. Можна розглядати інтерфейс як угоду про надання послуг між двома додатками, яка визначає їх взаємодію через взаємні запити та відповіді. Ця угода установлює, як вони обмінюються інформацією, щоб будь-який додаток міг використовувати цей інтерфейс з відносною легкістю. API орієнтовані на програмований веб, включаючи веб-додатки та мобільні додатки. Таким чином, технічні характеристики API розрізняють їх від сервісів попередніх поколінь.

Документація API містить інформацію про те, як розробники повинні структурувати свої запити та обробляти отримані відповіді.

API може стати основною точкою входу для корпоративних сервісів, власного веб-сайту і додатків, а також для інтеграції з партнерами та клієнтами.

Підхід API – це архітектурний підхід, який полягає в наданні програмованих інтерфейсів до набору сервісів для різних додатків, що обслуговують різні типи споживачів. Він створює слабко пов’язану архітектуру, яка дозволяє компонентам

сервісу мати широкий спектр можливостей. Приклад взаємодії з API зображено на рис.1.1

Рисунок 1.1 – Взаємодія з API

1.2 Ключові аспекти технології

API - це, по суті, набір правил і протоколів, які дозволяють одному програмному забезпеченню взаємодіяти з іншим. Він визначає методи і формати даних, які програми можуть використовувати для запитів і обміну інформацією. API можуть мати різні форми, зокрема веб-інтерфейси, API на базі бібліотек та API на рівні операційної системи.

1.2.1 Веб-інтерфейси

Веб-інтерфейси, або програмні інтерфейси веб-додатків, є фундаментальним компонентом сучасного Інтернету, що полегшує комунікацію та обмін даними між різними програмними додатками через Інтернет. Веб-інтерфейси відіграють важливу роль у забезпеченні інтероперабельності систем, дозволяючи розробникам інтегрувати сервіси, отримувати доступ до даних і створювати інноваційні додатки.

Ось огляд ключових аспектів, пов'язаних з веб-інтерфейсами:

- Визначення та призначення.

  Web APIs - це набори веб протоколів, методів, інструментів та програмних додатків які використовують для створення цільових програмних додатків. Вони дозволяють їм взаємодіяти один з одним через Інтернет. Він надає розробникам стандартизований спосіб запитувати та обмінюватися даними, часто використовуючи HTTP (Hypertext Transfer Protocol - протокол передачі гіпертексту) як протокол зв'язку.

- RESTful API.

  Representational State Transfer (REST) - це популярний архітектурний стиль для розробки веб-інтерфейсів. RESTful API використовують стандартні методи HTTP (такі як GET, POST, PUT, DELETE) для виконання операцій над ресурсами, ідентифікованими URI (Uniform Resource Identifiers). RESTful API відомі своєю простотою, масштабованістю та відсутністю стану.

- Кінцеві точки і ресурси.

  Веб-інтерфейси відкривають кінцеві точки, які представляють різні функціональні можливості або ресурси. Кожна кінцева точка відповідає певному URI, і розробники можуть робити HTTP-запити до цих кінцевих точок для виконання таких операцій, як отримання даних, створення, оновлення або видалення ресурсів.

- Запит і відповідь.

  Взаємодія з Web API включає в себе запити та відповіді. Клієнт надсилає запит до певної кінцевої точки API, вказуючи бажану операцію та будь-які необхідні параметри. Сервер обробляє запит і повертає відповідь, зазвичай у структурованому форматі даних, такому як JSON (JavaScript Object Notation) або XML (eXtensible Markup Language).

- Аутентифікація та авторизація.

  Безпека є найважливішим аспектом веб-API. Механізми автентифікації, такі як ключі API, токени OAuth або JWT (JSON Web Tokens), часто використовуються для перевірки особи клієнтів. Механізми авторизації контролюють права доступу клієнтів до певних ресурсів або операцій.

- Документація.

  Щоб розробники змогли ефективно використовувати API, необхідно мати належно задокументовані специфікації. У документації API зазвичай включена докладна інформація про доступні кінцеві точки, методи запитів, формати запитів і відповідей, методи автентифікації, а також ілюстративні сценарії використання.

- Керування версіями.

  Оскільки API розвиваються, важливо впроваджувати версії для забезпечення зворотної сумісності. Версії API дозволяють розробникам поступово оновлювати свої додатки, не порушуючи існуючі інтеграції.

- Варіанти використання.

  Веб-інтерфейси API застосовуються в широкому спектрі випадків використання. Вони полегшують інтеграцію зі сторонніми сервісами, дозволяють розробляти мобільні додатки, підтримують розробку односторінкових додатків (SPA) і забезпечують роботу бекенда багатьох веб-додатків.

- Управління API.

  Організації часто використовують платформи управління API для вирішення таких завдань, як управління трафіком, безпека, моніторинг та аналітика. Ці платформи надають централізований спосіб контролю та моніторингу використання API, забезпечуючи надійність та продуктивність.

- Проблеми.

Незважаючи на переваги, з веб-інтерфейсами пов'язані певні проблеми, зокрема вразливості безпеки, складнощі з версіонуванням та забезпеченням стабільної продуктивності. Вирішення цих проблем вимагає ретельного проектування, документування та постійної підтримки.

Таким чином, веб-інтерфейси є фундаментальним елементом сучасної веб-розробки, що забезпечує безперешкодний зв'язок та інтеграцію між різноманітними додатками та сервісами. Їхнє широке впровадження значно сприяло взаємопов'язаному та динамічному характеру цифрової екосистеми.

1.2.2 API на базі бібліотек

API на базі бібліотек, також відомі як бібліотеки програмування або бібліотечні API, - це колекції заздалегідь написаних модулів коду, які надають розробникам набір функцій і процедур для використання у своїх програмних додатках.

Ці бібліотеки інкапсулюють складні функціональні можливості, полегшуючи програмістам реалізацію конкретних функцій без необхідності писати код з нуля.

Ось огляд ключових аспектів, пов'язаних з бібліотечними API:

- Визначення та призначення.
  Бібліотечний API - це набір функцій, процедур і класів, які упаковані разом для виконання певних завдань або надання певних можливостей. Ці бібліотеки призначені для повторного використання в різних проектах, що дозволяє розробникам використовувати існуючий код та економити час і зусилля.
- Типи бібліотек.
  - Бібліотеки функцій: Містять набори функцій, які виконують певні операції. Розробники можуть викликати ці функції для виконання

завдань без необхідності розуміти або переписувати основний код.

- Бібліотеки класів: Складаються з класів та об'єктів, які інкапсулюють дані та поведінку. Об'єктно-орієнтовані мови програмування часто використовують бібліотеки класів для створення багаторазових компонентів.

- Популярні приклади.
  - Стандартна бібліотека шаблонів (STL): у C++ STL надає набір шаблонних класів і функцій, включаючи контейнери (наприклад, вектори і карти) та алгоритми (наприклад, сортування і пошук).
  - Стандартна бібліотека Java: Java пропонує комплексну бібліотеку, яка включає структури даних, мережеві можливості, операції вводу/виводу та багато іншого.
  - Стандартна бібліотека Python: Стандартна бібліотека Python охоплює широкий спектр модулів, від маніпулювання даними та роботи з файлами до роботи в мережі та веб-розробки.
- Інтеграція в код.

  Розробники інтегрують бібліотечні API у свій код шляхом включення бібліотеки у свій проект і використання наданих функцій або класів. Це сприяє повторному використанню коду та зменшує надмірність.
- Переваги.
  - Ефективність: Бібліотеки економлять час розробки, надаючи готові рішення для типових завдань.
  - Узгодженість: Стандартизовані бібліотеки забезпечують узгодженість реалізації коду в різних проектах.
  - Надійність: Добре зарекомендували себе бібліотеки проходять тестування і доопрацювання, що сприяє їх надійності.

- Абстрагування: Бібліотеки абстрагують складні функціональні можливості, дозволяючи розробникам зосередитися на більш високорівневих аспектах своїх додатків.

- Документація.

  Якісна документація має вирішальне значення для бібліотечних API. Розробники покладаються на документацію, щоб зрозуміти, як використовувати функції або класи, які параметри вони приймають і які результати очікуються. Чітка документація підвищує зручність використання бібліотеки.

- Залежності та керування версіями.

  Розробники повинні ретельно керувати залежностями, оскільки зміни в бібліотеці можуть вплинути на функціональність коду. Керування версіями необхідне для забезпечення зворотної сумісності та надання розробникам можливості вибрати версію, яка відповідає їхнім потребам.

- Бібліотеки з відкритим кодом.

  Багато бібліотек мають відкритий вихідний код, що дозволяє спільноті долучатися до їх створення, вдосконалення та налаштування. Такий спільний підхід сприяє інноваціям та відчуттю спільноти серед розробників.

- Платформна сумісність.

  Бібліотеки можуть бути

  платформно-специфічними або кросплатформними. Міркування щодо сумісності платформ є дуже важливими, особливо у проектах, орієнтованих на декілька операційних систем або середовищ.

- Проблеми.

  Хоча бібліотечні API мають багато переваг, вони можуть бути пов'язані з такими проблемами, як тривалість навчання для нових бібліотек,

потенційні конфлікти з іншими бібліотеками та необхідність постійного оновлення останніх версій і патчів безпеки.

Наприклад, у мові програмування Java, якщо розробник бажає отримати від користувача інформацію у програмах, спрямованих на текстові операції, він повинен використовувати клас "Scanner". Для цього необхідно імпортувати бібліотеку "java.util.Scanner". На рис.1.2 представлено приклад використання API, яке дозволяє взаємодіяти з бібліотеками в мові програмування Java.

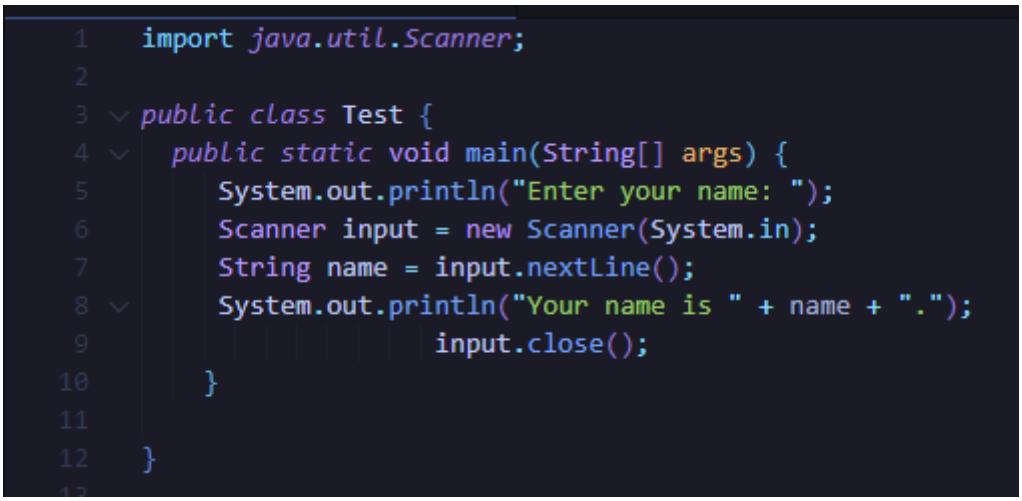


Рисунок 1.2 – Взаємодія в мові Java

Отже, бібліотечні API є безцінними інструментами для розробників, що дозволяють використовувати існуючий код і спростити процес розробки. З розвитком технологій наявність надійних і добре задокументованих бібліотек залишається рушійною силою ефективної та масштабованої розробки програмного забезпечення.

1.2.3 API на рівні операційної системи

API на рівні операційної системи, які часто називають системними API або API ядра, - це набори функцій і процедур, що надаються операційною системою для забезпечення взаємодії між програмами та базовим обладнанням.

Ці API слугують інтерфейсом, за допомогою якого програмне забезпечення може отримати доступ до різних служб операційної системи та використовувати їх,

зокрема управління файлами, керування процесами, розподіл пам'яті та взаємодію з апаратним забезпеченням.

Ось огляд ключових аспектів, пов'язаних з API на рівні операційної системи:

- Визначення та призначення.
  API на рівні операційної системи забезпечують зв'язок між додатками та ядром операційної системи. Вони надають низку сервісів і функцій, які програми можуть використовувати для виконання завдань на низькому рівні, взаємодіючи безпосередньо з основними компонентами операційної системи.
- Типи API на рівні операційної системи.
  - API файлової системи: Дозволяють програмам виконувати операції над файлами і каталогами, такі як читання, запис, створення і видалення.
  - API керування процесами: Дозволяють програмам створювати, завершувати та керувати процесами, а також контролювати міжпроцесну взаємодію.
  - API управління пам'яттю: Надають функції для виділення та звільнення пам'яті, керування віртуальною пам'яттю та захисту пам'яті.
  - Пристрої та апаратні API: Полегшують взаємодію з апаратними пристроями, включаючи операції вводу/виводу, конфігурацію пристроїв та взаємодію з драйверами.
  - Мережеві API: Підтримують мережеві можливості, включаючи програмування сокетів, обробку протоколів і конфігурацію мережі.
- Платформо-специфічні API.
  Різні операційні системи мають власні набори API. Наприклад:

  - Windows API (WinAPI): Використовується в операційних системах Microsoft Windows.
  - POSIX API: Набір стандартів, що включає API для Unix та Unix-подібних операційних систем.
  - macOS API: Використовується в операційній системі macOS від Apple.

- Доступ до системних ресурсів.

  API на рівні операційної системи надають додаткам доступ до критично важливих системних ресурсів, які зазвичай обмежені з міркувань безпеки та стабільності. Це включає доступ до привілейованих інструкцій, структур даних ядра та апаратних інтерфейсів.

- Низькорівневі операції.

  Ці API забезпечують низькорівневу функціональність, дозволяючи розробникам виконувати операції, які безпосередньо впливають на систему. Це контрастує з високорівневими API, які абстрагуються від цих деталей для зручності використання.

- Паралелізм і синхронізація.

  API на рівні операційної системи включають механізми керування паралельним виконанням, синхронізації між потоками або процесами та роботи зі спільними ресурсами для запобігання гонки та забезпечення цілісності даних.

- Обробка переривань.

  API на рівні операційної системи полегшують обробку апаратних переривань, дозволяючи програмам реагувати на такі події, як введення даних користувачем, апаратні збої або події таймера.

- Міркування щодо безпеки.

  Через критичну природу операцій на рівні системи, безпека є першочерговим завданням. Доступ до певних API може бути обмежено,

а для запобігання несанкціонованому доступу впроваджено належні механізми автентифікації та авторизації.

- Розробка драйверів.

  API на рівні операційної системи є важливими для розробки драйверів. Драйвери пристроїв, які є програмними компонентами, що забезпечують зв'язок між операційною системою та апаратними пристроями, взаємодіють з ядром через ці API.

- Системні виклики.

  Системні виклики є фундаментальною частиною API на рівні операційної системи. Коли програма робить системний виклик, вона переходить з режиму користувача в режим ядра, щоб виконати привілейовану операцію, викликаючи відповідну функціональність API.

Отже, API на рівні операційної системи є критично важливим шаром у стеку програмного забезпечення, що дозволяє програмам взаємодіяти з базовою операційною системою та апаратним забезпеченням. Вони надають інструменти та сервіси, необхідні розробникам для виконання низькорівневих операцій, керування системними ресурсами та створення надійного, ефективного і специфічного для конкретної платформи програмного забезпечення. Розуміння та використання цих API є важливими навичками для системних програмістів та розробників, які працюють над завданнями, що потребують глибокої інтеграції з операційною системою.

1.3 Використання API в різних сферах

Технологія API має широке застосування в різних сферах програмування. API слугують мостом між різними програмними та апаратними компонентами, дозволяючи їм спілкуватися, обмінюватися даними та співпрацювати не маючи обмежень. На рис.1.3 зображено приклад взаємодії API з різними сферами життя.

Рисунок 1.3 – Взаємодія API з різними сферами життя

Ми використовуємо цю технологію кожен день навіть не знаючи про неї. Кожен день людина дивиться прогноз погоди на своєму телефоні, використовує онлайн карти для побудови маршрутів, розплачується в магазині не підозрюючи що для роботи всього цього використовують API.

Використання API в погодних сервісах є найрозповсюдженішим прикладом з повсякденного життя. Ці API надають можливість розробникам інтегрувати функції пов’язані з погодою в свої веб-сайти, мобільні додатки та інше. Такі організації як OpenWeatherMap надають розробникам доступ до свого API,а ті в свою чергу використовують його для отримання поточних даних про погодні умови, прогнози та іншу метеорологічну інформацію.

Погодні API зазвичай працюють у взаємозв’язку з геолокаційними сервісами. Поєднуючи дані отримані від погодних API з місцезнаходженням пристрою можна отримати локалізовану та персоналізовану інформацію про погоду на основі місцезнаходження користувача.

Підключивши до цього додатку ще й Push Notification APIs додаток зможе використовувати сповіщення для інформування користувача про складні погодні умови, такі як шторми, урагани або екстремальні температури. Користувачі зможуть своєчасно отримувати важливу інформацію на основі їхнього місцезнаходження навіть тоді, коли додаток не використовується.

Але це все використання цього API в дрібних цілях. Авіакомпанії та транспортні служби використовують ці API для аналізу та прийняття обґрунтованих рішень щодо розкладу рейсів, планування маршрутів та загальної безпеки.

Таким чином погодні API відіграють важливу роль у багатьох галузях, надаючи цінну інформацію для прийняття важливих рішень. Розробники використовують ці API для створення багатофункціональних додатків, які покращують досвід користувача та сприяють розвитку різних галузей, що покладаються на погодну інформацію.

Проте це тільки одна з багатьох сфер використання API з усіх існуючих. API слугують мостом для взаємодії різних компонентів і ось огляд різних сфер, де ця технологія широко використовується в програмування:

- Веб-розробка. Веб-розробники постійно використовують API для інтеграції сторонніх компонентів/сервісів в свої проекти, асинхронного отримання та оновлення даних і створення динамічних та інтерактивних веб-додатків. Популярними прикладами такої взаємодії є сайти прогнозу погоди, інтеграції з соціальними мережами, картографічними та платіжними сервісами. Пізніше ця сфера буде розглянута більш детальніше.
- Розробка мобільних додатків. Не менш важливою є ця сфера, у ній API має вирішальне значення для доступу до функцій пристрою та внутрішніми службами девайсу. Інтеграція з цим API дозволяє використовувати такі функції, як push-сповіщення, визначати

місцезнаходження пристрою, робити покупки в додатках та багато іншого. На рис.1.4 зображено взаємодія API з різними додатками в мобільному пристрої.

Рисунок 1.4 – Взаємодія API з додатками в мобільних пристроях

- Хмарні обчислення. Постачальники хмарних послуг пропонують використання їх API для управління та взаємодії з ресурсами які вони надають. Розробники використовують ці API для надання серверів, зберігання даних та розгортання додатків на таких платформах як Azure та Google Cloud. На рис.1.5 зображено приклад взаємодії цих API з хмарними сервісами.

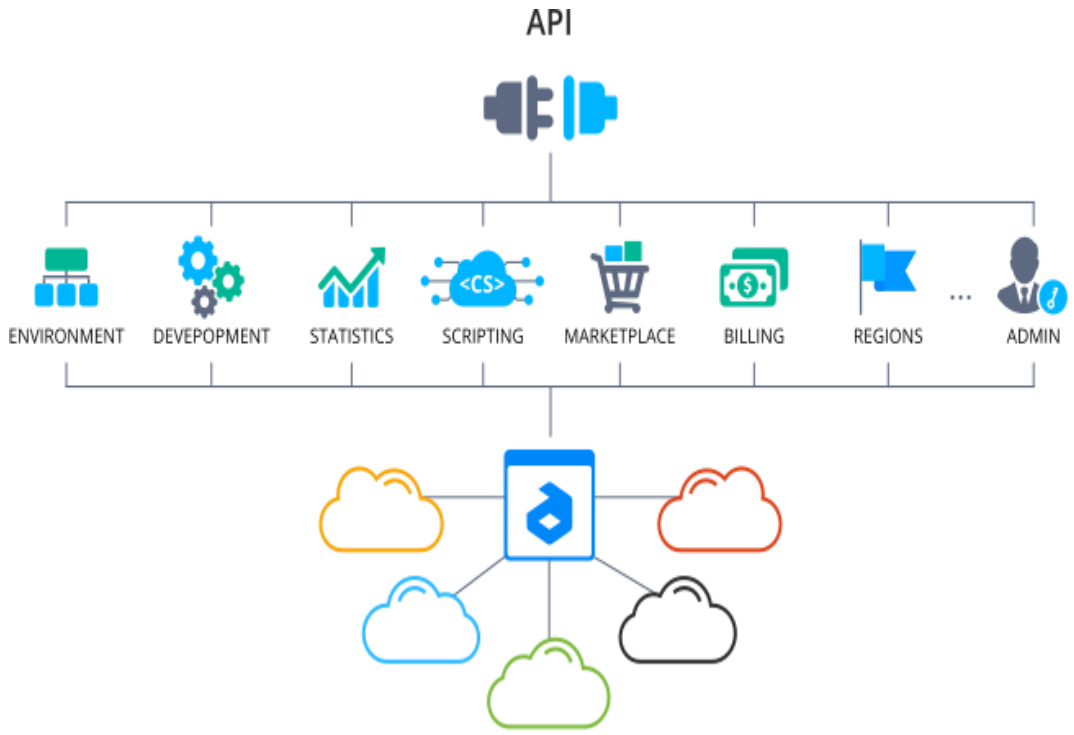


Рисунок 1.5 – Взаємодія API з Cloud Services

- Архітектура мікросервісів. В цій архітектурі різні компоненти взаємодіють між собою використовуючи API. Кожен мікросервіс надає API, які забезпечують комунікацію та обмін даними, сприяючи модульному підходу до розробки програмного забезпечення.
- IoT (Internet of Things). У цих додатках API відіграють важливу роль у підключенні пристроїв і датчиків. Ці API забезпечують зв’язок між ними, обробку даних та управління всією екосистемою IoT.
- Платформи електронної комерції використовують платіжні API для безпечної обробки транзакцій. Вони дозволяють компаніям приймати різні способи оплати, забезпечуючи безперебійний і безпечний процес оформлення та оплати замовлення.
- API сервісів машинного навчання та штучного інтелекту, наприклад, OpenAI або Google Cloud Ai дозволяють розробникам інтегрувати попередньо навчені моделі штучного інтелекту в свої додатки, не потребуючи для цього створення нової системи та її навчання.

- Також одним з популярних використанням API є системи управління контентом. Ці системи надають API для інтеграції з зовнішніми сервісами, імпорту та експорту контенту, а також автоматизації робочих процесів публікації.

Ці різноманітні використання технології API підкреслюють її важливість та універсальність у сучасному програмування. Вона дозволяє розробникам використовувати існуючи сервіси, створювати інноваційні рішення в різних галузях.

Вказані нижче приклади використання API у різноманітних областях, що ілюструють їхню роль у різних сегментах:

- API для роботи з погодою.

  OpenWeatherMapApi надає комплексний погодний API, який дозволяє розробникам отримати доступ до поточних погодних умов, прогнозів та історичних погодних даних для будь-якого місця.

  Одне з найпопулярніших використання цього API це додаток для планування подорожей, який відображає поточну погоду, тижневий прогноз погоди та історичні тенденції погоди для вибраних напрямків. На рис.1.6 зображено приклад запиту до цього API.

```
1  # Example API Request
2  https://api.openweathermap.org/data/2.5/weather?q=CityName&appid=YourApiKey
3
```

Рисунок 1.6 – Запит до OpenWeatherMapApi

- Фінансові API.

  Alpha Vantage API – це API даних фінансового ринку, який надає інформацію про фондові ринки в реальному часі та історичні дані, курси валют і дані про криптовалюти.

  Один з прикладів використання цього API – це мобільний додаток для ентузіастів фондового ринку, який відображає поточні ціні на акції,

історичні дані та технічні індикатори. На рис.1.7 зображено приклад запиту до цього API.

```
# Example API Request
https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=AAPL&apikey=YourApiKey
```

Рисунок 1.7 – Запит до Alpha Vantage API

- Картографічні API.

  Google Maps API дозволяє розробникам інтегрувати карти, геолокацію та сервіси на основі місцезнаходження у веб та мобільні додатки.

  Один з прикладів використання цього API – це картографічний додаток, який використовує Google Maps API для відображення інтерактивних карт, пошуку цікавих місць для відвідування та планування маршрутів. Популярний додаток який використовує це API – Google Maps. На рис.1.8 зображено підключення до цього API.

```
<!-- Example API Script Embedding -->
<script async defer src="https://maps.googleapis.com/maps/api/js?key=YourApiKey&callback=initMap"></script>
```

Рисунок 1.8 – Підключення Google Maps API до проекту

- API соціальних мереж.

  Twitter API дозволяє розробникам отримувати доступ до даних Twitter, включаючи твіти, профілі користувачів та тренди.

  Приклад використання цього Api – це мобільний додаток, який інтегрує Twitter API для відображення часової шкали користувача, публікації твітів та відстеження тенденцій. На рис.1.9 зображено приклад запиту до цього API.

```
# Example API Request
https://api.twitter.com/2/tweets?ids=12345,67890&expansions=author_id
```

Рисунок 1.9 – Запит до Twitter API

- API електронної комерції.

Shopify надає API для інтеграції з електронною комерцією, що дозволяє розробникам керувати продуктами, замовленнями та даними клієнтів. Один з найпопулярніших прикладів використання цього API – це інтернет магазин, який використовує його для синхронізації списків товарів, управління замовленнями та оновлення записів у базі даних. На рис.1.10 зображено приклад запиту до цього API.

```
# Example API Request
https://store.myshopify.com/admin/api/2023-01/products.json
```

Рисунок 1.10 – Запит до Shopify API

- API новин.

  News API надає простий HTTP REST API для доступу до поточних та історичних новин з різних джерел.

  Цей API зазвичай використовується для створення веб додатків який виводить статті новин на основі вподобань користувачів та популярних тем. На рис.1.11 зображено приклад запиту до цього API.

```
# Example API Request
https://newsapi.org/v2/top-headlines?country=us&apiKey=YourApiKey
```

Рисунок 1.11 – Запит до News API

- API машинного навчання.

  IBM Watson пропонує Api для обробки природної мови, перекладу та візуального розпізнавання.

  Приклад його використання – це додаток який інтегрує API перекладу для надання послуг перекладу в режимі реального часу. На рис.1.12 зображено приклад запиту до цього API.

```
# Example API Request
https://api.us-south.language-translator.watson.cloud.ibm.com/instances/YourInstanceId/v3/translate
```

Рисунок 1.12 – Запит до IBM Watson API

Ці приклади демонструють різноманітність API у різних сферах, підкреслюючи їхню роль у розширенні функціональності та інтеграції у додатки в Інтернеті та мобільних пристроях, а також використанні в повсякденному житті.

Більшість API з наведених вгорі прикладів мають схожі за синтаксисом підключення, проте це лише підключення до вашого проекту. Весь функціонал API розкривається в залежності від того який саме API ви використовуєте й від ваших запитів в межах його функцій. Якщо ви запросите у OpenWeatherMapApi дані про курси валют звичайно він ван нічого не відповість. Проте якщо ви будете запитувати його про погоду в будь якому місті він може видати вам повний спектр цих даних, від прогнозу в даний момент часу до тижневого прогнозу з урахування швидкості вітру, щільності осадків тощо.

В цьому розділі було проаналізовано багатогранний світ технології API, що охоплює різні сфери та застосування в програмуванні. Технології API є неймовірно універсальними і слугують для комунікації та інтеграції між різними програмними компонентами. Вони забезпечують безперешкодну взаємодію між різними платформами та системами. API знаходить застосування в різних середовищах, зокрема в Інтернеті, мобільних пристроях, на підприємствах, в Інтернеті речей, хмарі та настільних комп’ютерах. У кожному середовищі API відіграють певну роль, забезпечуючи зв’язок між пристроями, доступ до зовнішніх сервісів або полегшують обмін даними. Вивчення технології API передбачає заглиблення в офіційну документацію, книги про конкретні мови програмування та фреймворки. Вибір літератури залежить від конкретної мови на якій ви будете розроблювати проект та API, які будуть використовуватися в ньому.

Таким чином, API є невід’ємно частиною сучасної розробки програмного забезпечення, сприяючи інноваціям у різних сферах. Незалежно від того, чи це веб або мобільні, корпоративні або спеціалізовані додатки, використання API має важливе значення для створення надійних та багатофункціональних і взаємопов’язаних систем.

2 Веб-сайт та його особливості

2.1 Концепція та структура веб-сайтів

Веб-сайт – це сукупність пов’язаних між собою веб-сторінок і мультимедійного контенту, які зазвичай ідентифікуються спільним доменним ім’ям і публікуються хоча б на одному веб-сервері. Ці веб-сторінки можуть містити текст, зображення, відео та інші типи контенту, і вони зазвичай пов’язані між собою за допомогою гіперпосилань. Доступ до веб-сайтів здійснюється через Інтернет або приватну мережу за допомогою веб-браузера.

Ось деякі ключові компоненти пов’язані з веб-сайтами:

- Веб-сторінка – це документ, зазвичай написаний мовою програмування HTML (мова розмітки гіпертексту), який доступний через веб-браузер. Вона може містити різні типи контенту, такі як текст, зображення, мультимедійні елементи та інтерактивні функції.
  На рис.2.1 зображено приклад веб-сторінки.

Рисунок 2.1 – Веб-сторінка Spotify

- Доменне ім’я – це читабельна для людини адреса, яку користувачі вводять в адресний рядок браузера, щоб отримати доступ до веб-сайту.

Це зручний спосіб знайти та ідентифікувати певну веб-сторінку або набір веб-сторінок. На рис.2.2 зображено приклад доменного імені веб-сайту.

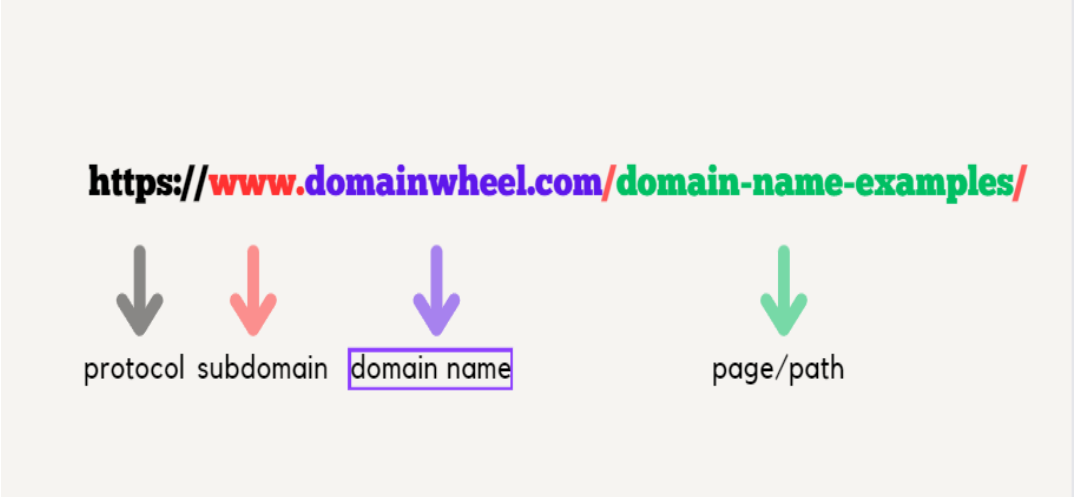


Рисунок 2.2 – Доменне ім'я

- Веб-сервер – це комп’ютер або програмне забезпечення, яке зберігає і доставляє веб-сторінки користувачам через Інтернет. Коли користувач запитує веб-сторінку, веб сервер обробляє запит і надсилає запитувану сторінку до браузера користувача. На рис.2.3 зображено як браузер взаємодіє з веб-сервером для отримання запитуваної веб-сторінки.

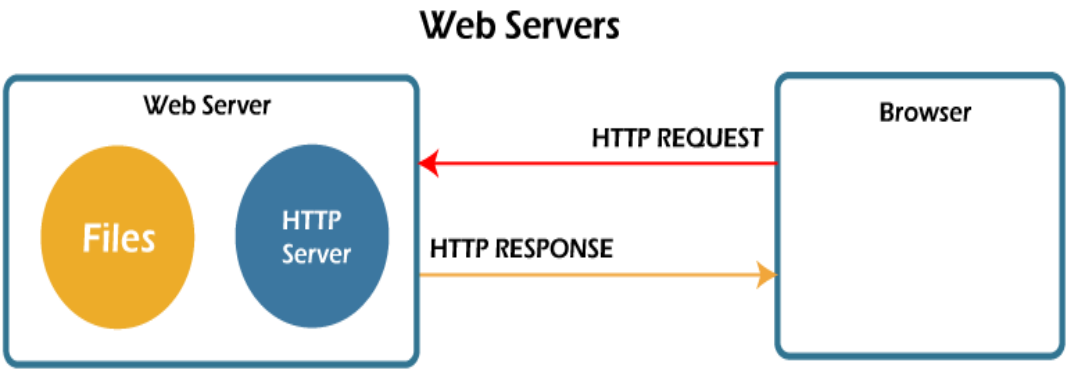


Рисунок 2.3 – Взаємодія браузера з веб-сервером

- Гіперпосилання, або просто посилання – це елементи на веб-сторінці, які дозволяють користувачам переходити на інші сторінки або ресурси. Натискання на гіперпосилання перенаправляє користувача в інше місце на тому ж веб-сайті або на зовсім інший веб-сайт. На рис.2.4 зображено

синтаксис гіперпосилання в коді веб-сайту. Програміст використовує мову програмування HTML для створення гіперпосилання, задає куди саме воно буде посилатися та як це посилання буде виглядати з боку користувачів сайту.

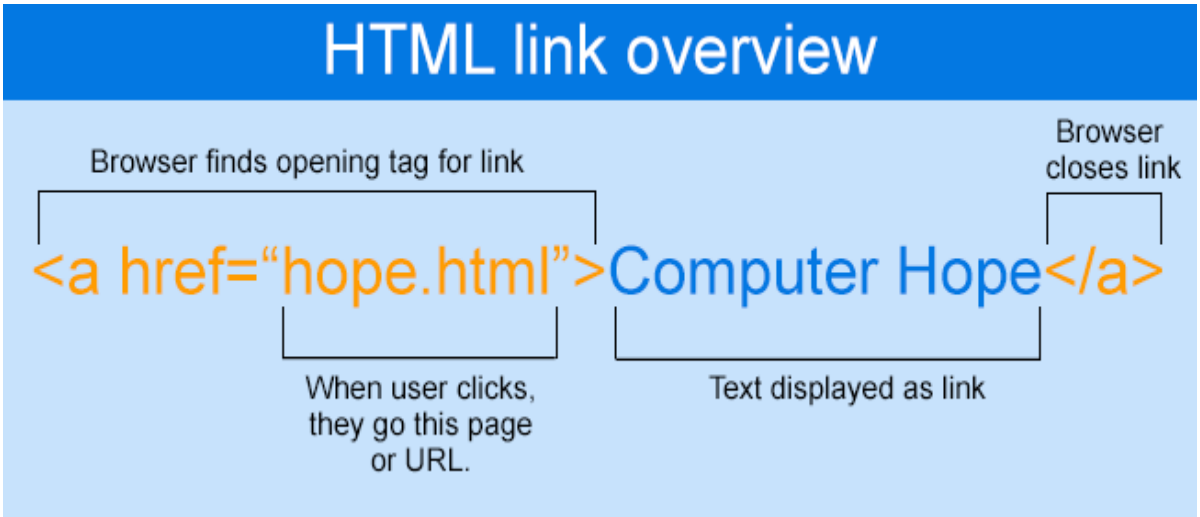


Рисунок 2.4 – Приклад синтаксису гіперпосилання в коді веб-сайту

- Веб-браузери – це програмне забезпечення, яке дозволяє користувачам отримувати доступ і переглядати веб-сторінки в Інтернеті. Прикладами веб-браузерів є Google Chrome, Mozilla Firefox, Microsoft Edge та Safari.
- Веб-розробка передбачає створення та підтримку веб-сайтів. Вона включає такі види діяльності, як веб-дизайн, кодування, створення контенту та управління сервером.
- Веб-сайти можуть служити різним цілям, включаючи інформаційні, освітні, розважальні, електронну комерцію, соціальні мережі тощо. Структура та дизайн веб-сайту залежать від його призначення та вподобань його творців.

Веб-сайти стали невід'ємною частиною нашого повсякденного життя, надаючи інформацію, послуги та розваги. Інтернет – це величезна мережа взаємопов'язаних веб-сайтів, що формують основу сучасного цифрового ландшафту. З розвитком технологій можливості та функції веб-сайтів

розвиваються, формуючи нові способи, за допомогою яких ми взаємодіємо з контентом в Інтернеті.

2.2 Вивчення категорій веб-сайтів

Веб-сайти, будівельні блоки цифрового будинку, який ми звемо Інтернет. Вони не є універсальними об’єктами, навпаки, вони ретельно розробляються і класифікуються для виконання певних цілей, задовольняючи постійно зростаючі потреби широкої і різноманітною аудиторії. Інтернет, подібно до величезною і динамічної екосистеми, містить безліч веб-сайтів, кожен з яких спрямований для виконання певних функцій і задоволення унікальних потреб користувачів.

Веб-сайти бувають різних категорій, кожна з яких призначена для певних цілей і задовольняє різні потреби користувачів.

Категоризація веб-сайтів схожа на організацію бібліотеки, де в різних полицях зберігаються різні жанри знань. У віртуальному світі ці категорії проявляються у вигляді спеціалізованих доменів, кожен з яких має власний набір функцій, контенту та взаємодії з користувачем. Цифрова екосистема процвітає завдяки такому різноманіттю – від простого сайту-візитки до інформаційно насичених освітніх веб-сайтів.

Основним принципом, що зумовлює таке різноманіття сайтів, є те, що користувачі виходять в Інтернет з багатогранними намірами. Сайти Інтернет-магазинів, наприклад, діють як віртуальні вітрини, демонструючи продукти та послуги потенційним клієнтам. З іншого боку, корпоративні веб-сайти виходять за рамки простої демонстрації вітрин, забезпечуючи широке представлення ідентичності, історії та цінностей компанії.

Освітні веб-сайти, використовуючи можливості Інтернету, надають можливості для навчання ширшій аудиторії, руйнуючи географічні бар’єри і надають доступ до своїх ресурсів з будь якої частини світу.

Поява соціальних мереж змінила способи спілкування та зв’язку між людьми, сплітаючи складну павутину віртуальних стосунків. Блоги пропонують персоналізований простір для людей та організацій, де вони можуть ділитися своїми ідеями, думками та історіями, роблячи свій внесок у різноманітний світ онлайн-контенту.

Паралельно з цим, веб-додатки виходять за традиційні межі статичних веб-сторінок, пропонуючи динамічний та інтерактивний досвід. Ці додатки, від прогнозів погоди до інструментів спільного управління проектами, демонструють універсальність веб-технологій у задоволенні потреб і посиленні залучення користувачів.

Подорожуючи цим цифровим світом користувачі стикаються з урядовими веб-сайтами, що надають офіційну інформацію та послуги, новинними веб-сайтами, що надають їх в режимі реального часу, та веб-сайтами-портфоліо, що демонструють творчі досягнення окремих людей.

Ось кілька найпоширеніших категорій веб-сайтів:

- Бізнес-сайти. Вони створюють для просування та представлення певного бізнесу або компанії в інтернеті. Зазвичай вони містять інформацію про продукти або послуги компанії, контактні дані, а також можуть мати такі розділи, як сторінка «Про нас», відгуки та блок.
- Корпоративні веб-сайти. Такі веб-сайти схожі на бізнес-сайти, але зазвичай асоціюються з великими корпораціями. Вони слугують обличчям юридичної особи. Ці веб-сайти часто містять інформацію про історію компанії, її місію, бачення, керівний склад, фінансові звіти та ініціативи корпоративної соціальної відповідальності.
- Веб-сайти електронної комерції сприяють здійсненню онлайн-транзакцій з купівлі-продажу товарів чи послуг. Вони включають списки товарів, кошики для покупок, безпечні платіжні сервіси, а іноді

й відгуки клієнтів. Прикладом таких веб-сайтів являються Amazon та eBay.

- Освітні веб-сайти надають інформацію та ресурси для навчання та академічних цілей. Вони можуть пропонувати онлайн-курси, лекції, навчальні матеріали та інтерактивні інструменти. Прикладами таких освітніх веб-сайтів є Udemy та Wikipedia.
- Новинні веб-сайти висвітлюють поточні події, статті та мультимедійний контент, пов’язаний з місцевими, національними чи міжнародними новинами. Вони включають розділи на різні теми, такі як останні новини, авторські статті, а іноді й контент, створений користувачами. Прикладами таких веб-сайтів є BBC News, CNN та The New York Times.
- Веб-сайти соціальних мереж об’єднують людей, дозволяючи їм створювати профілі, обмінюватися контентом та взаємодіяти з іншими в Інтернеті. Користувачі можуть спілкуватися з друзями, публікувати оновлення, ділитися фотографіями чи відео та брати участь у дискусіях. Прикладами таких мереж є Twitter та Facebook.
- Веб-сайти блоги – це платформи для приватних осіб або організацій, на яких вони можуть ділитися регулярним контентом на певні теми. Зазвичай вони містять статті, дописи і можуть дозволяти читацькі коментарі.
- Веб-додатки – це інтерактивні програми або інструменти, доступ до яких здійснюється через веб-браузер і які надають певні функціональні можливості. Прикладами є програми для прогнозування погоди, онлайн календарі, інструменти для управління проектами та платформи для спільної роботи, такі як Google Workspace.
- Веб-сайти-портфоліо – демонструють роботу, навички та досягнення окремих осіб, таких як художники, дизайнери, фотографи та письменники. Вони включають галереї, описи проектів та контактну інформацію.

- Урядові веб-сайти надають офіційну інформацію та послуги, пов’язані з державними установами та державними службами. Вони можуть містити інформацію про закони, нормативні акти, державні послуги та контактні дані. Прикладом таких веб-сайтів є mon.gov.ua

Ці категорії представляють широкий спектр різноманітної та динамічної природи веб-сайтів в Інтернеті, відображаючи широкий спектр інформації та послуг доступних користувачам в Інтернеті. Кожна категорія представляє певну грань Інтернету, пристосовану для збагачення, інформування, розваг та полегшення взаємодії. На рис.2.5 зображено графічне представлення найпопулярніших категорій.

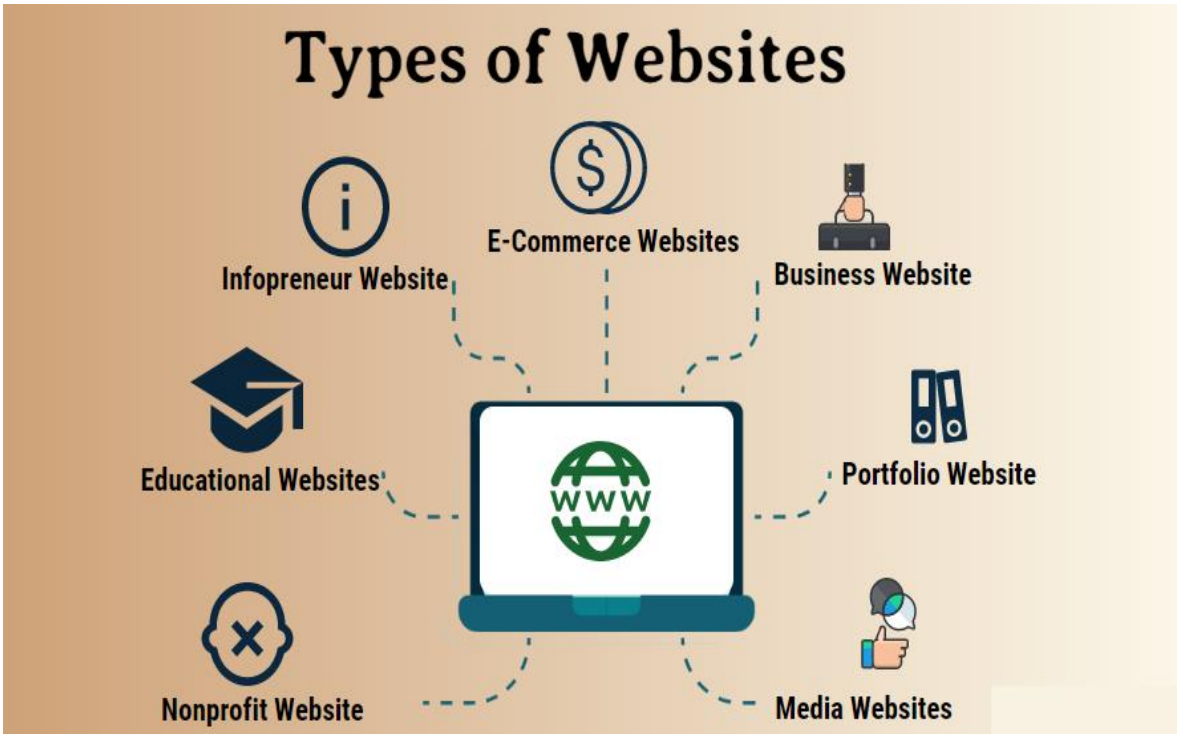


Рисунок 2.5 – Категорії веб-сайтів.

По суті, незліченна кількість категорій веб-сайтів підкреслює адаптивність та здатність цифрової сфери відповідати різноманітним потребам і вподобанням її користувачів. Інтернет у своїй безмежності віддзеркалює складність людського досвіду пропонуючи віртуальний простір, де окремі особо, компанії та організації

можуть знайти свою нішу, зв’язатися зі своєю аудиторією та зробити свій внесок у постійно еволюціонуючий віртуальний світ.

2.3 Архітектура веб-сайту

Архітектура веб-сайту – це структурна організація та конфігурація веб-сайту, яка визначає його складові частини, їх взаємозв’язки та організацію інформації. Це концептуальний план, який визначає, як будуть організовані та взаємодіяти між собою різні елементи веб-проекту. Архітектура сайту враховує не лише вигляд та оформлення сторінок, а й структури баз даних, логіку взаємодії, розміщення контенту, систему навігації та інші аспекти.

Ефективна архітектура сайту сприяє зручності використання для користувачів, полегшує процеси розробки та підтримки, а також сприяє оптимізації швидкодії та продуктивності веб-сайту. Архітектура веб-сайту може бути різною залежно від типу сайту (наприклад, інтернет-магазин, блог, соціальна мережа) та його функціональності.

Архітектура веб-сайту включає в себе ряд ключових аспектів, що визначають структуру та організацію веб-проекту:

- Структура інформації визначає, як інформація розподіляється по різним частинам сайту, як організовані меню, категорії, підрозділи.
  На рис.2.6 зображено приклад структуризації веб-сайту на різні категорії та підкатегорії в залежності від їхнього контенту.

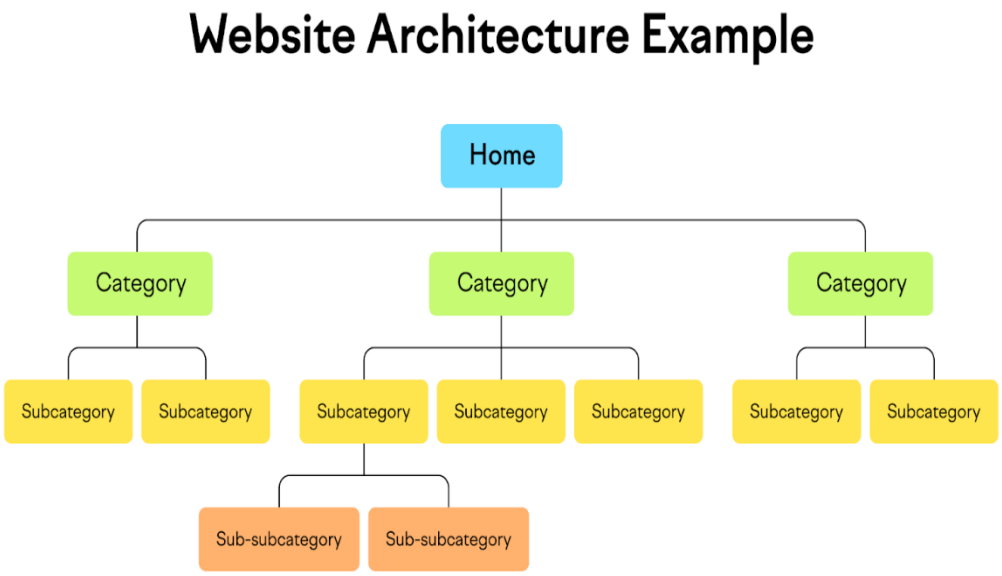


Рисунок 2.6 – Структура веб-сайту

- Навігація визначає систему переходів між різними сторінками та розділами веб-сайту. Ефективна навігація сприяє зручності користування та пошуку інформації.
- Вигляд та оформлення описує, як виглядає веб-сайт, включаючи дизайн, колірну палітру, використання мультимедійних елементів, шрифтів та інших аспектів естетики та візуальної привабливості.
- Функціональність встановлює, які функції та можливості надаються користувачам на веб-сайті. Це може включати реєстрацію, авторизацію, фільтрацію контенту, можливості пошуку, коментування тощо.
- Технічна архітектура. Специфікація щодо технічних рішень, використовуваних для розробки та підтримки веб-сайту. Це може включати набір мов програмування, фреймворків, баз даних, серверів та інших технічних засобів.
- Безпека. Заходи, що призначені для захисту веб-сайту від потенційних загроз, таких як атаки хакерів, вразливості веб-додатків, зберігання конфіденційної інформації тощо.

- Мобільна адаптація розглядає відповіді веб-сайту на різні типи пристроїв, забезпечуючи його коректне відображення та функціональність на комп’ютерах, планшетах та смартфонах.

Архітектура веб-сайту є комплексним поняття, що охоплює різноманітні аспекти для успішного розроблення, запуску та експлуатації веб-проекту. Ефективна архітектура веб-сайту допомагає забезпечити зручність користування, покращує продуктивність розробки та сприяє досягненню бізнес-цілей веб-проекту. Важливо враховувати потреби користувачів, особливості ринку та визначити оптимальний підхід до побудови архітектури веб-сайту для досягнення успіху. Її врахування з ранніх стадій розробки дозволяє створити ефективний та добре організований веб-сайт.

Також не менш важливим є багатомовність веб-сайту. Якщо веб-сайт спрямований на аудиторію з різних країн, архітектура повинна враховувати можливість підтримки декількох мов, локалізацію та адаптацію контенту.

На мою думку однією з найважливіших компонентів архітектури веб-сайту є організація системи зберігання та управління контентом, а також системи оптимізації продуктивності. Організація системи управління контентом (CMS), яка наддасть змогу легко додавати, редагувати та видаляти контент на веб-сайті. Сюди можуть входити такі елементи, як управління версіями контенту, обробка медіа-файлів та систематизація інформації. Іншим важливим компонентом є система оптимізації продуктивності. Використання стратегій кешування, мінімізація завантаженості ресурсів, оптимізація зображень та інші техніки для підвищення швидкодії та відповідності сайту покращують досвід користування з цим веб-сайтом. Користувачам набагато приємніше користуватися добре систематизованим та оптимізованим веб-сайтом, ані ж незрозумілим набором веб-сторінок, які не мають належності структури та ще й завантажуються по кілька хвилин.

Краще за все розглядати архітектуру веб-сайту на основі інтернет-магазинів. Найважливішим рішення для такого веб-сайту це те як ви структуруєте свій контент і навігацію. Це значною мірою впливає на досвід користувачів і на те, наскільки успішно ви будете просуватися в пошукових системах. Необхідно добре знати куди ви кладете веб-сторінки і чому. Краще всього визначитися з цим до початку запуску веб-сайту, оскільки процес реорганізації веб-сайту займає сотні годин розробки.

Веб-сайти інтернет-магазинів зазвичай складаються з чотирьох рівнів. На рис.2.7 наглядно зображені ці рівні. Головною сторінкою веб-сайту є «Index page». «Index page» повинна містити посилання на всі ваші основні категорії у верхньому або боковому меню, в залежності від побажання замовника, це як карта у входу у торгівельний центр. Відвідувач повинен одразу бачити, куди натиснути аби знайти потрібний розділ.

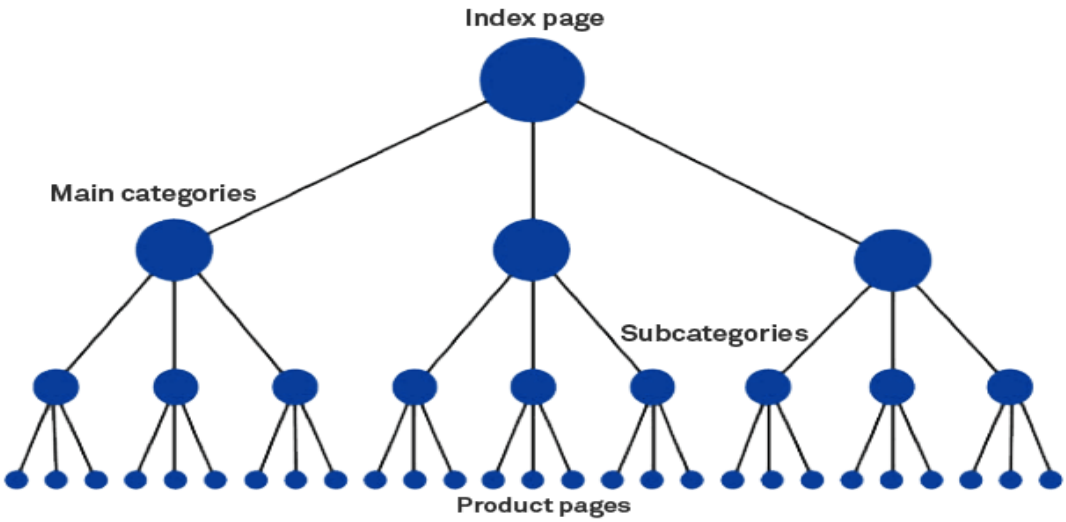


Рисунок 2.7 – Структура веб-сайту інтернет-магазину

З точки зору пошукових систем, чим менше внутрішніх посилань, тим краще. Якщо ви посилаєтесь, скажімо на 100 сторінок з «Index page», ви повідомляєте Google, що у вас ці 100 сторінок є однаково важливими. Однак це помилка, необхідно структурувати все це, розподілити ці сторінки по категоріям, та підкатегоріям і робити посилання з «Index page» вже на них.

Основні категорії ніколи не потребують пояснення, відвідувач повинен одразу розуміти, що він знайде в кожній з них. Крім того, більшість контенту

повинно відповідати категорії в якій він знаходиться. Якщо це не так,, вам варто переглянути структуру вашого веб-сайту. Основні категорії є статичні і не змінюються майже на всьому життєвому шляху веб-сайту. Гарне правило якого слід дотримуватися: кожна категорія повинна мати не більше 10 підкатегорій, аби користувач було легко знаходити необхідний їм матеріал на вашому веб-сайті.

Пошукова система може сканувати ваш веб-сайт у структурованому вигляді. Вона розуміє, які категорії є найважливішими і які підкатегорії пов'язані між собою. Замість того, щоб мати розкидану по всьому інформацію, ви можете створити добре зрозумілу структуру навколо важливих тем.

Для внутрішнього пошуку на сайті добре зроблена структура полегшує результативність його роботи. Подібно до того, як Google використовує структуру вашого веб-сайту, щоб визначити найважливіше в ньому, так само робить і внутрішня пошукова система. За винятком того, що він не дивиться на ваш сайт, він покладається на структуру даних вашого каталогу, тобто всі дані, що лежать в основі вашого веб-сайту – назви, опис, ієрархія категорій тощо. Якщо все зроблено правильно, пошук на сайті буде використовувати вашу категоризацію для покращення результатів. Він також може використовувати категоризацію для перенаправлення користувачів на сторінки певних категорій безпосередньо в автозавершенні пошуку. Наприклад, якщо хтось шукає «Scott» на сайті спортивного ритейлера, можуть бути показані такі категорії, як «Scott» у велосипедах, взутті, одязі, а також найпопулярніші товарі з цим пошуковим запитом у своїй назві.

2.4 Вибір мов програмування для розробки веб-сайту

Після того як архітектура веб-сайту була візуально розроблена, необхідно її відтворити на практиці. Розробнику необхідно буде визначитися за допомогою чого й як саме він буде створювати цей веб-сайт.

Існують різні методи для створення веб-сайтів, і вибір часто залежить від технічних знань, можливостей та вимог і вподобань замовника. Ось кілька методів які можуть буди обрані для виконання такого проекту:

- Написання веб-сайту з нуля. Це розуміє під собою повне написання всього веб-сайту використовуючи, наприклад, такі мови як HTML, CSS, JavaScript тощо. Переваги такого методу є повний контроль над дизайном та функціоналом веб-сайту, проте це займає багато часу та вимагає навичок кодування.
- В противагу методу написання веб-сайту з нуля слугує метод використання систем управління контентом (CMS). Можна використовувати такі CMS, як «WordPress», «Joompla» або «Drupal», для для створення та управління контентом вашого веб-сайту без необхідних знань кодування. Перевагою цього методу є зручність у використанні, велика підтримка спільноти, різноманітність плагінів та тем, однак із мінусів веб-сайт буде не таким гнучким у налаштуванні та можуть виникнути проблеми з безпекою.
- Розробник також може обрати буд який онлайн конструктор веб-сайтів, такі як «Wix», «Weebly» або «Squarespace», щоб створити веб-сайт за допомогою інтерфейсу «перетягування». Перевагою таких конструкторів є відсутність потреби в кодуванні, доступні шаблони готових рішень та простота у використанні. Мінусами є обмежені можливості у кастомізації для досвідчених користувачів.

Я навів всього лише кілька існуючих методів для вибору при розробці веб-сайту. Виберіть метод, який відповідає вашим цілям, технічним навичкам і специфічним вимогам вашого веб-сайту. Часто можна комбінувати ці методи залежно від різних аспектів вашого проекту.

В цій роботі я буду використовувати метод написання веб-сайту з нуля, тому тут будуть описані необхідні для цього методу матеріали.

2.4.1 HTML

HTML або Hyper Text Markup Language – є фундаментальним будівельний блоком Всесвітньої павутини. Це стандартна мова розмітки, що використовується

для створення та оформлення веб-сторінок. Вона була розроблена Тімом Бернерсом-Лі в 1991 році, HTML слугує основою для структурування контенту в Інтернеті.

Ось ключові аспекти HTML:

- Мова розмітки: HTML використовує систему тегів для структурування контенту. Теги – це елементи, укладені в кутові дужки (<>), які визначають різні частини веб-сторінки. Кожен тег служить певній меті, наприклад для позначення заголовків, абзаців, посилань, зображень тощо. На рис.2.8 зображено приклад синтаксису цієї мови для створення абзацу, посилання та відображення зображення.

```
<p>This is a paragraph.</p>
<a href="https://test.com">Visit Test.com</a>
<img src="Test.jpg" alt="An test image">
```

Рисунок 2.8 – Приклад мови розмітки

- Структура документа: Документи HTML мають ієрархічну структуру. Базова структура включає оголошення <!DOCTYPE HTML>, за яким слідують елементи <html>, <head> та <body>. В <head> міститься метаінформація про документ, тоді як <body> вже містить сам контент веб-сайту. На рис.2.9 зображено базова структура HTML документа, яка слугує відправною точкою для створення всієї веб-сторінки.

example > ...

```
<!DOCTYPE html>
<html>

<head>
  <title>My Web Page</title>
</head>

<body>
  <h1>Welcome to My Web Page</h1>
  <p>This is the main content of the page.</p>
</body>

</html>
```

Рисунок 2.9 – Базове оформлення HTML документу

- Елементи та атрибути: Елементи HTML – це будівельні блоки веб-сторінки і вони можуть містити атрибути, які надають додаткову інформацію про елемент. Атрибути зазвичай містяться у відкриваючому тезі. Наприклад на рис.2.10 атрибут «target="_blank"» міститься в відкриваючому тезі і слугую для переходу по гіперпосиланні в новому вікні браузера.

```
<a href="https://example.com" target="_blank">Visit Example.com</a>
```

Рисунок 2.10 – Приклад використання атрибуту в елементі веб-сторінки

- Семантичний HTML: Він має на меті надати сенс структурі веб-сторінки, роблячи її більш доступною як для користувачів, так і для пошукових систем. Такі елементи, як <header>, <footer>, <article> та <nav> передають передбачуване значення певних розділів.
- HTML5: Мова постійно розвивається протягом багатьох років і HTML5 є на даний момент останньою версією. Вона вводить нові елементи, атрибути та API для покращення досвіду веб-розробки. Такі функції, як <canvas> для малювання графіки та <video> для вбудовування відео, є частиною HTML5.
- Крос-браузерна сумісність: HTML розроблений таким чином, щоб бути сумісним з різними веб-браузерами, гарантуючи, що веб-сторінки виглядатимуть і функціонуватимуть однаково на різних платформах.
- Адаптивний веб-дизайн: HTML працює в поєднанні з CSS (каскадними таблицями стилів) і JavaScript для створення адаптивних і візуально привабливих веб-сторінок. CSS використовуються для стилізації, а JavaScript додає інтерактивності для покращення користувацького досвіду.

Отже, HTML є основою веб-розробки, забезпечуючи структуру і основу для створення контенту в Інтернеті. Розуміння HTML необхідне кожному, хто займається створенням і підтримкою веб-сайтів.

2.4.2 CSS

CSS або каскадні таблиці стилів – це важливий компонент веб-розробки, який працює разом з HTML, визначаючи візуальне представлення та макет веб-сторінки. Розроблений Консорціумом Всесвітньої павутини (W3C), CSS дозволяє розробникам відокремити структуру та зміст веб-сторінки від її стилю та дизайну.

Ось ключові аспекти CSS:

- Розділення завдань: CSS сприяє розділенню завдань, дозволяючи веб-дизайнерам і розробникам зберігати структуру і вміст веб-сторінки, яка оброблюється в HTML окремо від її візуального представлення оброблюваного в CSS. Таке розділення полегшує підтримку та оновлення веб-сайтів.
- Селектори та декларації: CSS використовує селектори для виділення елементів HTML і застосування правил стилів. Правила стилів складаються з декларацій, де властивості та значення визначають, як мають виглядати обрані елементи. На рис.2.11 зображено приклад стилізації одного з елементів веб-сторінки.

```
/* CSS Rule */
p {
color: #333;
font-size: 16px;
margin-bottom: 20px;
}
```

Рисунок 2.11 – Демонстрація CSS

- Box Model: Це фундаментальна концепція в CSS, яка описує, як елементи структуруються з точки зору вмісту, відступів, меж і полів.

Розуміння цієї моделі має вирішальне значення для створення макетів і інтервалів на веб-сторінці.

- Адаптивний дизайн: CSS відіграє важливу роль у створенні адаптивного веб-дизайну, який адаптується до різних розмірів екрану та пристроїв. На рис.2.12 зображено реалізації адаптивного дизайну для різного розширення екрану користувача. Медіа-запити дозволяються розробникам застосовувати різні стилі на основі таких факторів, як ширина, висота та роздільна здатність екрану.

```
/* Media Query for Responsive Design */
@media screen and (max-width: 600px) {
  body {
    font-size: 14px;
  }
}
```

Рисунок 2.12 – Реалізація адаптивного дизайну

- Flexbox and Grid Layout: CSS представляє такі моделі верстки як Flexbox та Grid, які спрощують створення складних і адаптивних макетів веб-сторінок. Ці моделі надають більше контролю над розташуванням і вирівнюванням елементів.
- Переходи та анімація: CSS дозволяє створювати плавні переходи та анімацію, покращуючи досвід користувача. Властивості переходів можна використовувати для керування поступовою зміною стилів елементів, а key frames дозволяють створювати більш складні анімації. На рис.2.13 наведено приклад використання цієї особливості.

```css
/* CSS Animation */
@keyframes fadeIn {
  from {
    opacity: 0;
  }

  to {
    opacity: 1;
  }
}

.fade-in {
  animation: fadeIn 1s ease-in-out;
}
```

Рисунок 2.13 – Створення анімації в CSS файлі

- CSS Preprocessors: CSS preprocessors, такі як SASS та LESS, розширюють можливості традиційного CSS, додаючи такі функції, як змінні, вкладеність і функції. Ці препроцесори роблять CSS більш зручним для підтримки та масштабування.
- CSS Frameworks: CSS Frameworks, такі як Bootstrap і Foundation, надають набір заздалегідь розроблених стилів і компонентів, які розробники можуть використовувати для спрощення процесу створення послідовних і візуально привабливих веб-сторінок.

Отже, CSS – це потужна мова стилів, яка дає змогу розробникам контролювати представлення та макетування веб-контенту. Її гнучкість і можливості сприяють створенню сучасних, адаптивних і візуально привабливих веб-сайтів.

2.4.3 JavaScript

JavaScript, часто скорочується до JS – це універсальна мова програмування, яка відіграє вирішальну роль у веб-розробці. Спочатку розроблена компанією Netscape, JavaScript перетворився на потужну і широко використовувану мову, яка підтримується всіма основними веб-браузерами. Вона дозволяє розробникам

створювати динамічний і інтерактивний контент на стороні клієнта, покращуючи досвід користування веб-сторінками.

Ось ключові аспекти JavaScript:

- Client-Side Scripting: JavaScript в основному використовується для написання сценаріїв на стороні клієнта, тобто виконується в веб-браузері користувача, а не на сервері. Це дозволяє розробникам маніпулювати об'єктною моделлю документа (DOM) в режимі реального часу, динамічно оновлюючи вміст без необхідності перезавантаження сторінки.
- Маніпуляції з DOM: Об'єктна модель документа представляє структуру веб-сторінки у вигляді дерева об'єктів. JavaScript надає можливість маніпулювати цією структурою, дозволяючи розробникам динамічно змінювати вміст, стилі та атрибути. На рис.2.14 зображено приклад такої маніпуляції.

```
// DOM Manipulation
document.getElementById("myElement").innerHTML="New content";
```

Рисунок 2.14 – Маніпуляція веб-сторінкою використовуючи DOM

- Обробка подій: JavaScript полегшує обробки взаємодії з користувачем за допомогою подій. Розробники можуть визначати функції для редагування на такі події, як кліки, натискання клавіш і заповнення форм, створюючи інтерактивні та адаптивні веб сторінки. На рис.2.15 зображено простий приклад такої обробки подій.

```
// Event Handling
document.getElementById('myButton').addEventListener('click', function () {
  alert('Button clicked!');
});
```

Рисунок 2.15 – Обробка події при натисканні на кнопку

- Асинхронне програмування: JavaScript підтримує асинхронне програмування за допомогою таких функцій, як зворотні виклики та обіцянки. Це дозволяє розробникам виконувати завдання, не блокуючи основний потік, підвищуючи продуктивність веб-додатків.
- JavaScript характеризується динамічною типізацією, що означає, що типи змінних встановлюються протягом виконання програми. Така гнучкість дозволяє розробникам писати більш лаконічний код, але вимагає ретельного підходу, щоб запобігти неочікуваній поведінці.
- Крос-браузерна сумісність: JavaScript підтримується всіма основними веб-браузерами, що робить його універсальною мовою сценаріїв для веб-розробки. Однак для забезпечення сумісності розробникам може знадобитися врахувати відмінності в реалізації браузерів.
- Фреймворки та бібліотеки: Різноманітні фреймворки та бібліотеки, побудовані на JavaScript, такі як React, Angular і Vue.js, спрощують розробку складних веб-додатків. Ці інструменти забезпечують багаторазове використання компонентів, управління станами та інші функції, що підвищують продуктивність.
- JavaScript на стороні сервера: Хоча JavaScript переважно відомий для написання сценаріїв на стороні клієнта, він також може використовуватися на стороні сервера. Node.js, серверне середовище виконання JavaScript, дозволяє розробникам використовувати JavaScript для створення масштабованих та ефективних серверних додатків.

Отже, JavaScript – це фундаментальна мова для веб-розробки, що дозволяє створювати динамічні та інтерактивні веб-сторінки. Її безперервний розвиток і широке розповсюдження сприяють її значущості в постійно мінливому ландшафті фронтенд і бекенд розробки.

В цьому розділі було розглянуто три основні компоненти веб-розробки: HTML, CSS, JavaScript. HTML, як мова розмітки, забезпечує структуру та зміст

веб-сторінок. CSS доповнює HTML за допомогою стилів і форматування контенту, дозволяючи створювати візуально привабливий та адаптивний дизайн. JavaScript універсальна мова програмування, яка покращує взаємодію з користувачем і додає динамічної поведінки веб-сторінкам на стороні клієнта.

Разом ці технології формують основу сучасної веб-розробки. HTML структурує контент, CSS його стилізує, а JavaScript оживляє його за допомогою інтерактивних функцій. Розуміння взаємодії між цими трьома компонентами необхідне веб-розробникам для створення цікавих, функціональних і зручних для користувача веб-сайтів.

Роль HTML у створенні структури документу, внесок CSS у стилізацію та макетування, а також здатність JavaScript забезпечувати динамічний контент та інтерактивність підкреслюють синергію між цими технологіями. Крім того, були зачеплені передові концепції, такі як семантичний HTML, адаптивний дизайн та фреймворки CSS, які сприяють подальшому вдосконаленню та підвищенню ефективності веб-розробки.

Оскільки технології продовжують розвиватися, вкрай важливо бути в курсі останніх стандартів, функцій та найкращих практик у HTML, CSS та JavaScript. Постійний розвиток цих технологій, про що свідчать оновлення HTML5 та препроцесорів CSS, відображає динамічну природу веб-розробки. Більше того, поява фреймворків і бібліотек, таких як ті, що побудовані на JavaScript, демонструє зусилля спільноти, спрямовані на впорядкування та вдосконалення процесу розробки.

Отже, цілісне розуміння HTML, CSS та JavaScript дає розробникам можливість створювати добре структуровані, візуально привабливі та інтерактивні веб-ресурси. Ці технології, працюючи у гармонії, формують сучасний цифровий ландшафт і відіграють ключову роль у створенні віртуального світу, з яким ми взаємодіємо щодня.

3 Розробка веб-додатку

У цьому розділі дипломної роботи буде детально описано проект, програмне забезпечення, що використовується для його створення, а також аналіз самого проекту. Цей розділ має на меті представити готовий проект та опис його процесу створення.

У ньому буде детально розглянуто програмне забезпечення, яке є невід'ємною частиною робочого процесу. Також буде описано процес створення самого проекту від початку, до кінця.

Для створення веб-додатку було використано багато різноманітних програм та додатків до них.

Ось перелік використаних програм:

- Visual Studio Code, або VSCode
- Adobe Photoshop
- Git
- Google Chrome
- Chrome DevTools

Також були використані наступні мови програмування:

- HTML
- CSS
- SCSS
- JavaScript

В наступних підрозділах будуть описані ці програми, для чого вони створені, як використовуються, їх переваги та недоліки.

3.1 Visual Studio Code

Visual Studio Code (VSCode) – популярний та універсальний редактор вихідного коду, розроблений компанією Microsoft. Запущений у квітні 2015 року,

VSCode набув широкого розповсюдження серед розробників завдяки своїм потужним можливостям, розширюваності та крос-платформною підтримкою.

Ось деякі ключові аспекти цього редактору:

- Крос-платформна сумісність. VSCode розроблено для безперебійної роботи на основних операційних системах, включаючи Windows, macOS та Linux. Ця крос-платформна сумісність робить його улюбленим серед розробників, які використовують різні середовища для своєї роботи.
- Відкритий код. Visual Studio Code є відкритим проектом, і його вихідний код можна знайти на платформі GitHub. Ця відкритість заохочує залучення спільноти, дозволяючи розробникам робити свій внесок у його розвиток, повідомляти про проблеми та створювати розширення.
- Зручний інтерфейс. Інтерфейс користувача VSCode чистий та інтуїтивно зрозумілий. Він має мінімалістичний дизайн з акцентом на простоту та продуктивність. Редактор пропонує бічну панель для зручної навігації, потужний функціонал пошуку та макет, що налаштовується відповідно до індивідуальних вподобань.
- Розширюваність. Однією з головних особливостей VSCode є його розширюваність. Він підтримує широку екосистему розширень, які розширюють його функціональність. Ці розширення охоплюють широкий спектр мов програмування, фреймворків та інструментів, що дозволяють розробникам налаштовувати середовище кодування відповідно до своїх потреб. Пізніше будуть описані деякі з цих розширень використані для створення проекту.
- Інтегрований Git. VSCode має вбудовану інтеграцію з Git-ом, що спрощує управління версіями для розробників. Ця функція дозволяє користувачам керувати репозиторіями, фіксувати зміни та вирішувати

конфлікти безпосередньо з редактора. Подання контролю вихідного коду забезпечує візуальне представлення змін і полегшує спільну роботу в командному середовищі.

- Інтегрований термінал. VSCode містить інтегрований термінал, що позбавляє розробників необхідності перемикатися між редактором і зовнішніми вікнами терміналів. Ця функція дозволяє користувачам запускати команди, скрипти та інструменти у редакторі, покращуючи робочій процес розробки.
- Інтелектуальне редагування коду. VSCode пропонує потужні функції редагування коду, включаючи підсвічування синтаксису, автозавершення та рефакторинг коду. Він також підтримує IntelliSense, функцію, яка надає контекстно-залежні підказки для змінних, методів та API, що робить кодування швидшим та ефективнішим.
- Підтримка налагодження. Редактор надає потужні можливості налагодження з підтримкою різних мов програмування. Точки зупинки, спостереження за змінними та покрокове налагодження допомагають розробникам виявляти та виправляти помилки в коді.
- Велика спільнота та підтримка. Visual Studio Code має жваву та активну спільноту розробників. Така розробка, керована спільнотою, забезпечує постійне вдосконалення, регулярні оновлення та виправлення помилок. Крім того, існує велика кількість онлайн-ресурсів, форумів та документації, які допоможуть користувачам розпочати роботу та вирішувати проблеми.

Таким чином, Visual Studio Code став популярним вибором для розробників завдяки своїй крос-платформній сумісності, розширюваності, інтегрованими функціями та потужній підтримці спільноти. Незалежно від того, чи працюєте ви над веб-розробкою, програмною інженерією або наукою про дані, VSCode надає універсальне та ефективне середовище для кодування.

Приблизно 90% часу створення проекту було проведено саме в цій програмі.

3.1.1 Auto Rename Tag

Зручний для перейменування тегів HTML. Якщо тег повторюється, при зміні одного за допомогою плагіна всі інші будуть перейменовані автоматично.

3.1.2 HTML CSS support

HTML CSS support – це цінне розширення для Visual Studio Code, яке покращує робочий процес веб-розробки завдяки розширеній підтримці мов HTML та CSS. Це розширення призначене для підвищення ефективності та точності кодування, пропонуючи такі функції, як автозавершення, підсвічування синтаксису та інтелектуальні підказки для коду HTML і CSS. Завдяки цій підтримці ви можете легко зменшити кількість помилок і прискорити створення адаптивних і візуально привабливих веб-сторінок у популярному редакторі коду.

3.1.3 Live SASS Compiler

Live SASS Compiler – це обов’язкове розширення для Visual Studio Code, яке спрощує процес веб-розробки, забезпечуючи компіляцію та попередню обробки SASS/SCSS файлів у режимі реального часу. Це розширення дозволяє вам писати ваші таблиці стилів на потужному і багатофункціональному синтаксисі SASS або SCSS і миттєво бачити застосовані зміни без необхідності ручної компіляції. За допомогою Live SASS Compiler ви можете насолоджуватися попереднім переглядом ваших стилів під час кодування, що робить розробку та налагодження стилів більш ефективною. Це розширення легко інтегрується у середовище Visual Studio Code, пропонуючи зручні налаштування для адаптації до конкретних вимог проекту.

3.1.4 Live Server

Live Server – це незамінне розширення для Visual Studio Code, яке змінює досвід веб-розробки. Це розширення перетворює середовище кодування на сервер розробки в режимі реального часу, дозволяючи миттєво переглядати та взаємодіяти

з веб-додатками під час написання коду. За допомогою Live Server можливо легко запустити локальний сервер для обслуговування HTML, CSS та JavaScript файлів, забезпечуючи динамічний і адаптивний попередній перегляд вашої роботи в режимі реального часу. Це розширення усуває необхідність ручного оновлення браузера, спрощуючи робочий процес розробки та заощаджуючи час. Live Server змінює правила гри, розширюючи можливості створення, тестування та налагодження веб-додатків з неперевершеною ефективністю.

3.2 Adobe Photoshop

Adobe Photoshop – це потужне і широко використовуване програмне забезпечення для редагування растрової графіки, розроблене компанією Adobe Inc. Це галузевий стандарт для маніпулювання цифровими зображеннями, редагування фотографій та графічного дизайну. Photoshop став синонімом редагування зображень і відіграв вирішальну роль у формуванні візуального ландшафту різних галузей, включаючи фотографію, графічний дизайн, веб-дизайн та мультимедіа.

Основні можливості Adobe Photoshop включають в себе:

- Інструменти для редагування зображень. Photoshop надає повний набір інструментів для базового та просунутого редагування зображень. Користувачі можуть обрізати, змінювати розмір і налаштовувати експозицію, колірний баланс і насиченість зображень. Програма також пропонує розширені функції, такі як заливка з урахуванням вмісту та виправлення плям.
- Шари та маски. Однією з визначних особливостей Photoshop є підтримка шарів. Шари дозволяють користувачам працювати з різними елементами зображення окремо, що полегшує редагування та маніпуляції з окремими частинами, не впливаючи на все зображення. Маски ще більше розширюють ці можливості, дозволяючи точно контролювати, які ділянки зображення є видимими, а які прихованими.

- Графічний дизайн. Photoshop широко використовується для графічного дизайну, зокрема для створення плакатів, флаєрів, банерів та інших маркетингових матеріалів. Інструменти для роботи з текстом і фігурами, а також стилі шарів і можливості змішування роблять його універсальним інструментом для створення візуально привабливих дизайнів.

Незважаючи на широке розповсюдження та популярність, Adobe Photoshop є складною програмою для навчання через свій широкий набір функцій. Однак незліченна кількість навчальних посібників,

онлайн-ресурсів та потужна спільнота роблять його доступним, як для початківців, так і для професіоналів.

3.3 Git

Git – це розподілена система контролю версій (DVCS), яка широко використовується для відстеження змін у вихідному коді під час розробки програмного забезпечення. Створена Лінусом Торвальдсом у 2005 році, Git став стандартом для контролю версій завдяки своїх ефективності, гнучкості та потужним можливостям розгалуження та об’єднання.

Ключові поняття:

- Репозиторій. Репозиторій Git-а – це колекція файлів і вся історія змін, внесених до цих файлів. Сховища можуть бути локальними, віддаленими або їх комбінацією.
- Commit. Це «знімок змін», внесених до файлів у сховищі в певний момент часу. Кожний коміт ідентифікується унікальним хешем і включає повідомлення, що описує зміні.
- Branch. Git використовує це для забезпечення паралельної розробки. Основна гілка зазвичай називається «master», а розробники створюють функціональні гілки для роботи над конкретними завданнями. Гілки

можна легко об’єднувати, що дозволяє вносити зміни з однієї гілки в іншу.

- Pull Request. У середовищі спільної розробки учасники створюють запити на витягування, щоб запропонувати зміни до основної гілки. Pull-запити надають можливість переглянути та обговорити зміни коду перед тим, як вести їх до основної кодової бази.

Git грає ключову роль у сучасній програмній розробці, сприяючи співпраці, надаючи надійну систему управління версіями та спрощуючи керування складними кодовими базами. Його популярність поширюється не лише на звичайні проекти розробки програмного забезпечення, але і на різноманітні галузі, що робить його універсальним та невід’ємним інструментом у світі технологій.

3.4 Google Chrome

Google Chrome – популярний веб-браузер, розроблений компанією Google. Запущений у вересні 2008 року, він швидко набув широкого розповсюдження і став одним з найпоширеніших браузерів у світі.

Ось деякі ключові аспекти цього веб-браузера:

- Інтерфейс користувача. Google Chrome відомий своїм мінімалістичним і зручним інтерфейсом. Браузер має чистий дизайн з простим адресним рядком, широко відомим як Omnibox, де користувачі можуть вводити URL-адреси або здійснювати пошук. Вкладки розташовані у верхній частині вікна, і користувачі можуть легко відкривати, закривати або переставляти їх.
- Продуктивність. Chrome відомий своєю швидкою роботою та ефективним використанням системних ресурсів. У ньому використовується движок JavaScript V8, який призначений для швидкого виконання JavaScript-коду, що підвищує загальну швидкість роботи веб-додатків та веб-сайтів.

- Керування вкладками. У Google Chrome з’явилися інноваційні функція керування вкладками. Користувачі можуть легко відкривати нові вкладки, перетягувати їх, щоб змінити порядок і групувати їх в окремі процеси. Така ізоляція процесів гарантує, що якщо одна вкладка вийде з ладу, це не вплине на роботу всього браузера.
- Безпека. Безпека є пріоритетом для Google Chrome. Він включає в себе такі функції, як автоматичне оновлення, щоб забезпечити користувачам найновіші патчі безпеки. Браузер також має вбудовану систему захисту від фішингу та шкідливого програмного забезпечення, щоб захистити користувачів від шкідливих веб-сайтів і завантажень.
- Синхронізація та інтеграція з обліковим записом. Google Chrome дозволяє користувачам входити в систему за допомогою свого облікового запису Google. Ця функція дозволяє синхронізувати вкладки, закладки, історію, паролі та інші налаштування на різних пристроях. Користувачі можуть легко переходити з одного пристрою на інший, зберігаючи при цьому всі свою дані.

Підсумовуючи, можна сказати, що Google Chrome став домінуючою силою на ринку веб-браузерів завдяки своїй швидкості, простоті та надійним функціям. Постійні оновлення та прихильність до безпеки сприяють його постійній популярності серед користувачів у всьому світі.

3.5 Chrome DevTools

Chrome DevTools – це набір інструментів, призначений для веб-розробників, що входить у склад браузера Google Chrome. Ці інструменти дозволяють розробникам перевіряти, налагоджувати та оптимізувати веб-сторінки та веб-додатки, що робить їх важливим ресурсом для всіх, хто займається веб-розробкою. Незалежно від того, чи ви є фронтенд-розробником, дизайнером або тим, хто налагоджує веб-додатки, Chrome DevTools надає повний набір функцій, які допоможуть у процесі розробки.

Основні можливості Chrome DevTools включають в себе наступні:

- Перевірка елементів. Розробники можуть перевіряти і змінювати HTML і CSS веб-сторінки в режимі реального часу. Панель «Елементи» дозволяє досліджувати об’єктну модель документа сторінки.
- Консоль. Панель консолі дозволяє розробникам реєструвати інформацію, виконувати код JavaScript і взаємодіяти з сторінкою. Помилки, попередження та інші повідомлення відображаються в цій консолі, що полегшує процес налагодження.
- Моніторинг мережі. Розробники можуть аналізувати мережеву активність, відстежувати час завантаження ресурсів і виявляти вузькі місця. Панель «Мережа» надає інформацію про HTTP-запити, відповіді та передачу даних.
- Мобільна емуляція. Розробники можуть імітувати різні мобільні пристрої та мережеві умови, щоб перевіряти адаптивність своїх веб-сайтів.

Chrome DevTools – безцінний ресурс для розробників, що надає багатий набір інструментів для оптимізації процесу розробки та налагодження. Він став невід’ємною частиною робочого процесу веб-розробки, дозволяючи розробникам створювати якісні, продуктивні та безпечні веб-додатки.

3.6 Розробка клієнтської частини веб-додатку

В клієнтській частині розроблюється максимально зручний та доступний для користувача інтерфейс веб-додатку. Інтерфейс повинен бути зручним у використанні, інтуїтивно зрозумілим та не мати багато зайвої інформації.

На рис.3.1 зображено макет веб-додатку для відображення інформації про погоду в запитуваному місті в даний момент часу. На макеті представлено початковий вигляд веб-додатку, коли користувачу необхідно ввести необхідне для нього місто, та результат виводу цієї інформації. Також зображено різні погодні умови в залежності від погодних умов в шуканому місті.

Рисунок 3.1 – Макет та матеріали веб-додатку

3.7 Розробка програмної частини веб-додатку

На рис.3.2 зображена логіка роботи погодного веб-додатку, який розроблюється в цій роботі. При використанні цього веб-додатку користувачу необхідно спочатку ввести місто в «SearchBar», погодні умови якого він бажає дізнатися. На цьому моменті у веб-додатку є два шляхи поточної роботи. Якщо користувач ввів коректне місто в світі, веб-додаток зобразить йому погодні умови в місті на даний момент, включаючи графічне зображення погодних умов, вологість повітря та швидкість вітру. В іншому випадку, якщо користувач ввід некоректне місто, або помилився в його назві веб-додаток зобразить помилку, та опише де була зроблена помилка. Коли користувач бажає подивитися інформацію про інше місто він може ввести назву в «SearchBar», воно нікуди не зникає й завжди відображається.

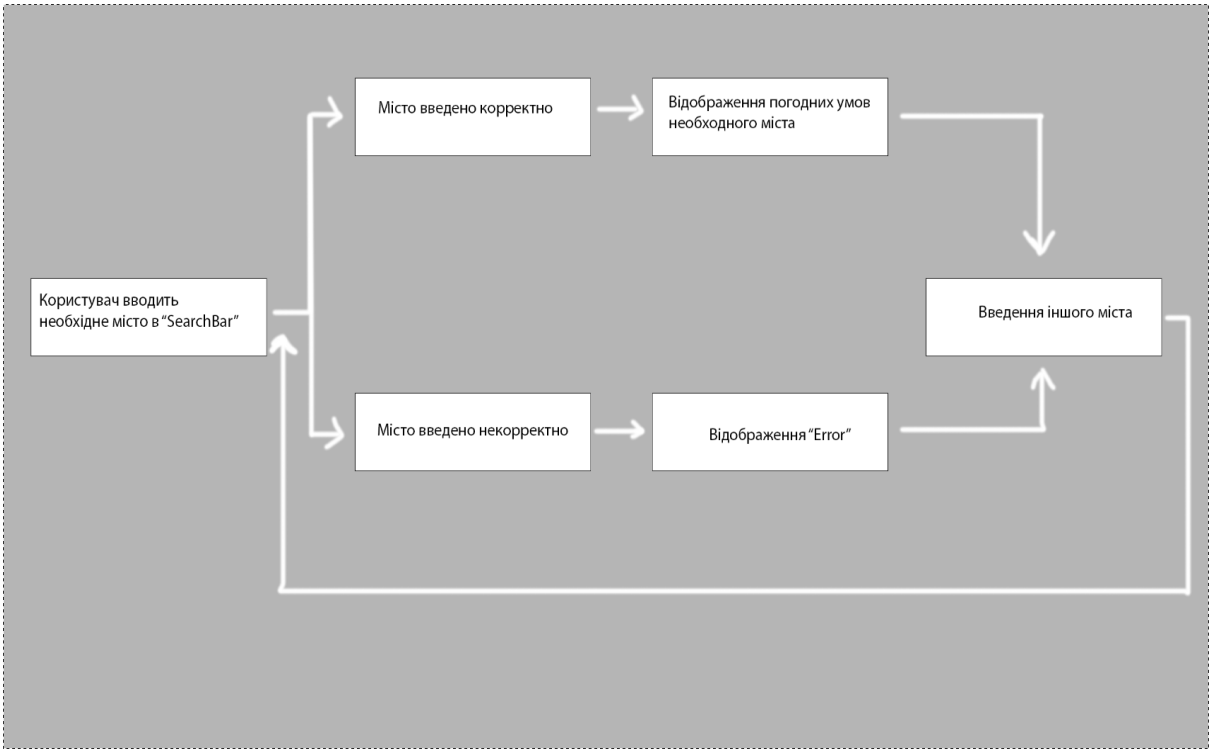


Рисунок 3.2 – Модель роботи веб-додатку

3.7.1 Пояснення роботи HTML

Цей веб-додаток розроблюється в редакторі коду Visual Studio Code.

Для його кодування використовуються такі мови програмування:

- HTML
- CSS
- SCSS
- JavaScript

HTML код веб-додатку записуються в однойменному за форматом файлі, так само, як й CSS, SCSS та JavaScript. На рис.3.3 зображено файлову структуру проекту. Кожен файл цього проекту лежить в конкретному місці, відповідаючи його типу. Так наприклад зображення використовувані в цьому проекті лежать в папці «images», а CSS та JavaScript файли в відповідних вже ним папкам. Це полегшує процес пошуку в проекті, коли він великий і файлів не 1-3, а набагато більше.

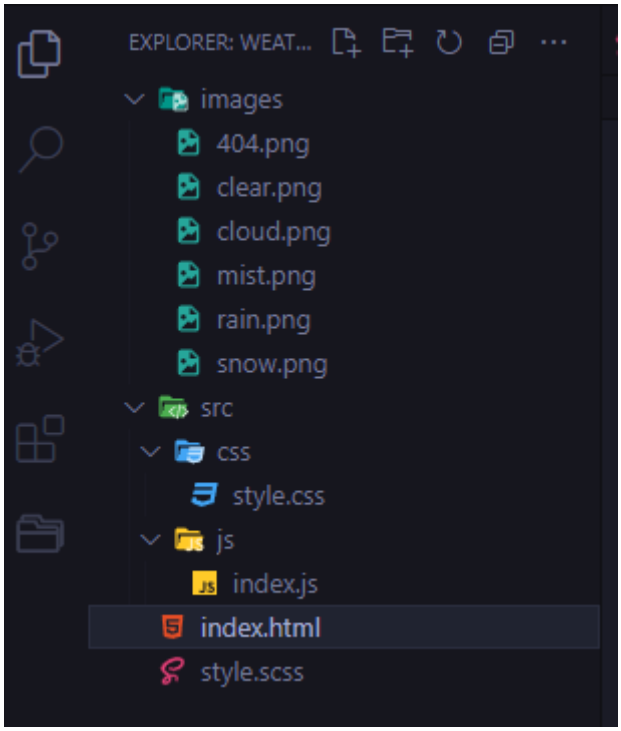


Рисунок 3.3 – Файлова структура

Головним файлом цього проекту є «index.html». В цьому файлі створюється розмітка самого веб-додатку, створюються елементи та наповнюються контентом. До цього файлу підключені файли «style.css» та «index.js».

Файл «index.html» починається з оголошення <!DOCTYPE HTML> за яким слідує елемент <html> та <head, як це зображено на рис.3.4

Рисунок 3.4 – Початок «index.html»

В елементі <head> вказана всі необхідні для пошукових систем метадані. Також в саме цьому елементі підключаються таблиці стилів (CSS), вказується назва веб-сторінки, та можуть бути підключені стороні шрифти. Шрифт може буди

підключений й в самому CSS файлі, проте я підключаю цей шрифт саме в файлі HTML, тому що він всього лише один.

Підключення стороннього шрифту з Google Fonts Api:

<link

href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800;900&family=Roboto:wght@300;400;500;700;900&display=swap"

rel="stylesheet">

В цій частині коду виконується запит до Google Fonts Api, вказуючи, що саме необхідно отримати, його атрибути та характеристики. Якщо не використовувати технологію API, необхідно було б знайти цей шрифт, завантажити його на сервер.

CSS файли підключаються трохи схоже:

<link rel="stylesheet" href="src/css/style.css">

Наступним елементом йде тег <body>. В ньому вже містяться всі інші елементи та їх атрибути, такі як абзаци, малюнки, поля вводу тощо. Також саме в ньому йде підключення сторонніх скриптів.

Елементи в цьому тегу описані мною за допомогою базових блокових елементів - . Це фундаментальний будівельний блок, який використовується для групування та структурування контенту на веб-сторінці, це контейнер, який не несе жодного конкретного навантаження. Проте, це загальний елемент, який часто використовується для групування інших елементів, що дозволяють застосовувати стилі або маніпулювати ними як єдиним цілим. На рис.3.5 зображено використання цього елементу для групування в ньому інших елементів.

```
<div class="search-box">
  <i class="fa-solid fa-location-dot"></i>
  <input type="text" placeholder="Enter your location.">
  <button class="fa-solid fa-magnifying-glass" onclick="get()"></button>
</div>
```

Рисунок 3.5 – Використання

В наведеному рисунку елемент , далі пошуковий контейнер, слугує обгорткою для 3-х інших елементів в ньому. В пошуковому контейнері знаходяться кілька елементів, а саме: елемент іконки, поле вводу та кнопка. Вони мають загальні налаштування стилю, набагато легше налаштувати їх всіх разом, аніж кожному елементу окремо прописувати ці налаштування в CSS файлі. На рис.3.6 зображено результат цього контейнеру в суміші з налаштуваннями в CSS файлі.

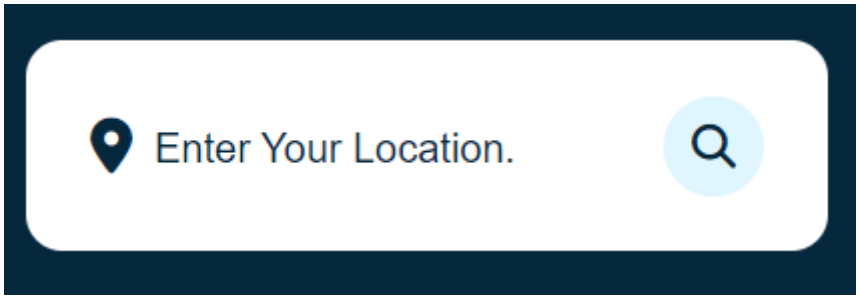


Рисунок 3.6 – Пошуковий контейнер

В схожих контейнерах описуються інші елементи веб-додатку. На рис.3.7 зображено контейнер з різними типами елементів, які виконують певну задачу. Результатом цього коду є рис.3.8 на якому зображено погодні умови в місті «Київ». Всі елементи цього рисунку містяться на рис.3.7

```
<div class="weather-details">
  <div class="humidity">
    <i class="fa-solid fa-water"></i>
    <div class="text">
      <span></span>
      <p>Humidity</p>
    </div>
  </div>
  <div class="wind">
    <i class="fa-solid fa-wind"></i>
    <div class="text">
      <span></span>
      <p>Wind Speed</p>
    </div>
  </div>
</div>
```

Рисунок 3.7 – Код відображення інформації про погодні умови міста

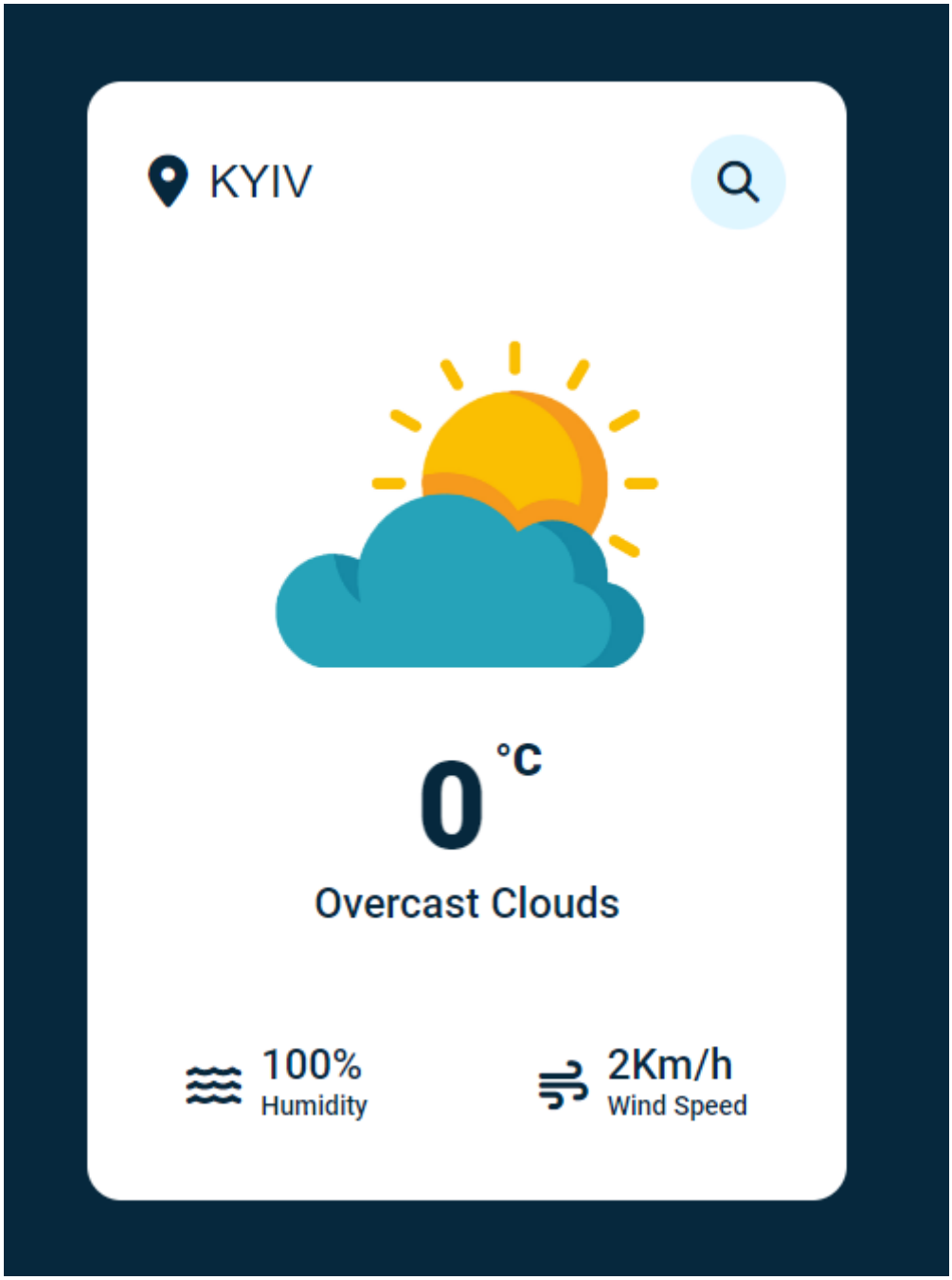


Рисунок 3.8 – Демонстрація роботи веб-додатку

В самому кінці тегу <body> йде підключення сторонніх скриптів.

<script src=https://kit.fontawesome.com/7c8801c017.js crossorigin="anonymous"></script>

<script src="src/js/index.js"></script>

3.7.2 Пояснення роботи CSS та SCSS

В цій частині описується принцип роботи CSS та SCSS.

SCSS – це мова сценаріїв, яка є підмножиною каскадних таблиць стилів (CSS). Вона додає додаткові функції та можливості до CSS, роблячи її потужнішою в підтримці. SCSS зазвичай використовується у веб-розробці для стилізації та дизайну зовнішнього вигляду веб-сайтів і веб-додатків. SCSS використовує більш стислий і читабельний синтаксис у порівнянні з традиційним CSS. Він вводить такі функції, як змінні, вкладені правила та міксини, які спрощують процес написання коду. Це робить код більш модульним і легшим в обслуговуванні.

Для використання SCSS необхідно встановити розширення «Live SASS Compiler» до Visual Studio Code. Всі стилі для форматування елементів будуть кодуватися в файлі «style.scss». Однак, підключати до HTML необхідно саме «style.css».

Однією з особливостей цієї мови є можливість створювати змінні й використовувати їх в будь якій частині файлу. В моєму проекті створено кілька таких змінних для полегшення вводу деяких даних, наприклад Hex коду різних кольорів. Набагато легше вести змінну, в якій знаходиться необхідний колір, аніж писати його код кожен раз коли він необхідний. На рис.3.9 зображено створення таких змінних.

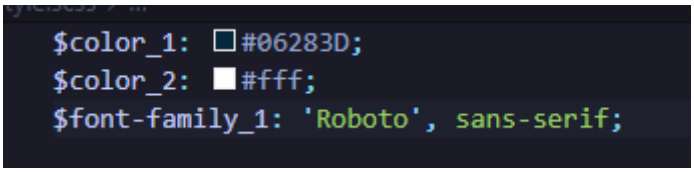


Рисунок 3.9 – Приклад створення змінних в SCSS

Можливість використання змінних, не єдина особливість SCSS. SCSS дозволяє використовувати вкладеність при написанні стилів. Наприклад в традиційному CSS розробнику необхідно було писати

.search-box input::placeholder {

```
  font-size: 20px;
  font-weight: 500;
  color: #06283D;
  text-transform: capitalize;
}
.search-box input::placeholder {
  font-size: 20px;
  font-weight: 500;
  color: #06283D;
  text-transform: capitalize;
}
```

для звернення до атрибуту «placeholder» тегу «input», який знаходиться в контейнері «search-box». На мою думку, це не дуже зручно, в плані читабельності.

В SCSS цей запис виглядає ось так:

```
.search-box {
  width: 100%;
  height: min-content;
  display: flex;
  align-items: center;
  justify-content: space-between;

  input {
    color: $color_1;
    width: 80%;
```

```
    font-size: 24px;
    font-weight: 500;
    text-transform: uppercase;
    padding-left: 32px;

    &::placeholder {
     font-size: 20px;
     font-weight: 500;
     color: $color_1;
     text-transform: capitalize;
    }
  }
```

Розробник пише стилі для контейнеру «search-box», в ньому ж звертається до «input», описує його, а потім може звернутися до будь якого атрибуту «input», описати його, а потім продовжити описувати, наприклад, «search-box» не шукаючи кожен раз в файлі де він його починав описувати.

3.7.3 Пояснення роботи JavaScript

JavaScript у веб-додатку виконує кілька ключових функцій, він використовується для взаємозв'язку з OpenWeatherMapApi та динамічного оновлення контенту.

Весь код цього скрипту описаний в файлі «index.js».

На початку файлу проводиться ініціалізація всіх змінних та їх значення для використання в цьому скрипті. Використовуючи DOM створюються змінні з CSS класами необхідних елементів веб-сторінки. Це необхідно для подальшої маніпуляції з ними для створення динамічного та інтерактивного інтерфейсу.

```
const container = document.querySelector('.container');
const search = document.querySelector('.search-box button');
const weatherBox = document.querySelector('.weather-box');
const weatherDetails = document.querySelector('.weather-details');
const error404 = document.querySelector('.not-found');
```

Для того щоб створена та стилізована в HTML та CSS кнопка хоча б щось робила їй необхідно створити «Event Listener» в JavaScript-файлі. В нашому випадку створюється функція, яка спрацьовує при натисканні на кнопку, або натисканні на клавіатурі клавіши «Enter».

```
document.addEventListener('keydown', function (e) {
  if (e.keyCode == 13) {
    get();
  }
});
```

Для роботи з OpenWeatherMapApi необхідно в JavaScript написати звертання до нього. Спочатку треба створити API-key для проекту, він необхідний для ідентифікації запиту. В проекті файлі створюється змінна з цим ключем, та змінна для отримання введених даних з поля <input>.

```
const APIKey = 'occm6b3ld5zboczgv47tim8tvv64ysk8';
const city = document.querySelector('.search-box input').value;
```

Далі створюється взаємодія з OpenWeatherMapApi.

```
fetch(
  `https://api.openweathermap.org/data/2.5/weather?q=${city}&units=metric&appid=${APIKey}`
)
```

```
.then((response) => response.json())
.then((json) => {
  if (json.cod === '404') {
    container.style.height = '400px';
    weatherBox.style.display = 'none';
    weatherDetails.style.display = 'none';
    error404.style.display = 'block';
    error404.classList.add('fadeIn');
    return;
  }
```

Веб-додаток звертається до API, записує до запиту необхідне місто та API-key. Результат цієї роботи записуються, якщо в процесі виконання не знаходить необхідне місто – видається помилка і створюється вікно з нею. Якщо необхідне місто було знайдено й дані було отримано, скрипт запускає процес відображення цих даних. Вносяться необхідні змінні в елементи веб-сторінки на основі отриманих даних.

```
  switch (json.weather[0].main) {
    case 'Clear':
      image.src = 'images/clear.png';
      break;

    case 'Rain':
      image.src = 'images/rain.png';
      break;
```

```
        case 'Snow':
          image.src = 'images/snow.png';
          break;

        case 'Clouds':
          image.src = 'images/cloud.png';
          break;

        case 'Haze':
          image.src = 'images/mist.png';
          break;

        default:
          image.src = '';
      }
```

В цій частині коду описані різні варіанти отриманих даних, в залежності від типу погоди в місті скрипт створює на веб-сторінці різні зображення, які відповідають погодним умовам.

3.7.4 Демонстрація проекту

На рис.3.10 зображено головну сторінку веб-додатку. На цій сторінці користувач може ввести необхідне йому місто для відображення погодних умов.

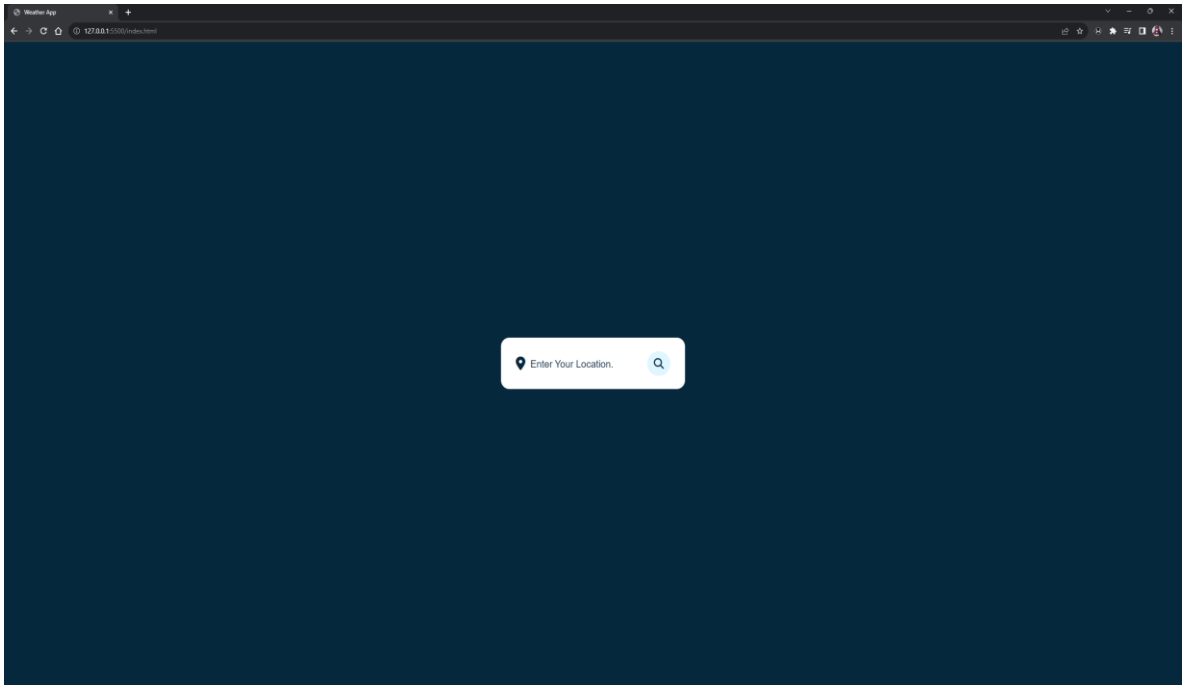


Рисунок 3.10 – Головна сторінка веб-додатку

На рис.3.11 зображено результат роботи додатку. За заданим місто користувач може дізнатися погодні умови в місті, а саме температуру в місті, її стан, швидкість вітру та вологість повітря.

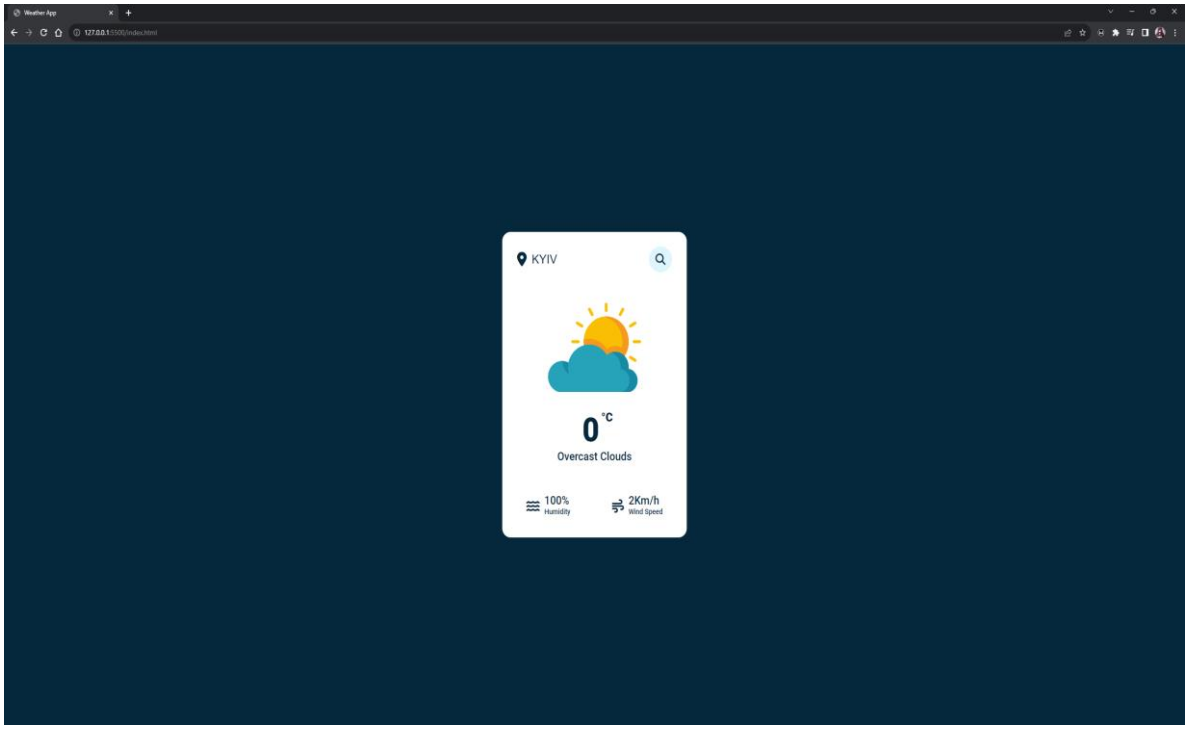


Рисунок 3.11 – Демонстрація працездатності додатку

На рис.3.12 зображено веб-додаток в тому випадку, якщо користувач ввід неправильну назву міста.

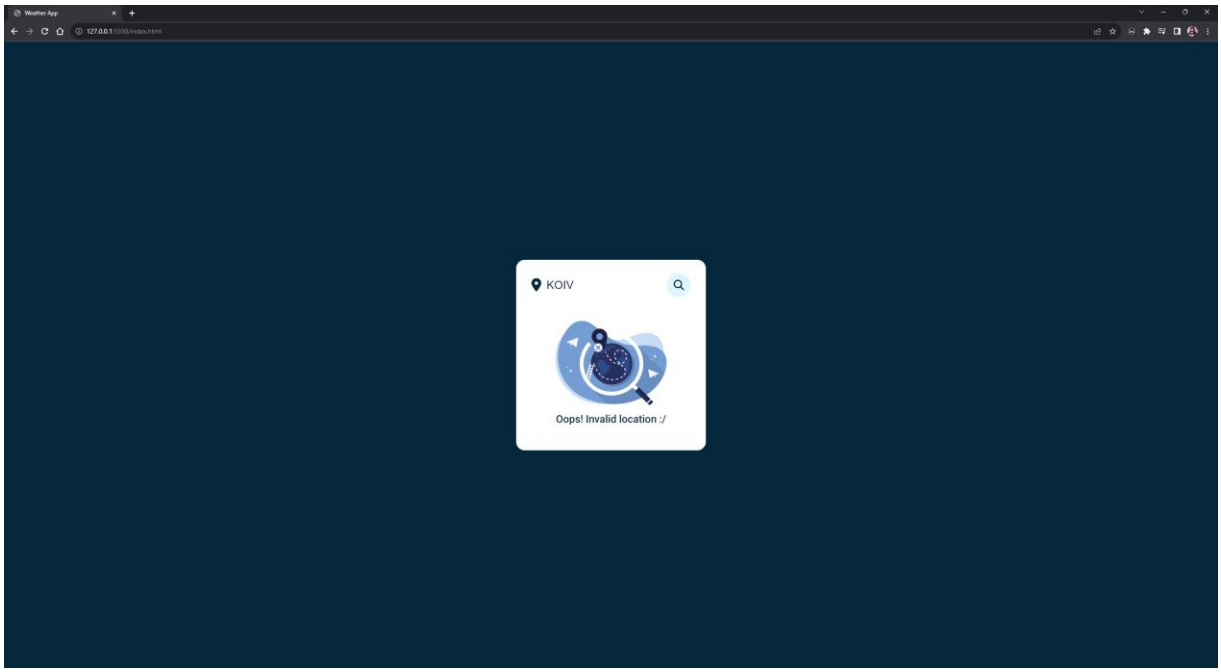


Рисунок 3.12 – Демонстрація некоректного пошуку й реакція на це

Створення погодного веб-додатку – це завдання, яке передбачає поєднання програмування, дизайну та інтеграції з погодним API для отримання актуального інформації.

Процес включає в себе розробку фронтенду для інтерфейсу користувача, забезпечуючи взаємодії з сервером та використання API для отримання прогнозів та інших погодних даних.

Ключові етапи включають вибір технологій, створення бази даних для зберігання інформації, розробку інтерфейсу та забезпечення роботи веб-додатку. Забезпечення швидкості та надійності, а також відповідність дизайну та досвіду користувача – важливі аспекти у створенні успішного веб-додатку.

Висновок

Дослідження технології API у сфері веб-розробки показало її ключову роль у підвищенні функціональності, інтерактивності та динамічності веб-сайтів. У цьому дослідженні було розглянуто різні аспекти веб-розробки, включаючи HTML, CSS та JavaScript, визнаючи їх синергічний зв'язок з API.

HTML, як основа веб-контенту, закладає фундамент для структурування інформації. CSS, завдяки своїм можливостям стилізації, додає естетичної привабливості та покращує взаємодію з користувачем. JavaScript, будучи універсальною мовою сценаріїв, забезпечуючи динамічні та інтерактивні функції. Інтеграція API розширює можливості цих базових мов, надаючи розробникам можливість безперешкодно отримувати дані та маніпулювати ними в режимі реального часу.

Крім того, було досліджено різні категорії та типи веб-сайтів, кожен з яких має унікальні вимоги. Сайти електронної комерції отримують вигоду від API, інтегруючи платіжні системи та системи управління записами. Платформи соціальних мереж використовують API для автентифікації користувачів, обміну контентом і пошуку даних. Освітні сайти можуть використовувати API для інтеграції інтерактивних інструментів та оновлень у режимі реального часу.

У контексті створення погодного веб-додатку API стають особливо важливими. Інтеграція погодного API дозволяє отримувати поточні погодні умови, прогнози та пов'язані з ними дані. Це покращує користувацький досвід, надаючи точну та актуальну інформацію, перетворюючи статичну погодну веб-сторінку на динамічний інформативний інструмент.

Завершуючи дослідження технології API у веб-розробці, очевидно, що API слугують мостом, забезпечуючи безперешкодний зв'язок між різними компонентами веб-додатків. Майбутнє веб-розробки, безсумнівно пов'язане з розвитком і впровадження API, оскільки вони сприяють інноваціям, масштабованості та більш захоплюючому досвіду роботи в Інтернеті.

Перелік посилань

1. "Operating System Concepts" by Abraham Silberschatz, Peter B. Galvin, and Greg Gagne.
2. Measuring API Usability by Steven Clarke
3. API-fication Core Building Block of the Digital Enterprise by Charu Rudrakshi, Amit Varshney, Bharath Yadla, Dr. Rama Kanneganti, Fellow Kiran
4. "iOS Programming: The Big Nerd Ranch Guide" by Christian Keur and Aaron Hillegass
5. "Android Programming: The Big Nerd Ranch Guide" by Bill Phillips and Chris Stewart
6. "Amazon Web Services in Action" by Andreas M. Wittig and Michael Wittig.
7. "Azure Essentials" by Michael Collier and Robin Shahan.
8. "Google Cloud Platform in Action" by Diane Phan, Brian Dorsey, and Mete Atamel.
9. "Building Microservices" by Sam Newman.
10. "IoT Solutions in Microsoft's Azure IoT Suite" by Jim Bennett.
11. "Architecting the Internet of Things" by Dieter Uckelmann, Mark Harrison, and Florian Michahelles.
12. "Mastering Magento" by Bret Williams.
13. "Python for Finance" by Yves Hilpisch.
14. "Shopify Application Development" by Michael Larkin.
15. "Professional WordPress: Design and Development" by Brad Williams, David Damstra, and Hal Stern.
16. "Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14" by Scott Meyers
17. "Web Development with Node and Express" by Ethan Brown
18. "Don't Make Me Think" by Steve Krug
19. "Web Design with HTML, CSS, JavaScript and jQuery Set" by Jon Duckett:
20. "Information Architecture: For the Web and Beyond" by Louis Rosenfeld and Peter Morville:

21. "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow" by Aurélien Géron.
22. "The Elements of User Experience" by Jesse James Garrett:
23. "Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity" by Avinash Kaushik:
24. "Responsive Web Design" by Ethan Marcotte:
25. "The Design of Everyday Things" by Don Norman:
26. "The Pragmatic Programmer: Your Journey to Mastery" by Dave Thomas and Andy Hunt:
27. "The Art of SEO: Mastering Search Engine Optimization" by Eric Enge, Stephan Spencer, and Jessie Stricchiola:
28. "Building Scalable Web Sites" by Cal Henderson:
29. "Lean UX: Designing Great Products with Agile Teams" by Jeff Gothelf and Josh Seiden:
30. Comprehensive documentation on web APIs, including browser APIs.

    https://developer.mozilla.org/en-US/docs/Web/API
31. Official documentation for the Python Standard Library.

    https://docs.python.org/3/library/
32. Official documentation for the Java API.

    https://docs.oracle.com/en/java/
33. Linux manual pages provide detailed information on system calls and library functions.

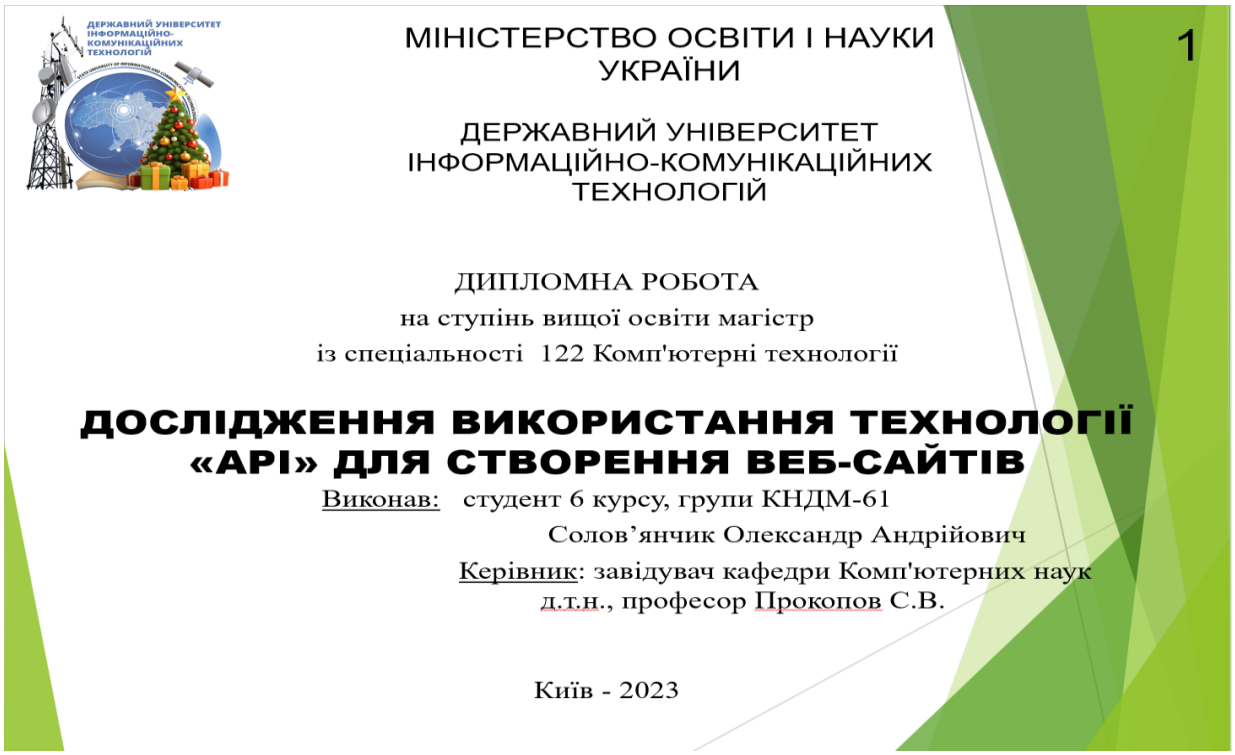
    https://man7.org/linux/man-pages/index.html
34. Official documentation for the Windows API.

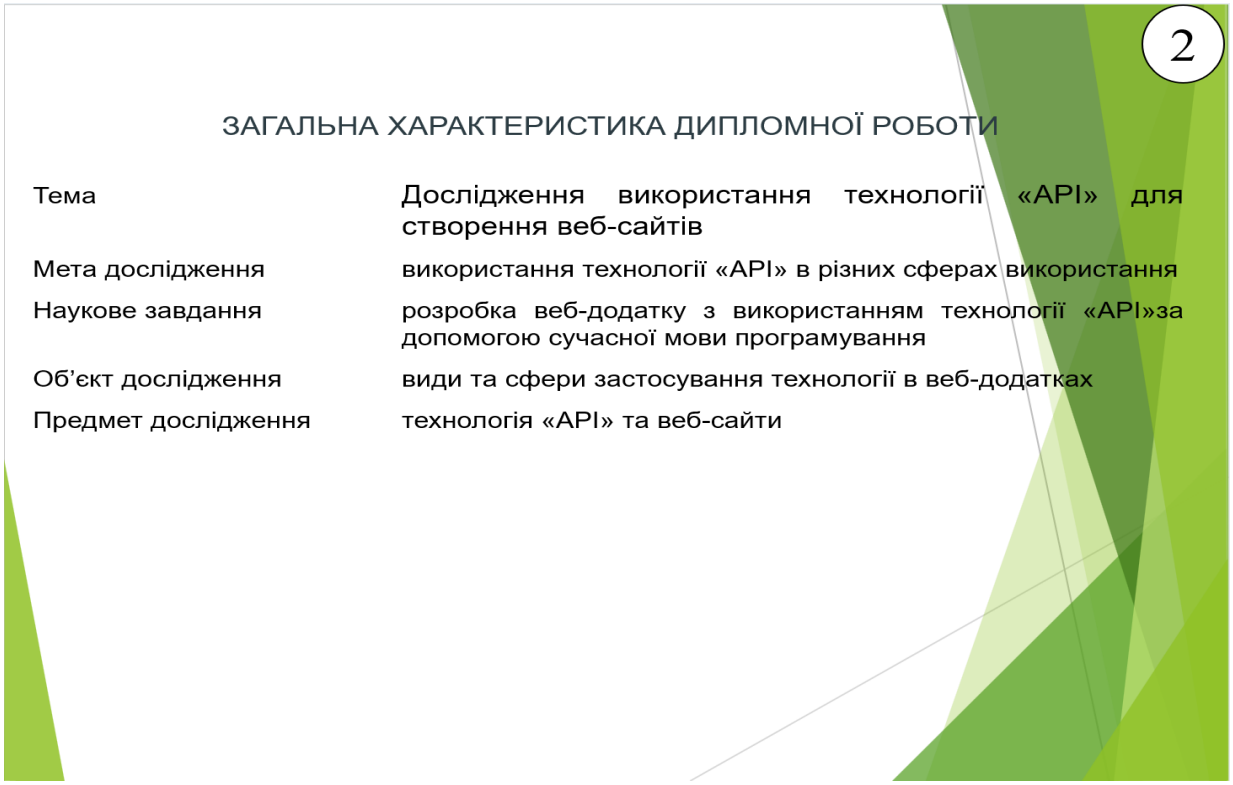
    https://learn.microsoft.com/en-us/windows/win32/api/
35. Documentation for iOS and Android development platforms.
36. Official documentation for Azure, and Google Cloud.
37. Documentation for financial APIs (e.g., Bloomberg API, Alpha Vantage).
38. Documentation for machine learning APIs
39. Documentation for CMS APIs (WordPress REST API, Drupal API).

40. Documentation for OpenWeatherMap APIs
https://openweathermap.org/
41. Documentation for Alpha Vantage APIs
https://www.alphavantage.co/documentation/
42. Documentation for Google Maps API
https://developers.google.com/maps/documentation
43. Documentation for Twitter API
https://developer.twitter.com/en/docs/twitter-api
44. Documentation for Shopify API
https://shopify.dev/docs/api
45. Documentation for News API
https://newsapi.org/docs
46. Documentation for IBM Watson API
https://developer.ibm.com/components/watson-apis/
47. W3C Web Standards Documentation:
https://www.w3.org/
48. Documentation for SCSS
https://sass-lang.com/documentation/

Додаток А

Презентація





Зміст дипломної роботи

- Вступ
- Аналіз технології «API»
- Веб-сайт та його особливості
- Розробка веб-додатку
- Висновки
- Перелік посилань

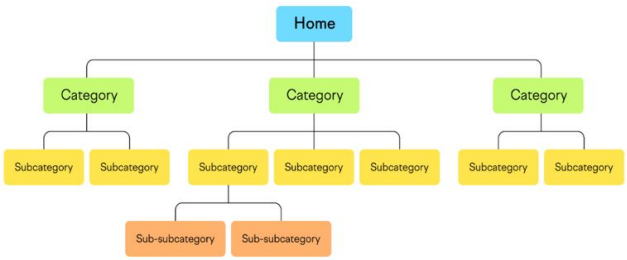
Використання API

- Технологія API має широке застосування в різних сферах програмування. API слугують мостом між різними програмними та апаратними компонентами, дозволяючи їм спілкуватися, обмінюватися даними та співпрацювати не маючи обмежень.
- Кожен день людина дивиться прогноз погоди на своєму телефоні, використовує онлайн карти для побудови маршрутів, розплачується в магазині не підозрюючи що для роботи всього цього використовують API.

Веб-сайт та його особливості

Подорожуючи цифровим світом користувачі стикаються з різними веб-сайтами, що надають офіційну інформацію та послуги, новинними веб-сайтами, що надають їх в режимі реального часу, та веб-сайтами-портфоліо, що демонструють творчі досягнення окремих людей.

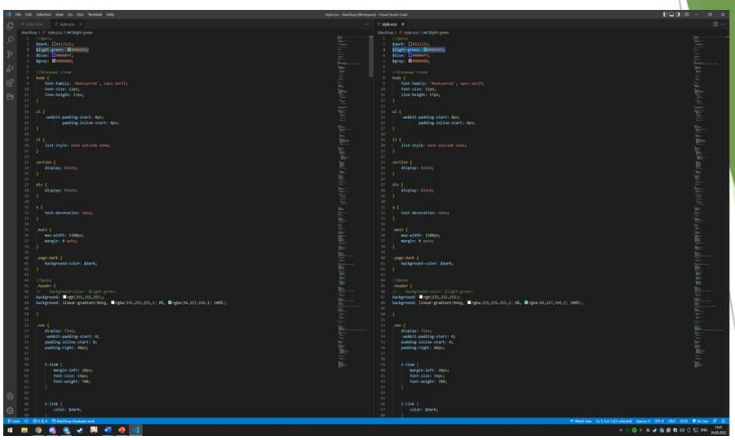
Website Architecture Example



Структура інформації визначає, як інформація розподіляється по різним частинам сайту, як організовані меню, категорії, підрозділи.

Навігація визначає систему переходів між різними сторінками та розділами веб-сайту. Ефективна навігація сприяє зручності користування та пошуку інформації.

Редактор коду VS Code

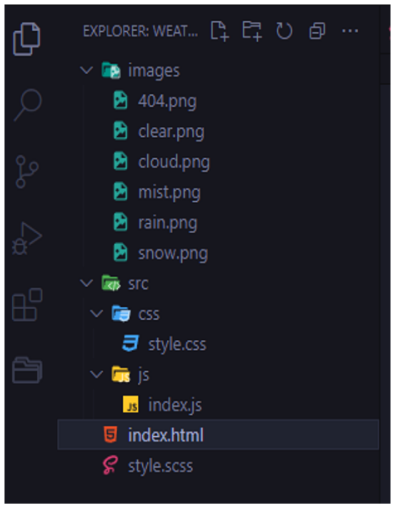


Visual Studio Code (VSCode) – популярний та універсальний редактор вихідного коду, розроблений компанією Microsoft.

VSCode набув широкого розповсюдження серед розробників завдяки своїм потужним можливостям, розширюваності та крос-платформною підтримкою.

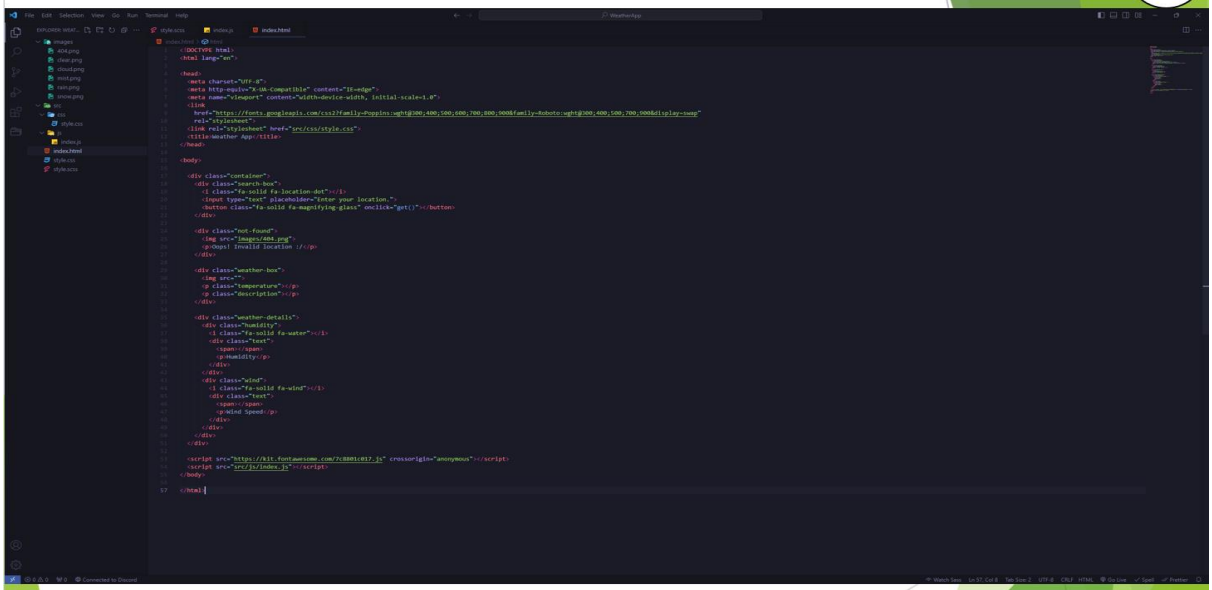
Макет веб-додатку

На цьому рисунку зображено макет веб-додатку для відображення інформації про погоду в запитуваному місті в даний момент часу. На макеті представлено початковий вигляд веб-додатку, коли користувачу необхідно ввести необхідне для нього місто, та результат виводу цієї інформації. Також зображено різні погодні умови в залежності від погодних умов в шуканому місті.



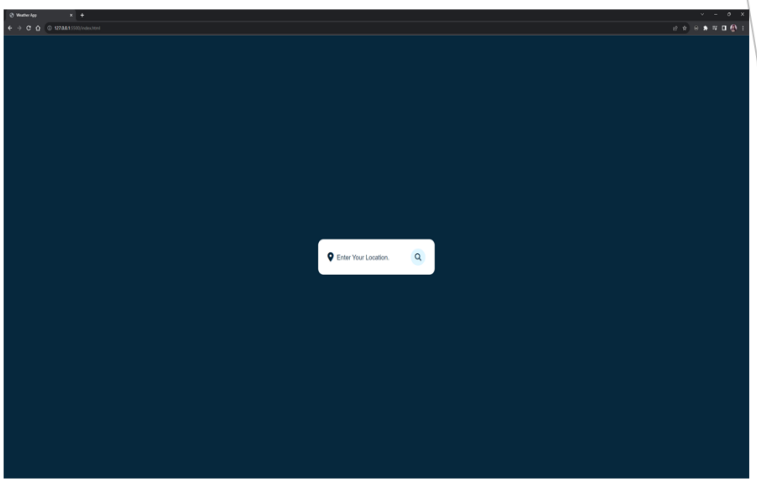
Файлова структура проекта

Кожен файл цього проекту лежить в конкретному місці, відповідаючи його типу. Так наприклад зображення використовувані в цьому проекті лежать в папці «images», а CSS та JavaScript файли в відповідних вже ним папкам.

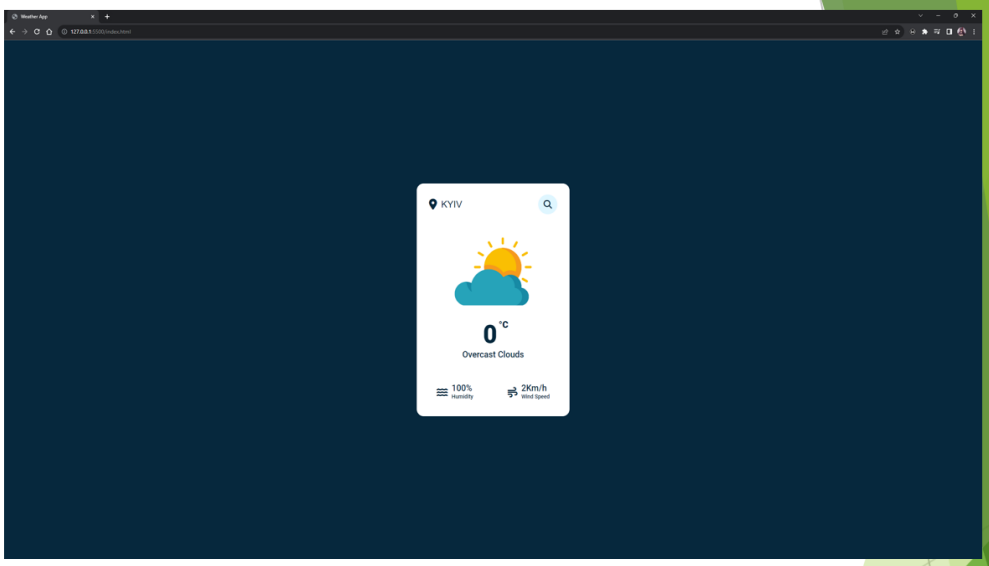


Головним файлом цього проекту є «index.html». В цьому файлі створюється розмітка самого веб-додатку, створюються елементи та наповнюються контентом. До цього файлу підключені файли «style.css» та «index.js».

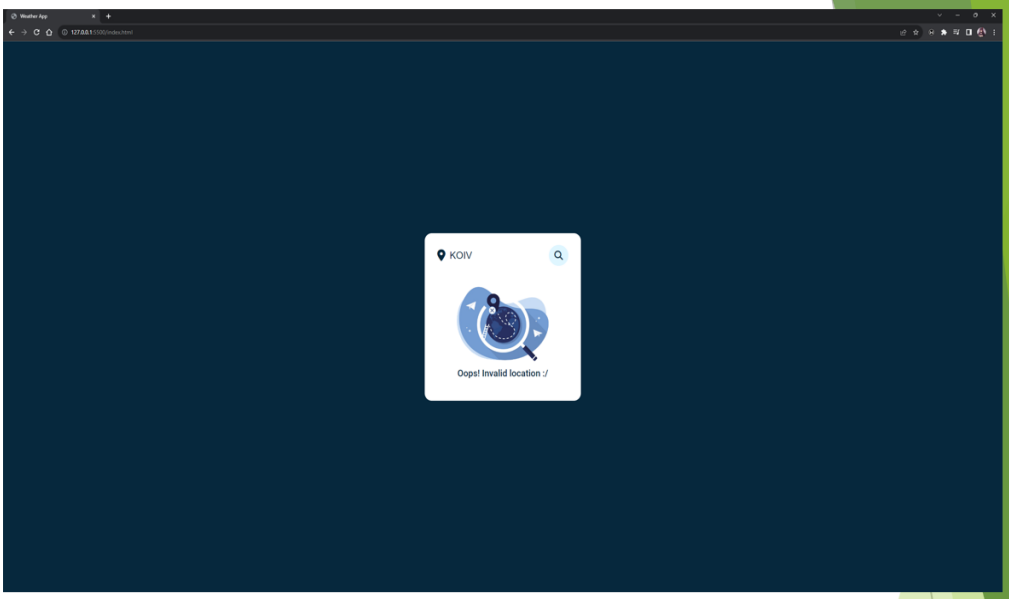
Демонстрація проекту



На цій сторінці користувач може ввести необхідне йому місто для відображення погодних умов.



За заданим місто користувач може дізнатися погодні умови в місті, а саме температуру в місті, її стан, швидкість вітру та вологість повітря.



На цьому рисунку зображено веб-додаток в тому випадку, якщо користувач ввід неправильну назву міста.

Створення погодного веб-додатку – це завдання, яке передбачає поєднання програмування, дизайну та інтеграції з погодним API для отримання актуального інформації.

Процес включає в себе розробку фронтенду для інтерфейсу користувача, забезпечуючи взаємодії з сервером та використання API для отримання прогнозів та інших погодних даних.

Ключові етапи включають вибір технологій, створення бази даних для зберігання інформації, розробку інтерфейсу та забезпечення роботи веб-додатку. Забезпечення швидкості та надійності, а також відповідність дизайну та досвіду користувача – важливі аспекти у створенні успішного веб-додатку.

Висновки

Дослідження технології API у сфері веб-розробки показало її ключову роль у підвищенні функціональності, інтерактивності та динамічності веб-сайтів. У цьому дослідженні було розглянуто різні аспекти веб-розробки, включаючи HTML, CSS та JavaScript, визнаючи їх синергічний зв’язок з API.

HTML, як основа веб-контенту, закладає фундамент для структурування інформації. CSS, завдяки своїм можливостям стилізації, додає естетичної привабливості та покращує взаємодію з користувачем. JavaScript, будучи універсальною мовою сценаріїв, забезпечуючи динамічні та інтерактивні функції.