МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
(повне найменування інституту, назва факультету)

Кафедра комп'ютерної інженерії та електромеханіки
(повна назва кафедри)

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА

на тему

РОЗРОБКА WEB – ЗАСТОСУНКУ ДЛЯ КОРОТКОСТРОКОВОГО ПРОГНОЗУВАННЯ ПОГОДИ

Виконав: студент групи БКІ-19

спеціальності 123 «Комп'ютерна інженерія
(шифр і назва спеціальності)

Стрічко О.С.
(прізвище та ініціали)

Керівник д.т.н., проф. Злотенко Б. М.

(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Електромеханіка»

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Злотенко Б.М.

"_____" _____2023 року

З А В Д А Н Н Я

НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Стрічку Олексію Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема дипломної бакалаврської роботи **Розробка web – застосунку для короткострокового прогнозування погоди**

Науковий керівник роботи Злотенко Борис Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
д.т.н., професор

затверджені наказом вищого навчального закладу від 15.03.2023 № 75-уч.

2. Строк подання студентом роботи 1 червня 2023 року

3. Вихідні дані до дипломної бакалаврської роботи Розробка web – застосунку для короткострокового прогнозування погоди .

4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): 1. Провести аналітику щодо web - розробки. 2. Розробити застосунок, який допоможе спрогнозувати погодні явища на цілу неділю .

5. Дата видачі завдання 10.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної бакалаврської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	01.02.2023	
2	Розділ 1. Аналіз існуючого інструментарію розробки web-застосунків	15.02.2023	
3	Розділ 2. Вибір засобів створення web-застосунку для прогнозування погоди	15.03.2023	
4	Розділ 3. Програмна реалізація web–застосунку для прогнозування погоди.	05.04.2023	
5	Розділ 4. Практичне використання web–застосунку для прогнозування погоди		
6	Висновки	10.05.2023	
7	Оформлення дипломної бакалаврської роботи (чистовий варіант)	20.05.2023	
8	Здача дипломної бакалаврської роботи на кафедру для рецензування (за 14 днів до захисту)	25.05.2023	
9	Перевірка дипломної бакалаврської роботи на наявність ознак плагіату (за 10 днів до захисту)	28.05.2023	
10	Подання дипломної бакалаврської роботи на затвердження завідувачу кафедри (за 7 днів до захисту)	05.06.2023	

Студент _____ Стрічко О. С.
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Злотенко Б. М.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Стрічко О. С. Розробка web – застосунку для короткострокового прогнозування погоди. : дипломна бакалаврська робота за спеціальністю 123 Комп'ютерна інженерія / О. С. Стрічко ; наук. кер. Б. М. Злотенко. – Київ : КНУТД, 2023. – 82 с.

Мета роботи полягає у розробці web – застосунку який допоможе перетворити незрозумілі та складні для звичайної людини дані в сучасний сервіс прогнозування погоди та іншого. Прогнозування погоди - це застосування науки і техніки для прогнозування стану атмосфери для даного місця і часу. Прогнози погоди робляться шляхом збору кількісних даних про поточний стан атмосфери в даному місці і використання метеорології для прогнозування зміни атмосфери. Колись прогноз погоди розраховували вручну, розраховуючи на змінення атмосферного тиску, поточних погодних умов і т. д. На сьогоднішній час, практично всю роботу робить комп'ютер. Основними мовами програмування, на яких буде розроблений саме мій застосунок, буде JavaScript, CSS, Html.

Ключові слова: web, css, html, javascript, атмосфера

ABSTRACT

Strcihko O.S Development of a web application for short-term weather forecasting. Bachelor's thesis in the specialty 123 Computer Engineering - Kyiv National University of Technology and Design, Kyiv, 2023.

The purpose of the work is to develop a web application that will help to transform incomprehensible and complex data for an ordinary person into a modern weather forecasting service and more. Weather forecasting is the application of science and technology to predict the state of the atmosphere for a given place and time. Weather forecasts are made by collecting quantitative data about the current state of the atmosphere in a given location and using meteorology to predict changes in the atmosphere. Once upon a time, weather forecasts were calculated manually, relying on changes in atmospheric pressure, current weather conditions, etc. Today, almost all the work is done by a computer. The main programming languages in which my application will be developed will be JavaScript, CSS, Html.

Keywords: web, css, html, javascript, atmosphere

ЗМІСТ

ВСТУП

Прогнозування погоди - це застосування науки і техніки для передбачення стану атмосфери у певному місці та у певний час. Прогнозування погоди здійснюється шляхом збору кількісних даних про поточні атмосферні умови в тому чи іншому місці та використання метеорології для прогнозування змін в атмосфері. Колись прогнозування погоди здійснювалося вручну шляхом розрахунку змін атмосферного тиску та поточних погодних умов. Сьогодні більшу частину цієї роботи виконує комп'ютер.

В наш час галузь погоди давно стало невід'ємною частиною нашого життя. Сьогодні можна з упевненістю сказати, що інформація, що надається Метеорологічним управлінням, з кожним днем стає дедалі точнішою, і це вже не питання "ворожити чи не гадати". З давніх часів люди намагалися пророкувати погоду. Серед народних знань, які незаслужено забуті або просто ігноруються, є розроблена століттями система прогнозування погоди, яка ґрунтується на спостереженнях і зафіксована у вигляді прислів'їв, приказок та прикмет. Ця система широко використовується і сьогодні. Необхідність передбачати погоду тісно пов'язана з господарською діяльністю людини, тому пророки як система розвивалися переважно серед аграрних народів. Це тим, що добробут людей залежало від правильного вибору часу посіву і посадки. Зв'язок природних явищ з поведінкою птахів, тварин та рослин створив систему передбачень, корисну практично для всіх видів людської діяльності.

Погода є важливим питанням для багатьох галузей економіки та повсякденного життя людини. Сьогодні зростаючі обсяги даних та обчислювальні потужності роблять розробку ефективних інструментів прогнозування погоди ще більш актуальною. У цьому контексті веб-застосунки для прогнозування погоди є потужними інструментами, які дозволяють користувачам швидко і легко отримувати актуальну інформацію

про погоду в будь-якій точці світу. У цьому дипломі розглядається розробка веб-застосунку для прогнозування погоди, за допомогою якого користувачі зможуть отримувати інформацію про погоду у своєму регіоні. У програмі, що розробляється, будуть використовуватися новітні технології веб-розробки, а також потужні алгоритми та методи машинного навчання для прогнозування погоди. Майбутні програми можуть бути використані в різних секторах економіки, таких як сільське господарство, будівництво та транспорт.

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧОГО ІНСТРУМЕНТАРІЮ РОЗРОБКИ WEB-ЗАСТОСУНКІВ

1.1. Сучасний стан і тенденції розвитку web-програмування

Web-програмування є однією з областей розробки програмного забезпечення, що найбільш швидко зростають. З появою нових технологій веб-програмування стало більш складним та розширеним процесом, що ставило вимоги до високих знань у програмуванні та технологіях (рис. 1.1).

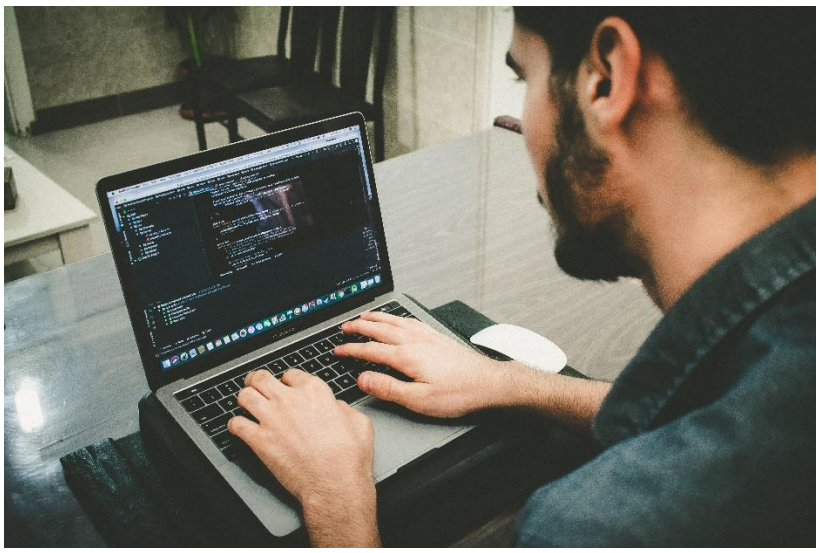


Рисунок 1.1 – Web – програмування

Однією з найважливіших тенденцій розвитку веб-програмування є зростання популярності фреймворків і бібліотек, що дозволяють розробникам ефективно створювати складні веб-додатки. Найбільш відомі фреймворки, такі як React, Angular та Vue.js, пропонують розробникам безліч інструментів, що спрощують розробку та підтримку веб-додатків (рис. 1.2) [11].

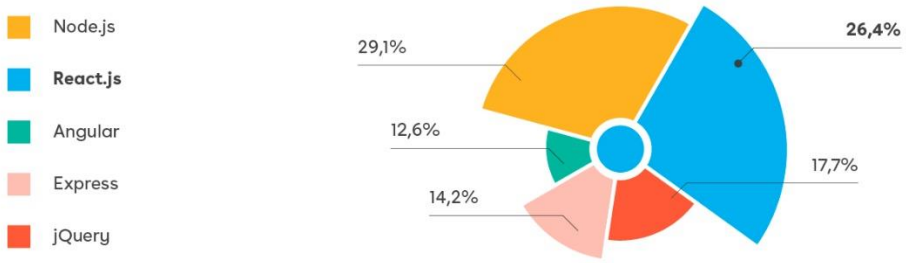


Рисунок 1.2 – Графік найпопулярніших фреймворків

Іншою важливою тенденцією є зростання популярності веб-застосунків, які можна використовувати на різних пристроях, таких як смартфони та планшети. Тому багато веб-розробників зосереджуються на створенні веб-застосунків, що працюють на різних платформах, що дозволяє їм досягти максимального охоплення користувачів (рис. 1.3).

Ще однією важливою тенденцією є збільшення кількості веб-додатків, які використовують штучний інтелект та машинне навчання. Багато веб-програм використовують інструменти машинного навчання для вирішення проблем, пов'язаних з обробкою великих обсягів даних, прогнозуванням та іншими аналітичними завданнями [17].

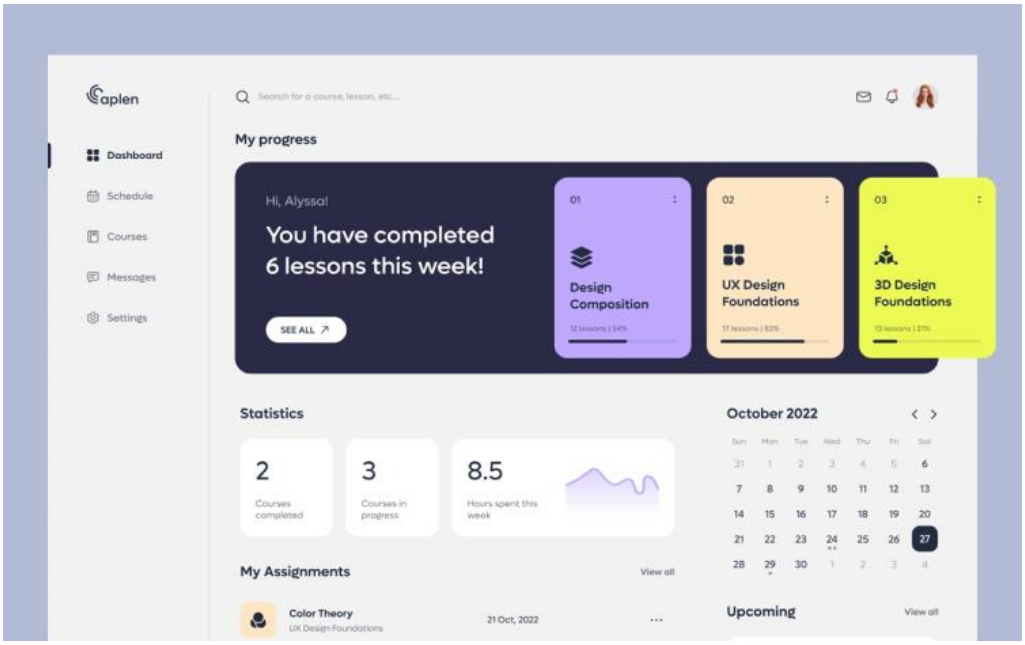


Рисунок 1.3 – Приклад сторінки застосунка

Для створення інтерактивних та компактних веб-додатків необхідно розуміти багато сучасних веб-стандартів.

Швидке зростання Інтернету за останні кілька років призвело до збільшення кількості нових онлайн-сервісів. З початкових функцій та базових послуг, доступних користувачам (пошта та Інтернет), інтернет перетворився на велике всесвітнє павутиння з великою кількістю різних послуг, технічних рішень та фізичним місцем збору ділових організацій з усього світу.

Збільшилася кількість компаній, які звертаються до електронної комерції. Відкривались домени своїх компаній, створювались свої веб-сайти, рекламувались свої товари та послуги в Інтернеті та здійснюють онлайн-продаж [18].

Основні ролі в технічній частині роботи належать веб-програмістам та розробникам. Їхні завдання:

- створити веб-сайт
- макет
- підключити їх до відповідної бази даних

- написання веб-сервлетів
- створити веб-сервіс
- встановити інтернет-з'єднання
- створюйте веб-програми для різних цілей

1.2 Класифікація web-застосунків за призначенням

Веб-програми є інтерактивними програмами і складаються з двох частин:

- Один завантажується у браузер, коли користувач вводить відповідну команду.
- Другий розміщується на веб-сервері, який містить усі дані, вбудовані у програму. Він отримує запити від користувачів, шукає інформацію та надсилає її до браузера.

Для більшості клієнтів веб-застосунок нічим не відрізняється від веб-сайту. Але якщо останній надає більше інформації, то веб-додаток має багато вбудованого функціоналу — від роботи з графіками та таблицями до онлайн-замовлення.

Переваги веб-додатків

- Веб-застосунки не потрібно встановлювати в пам'ять пристрою;
- Оновлення програмного забезпечення відбуваються автоматично та централізовано, тому користувачам не потрібно про них думати.
- Всі користувачі мають доступ до однієї версії веб-програми, що дозволяє уникнути непорозумінь.
- Веб-додаток розроблено таким чином, що його можна відкрити з будь-якого браузера та будь-якої операційної системи.

- Веб-додатки вимагають менше ресурсів (робочої сили та обладнання) для експлуатації та обслуговування, що призводить до економії коштів як для підприємства, так і для кінцевого користувача.
- Програми прості у використанні та не вимагають спеціальних знань чи навичок, що робить їх придатними для широкого кола аудиторії.
- Веб-додатки економічно ефективні і дешевші за традиційні програмні рішення, тому їх можуть використовувати підприємства з обмеженим бюджетом.

Типи веб-додатків:

- Асоціація індустрії послуг, SPA (Single Page Application) - це інтерактивна програма, яка завершена на одній сторінці - не просто одна сторінка, а повноцінна програма (рис. 1.4).

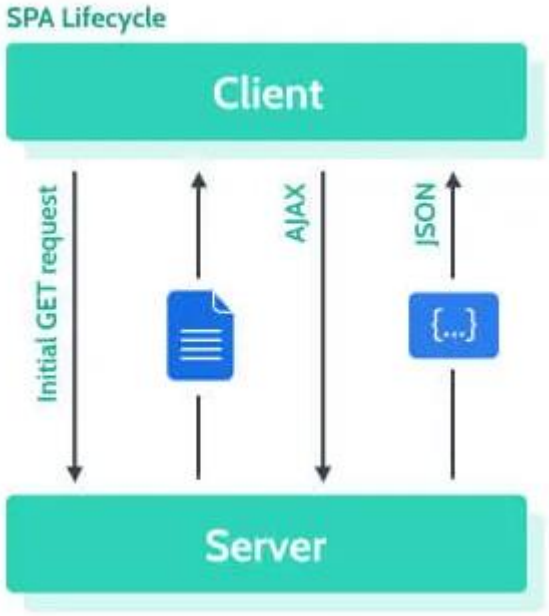


Рисунок 1.4 – Зовнішний вигляд Single Page Application

Тому інформаційний сайт може складатися з однієї сторінки, але це, насправді, не SPA. В односторінковому веб-застосунку користувач залишається на одній сторінці навіть при перемиканні між вкладками. Завантажується та оновлюється лише необхідний контент, що відповідає швидкості SPA [4].

Прикладом односторінкової програми є Gmail (рис. 1.5) [7].

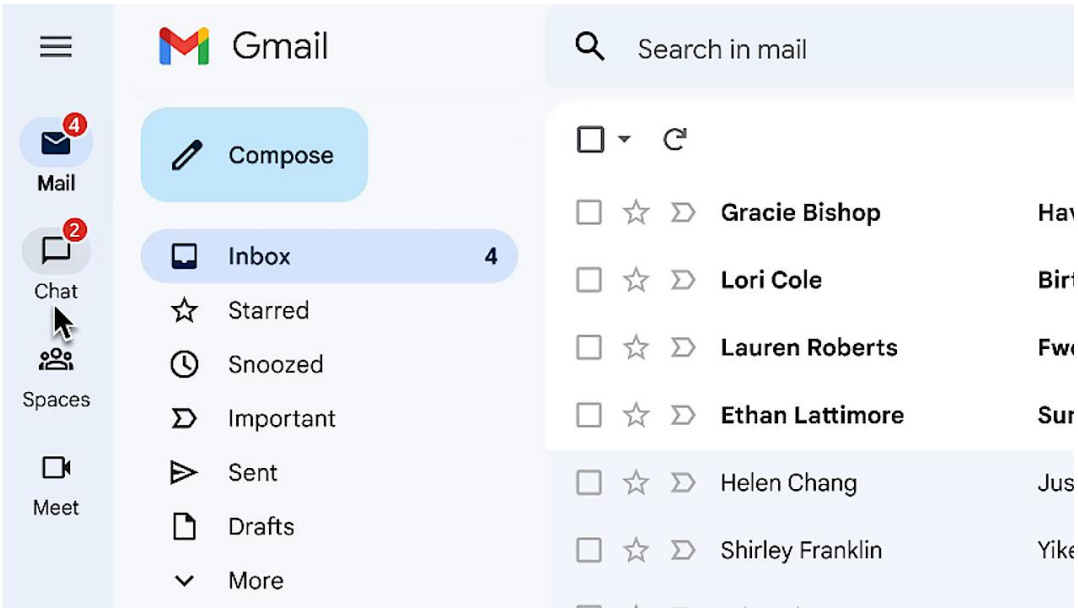


Рисунок 1.5 – Застосунок gmail

Зверніть увагу - при перемиканні між списками повідомлень адреса сторінки не змінюється. Це ключова особливість SPA.

Основною мовою створення SPA є JavaScript; Для створення невеликих односторінкових програм можна використовувати бібліотеку jQuery. Однак цей варіант не є ідеальним для великих проектів; краще використовувати фреймворки Vue, React чи Angular.

- Багатосторінковий додаток (MPA) - це традиційний багатосторінковий веб-додаток. Коли користувач взаємодіє з веб-сайтом, завантажується нова сторінка HTTP. Тому обмін даними відбувається повільніше, ніж у SPA. Це особливо важливо, якщо є проблеми з

інтернет-з'єднанням або хостингом сайту. Прикладами MPA є інтернет-магазини, такі як Rozetka та Amazon.

- PWA. - прогресивне програмне забезпечення схоже на нативні комп'ютерні та мобільні програми за своєю функціональністю, можливостями та якістю користувальницького досвіду.

Немає чіткої межі між не-PWA і PWA додатками. Однак, можна виділити деякі характеристики (рис. 1.6).

Рисунок 1.6 – Програмне забезпечення PWA

Зокрема, PWA повинні включати проксі-шар (Service Worker) та маніфест веб-додатку. По суті, браузер діє як віртуальна машина, на якій запускається веб-програма, подібно до того, як Windows запускає exe-файл або Android запускає apk.

Service Worker - це проксі-шар між стороною сервера та стороною клієнта. Він розміщується в браузері і запити проходять через нього (рис. 1.7). Таким чином, існує два зовнішні шари: один, де написаний інтерфейс, і інший, де написана логіка. Це дозволяє запускати повноцінні програми для Інтернету - Service Workers зазвичай пишуться на чистому JS [17].

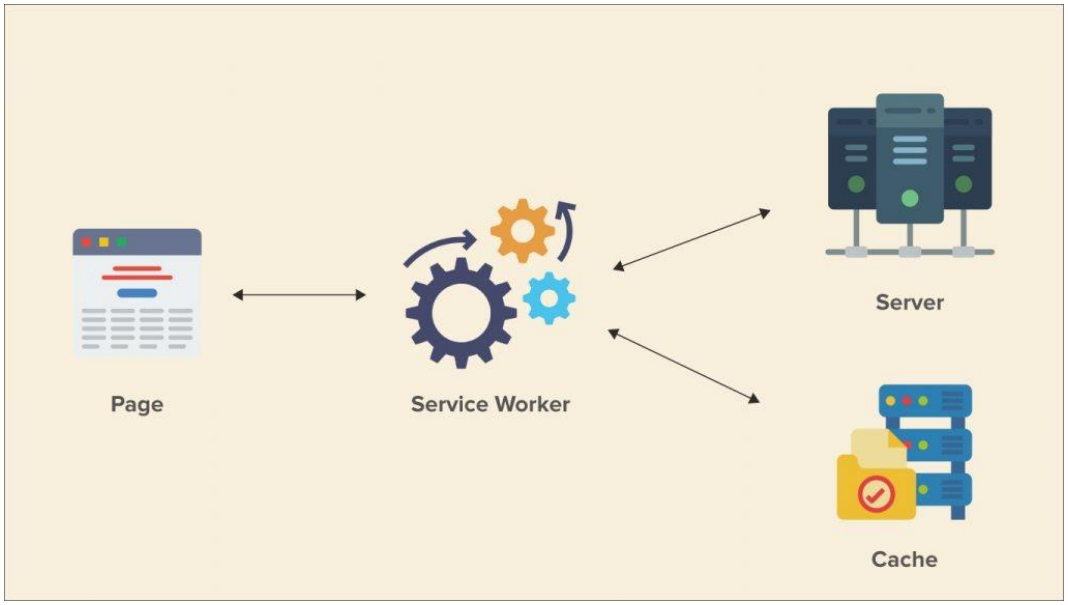


Рисунок 1.7 – Принцип роботи Service Worker

1.3 Особливості web-програмування

Одна людина не може бути професіоналом у всьому, тому програмісти у web-сфері спеціалізуються на вузьких напрямках:

frontend (клієнтська частина) розробники;

backend (серверна частина) розробники;

fullstack-розробники.

Frontend-розробники займаються відображенням інформації у браузері. Це ті люди, які працюють у зв'язці з дизайнерами та відповідають за коректність макета, плавність анімації, інфографіку на сайті – за все, що бачать користувачі, за frontend-частину (бік) продукту. А вона взаємодіє з браузером та використовує його для виконання певних завдань. Технології (мови), які розуміє браузер, – це HTML, CSS та JavaScript. Все це треба знати web-програмісту, що працює з frontend-частиною. Якщо у вас поплив текст або кнопка тікає від вас при наведенні, знайте, що у всьому винен фронтендер.

Backend-розробник займається реалізацією логіки, прихованої від клієнтів. Це може бути аутентифікація користувачів, балансування навантаження на сервер, запит запиту фронтенда з бази даних. Backend-розробники іноді взаємодіють із системними адміністраторами, оскільки працездатність сервера дуже важлива. Наразі існує безліч мов загального призначення, які використовуються на сервері. Найпоширеніший на даний момент – це PHP, для enterprise-рішень стандартне рішення – Java (не плутати з JavaScript), а також Python, Ruby і набирає популярності Golang. Якщо заповнити форму і при натисканні на кнопку відправки раптом з'являється повідомлення, що щось пішло не так, то, швидше за все, винен винний бекендер.

Fullstack-програміст - це людина, яка відповідає за всі етапи розробки web-програми, тобто вона поєднує обов'язки і frontend-, і backend-розробника. Можна бути майстром на всі руки, а можна бути дуже ревним і вирости з backend/frontend-розробника вище. У невеликих підприємствах або на фрілансі fullstack-розробник – це іноді ще системний адміністратор, і дизайнер в одній особі. Таких спеціалістів зазвичай називають web-майстер. Найчастіше fullstack-програмісти стають архітекторами. Це найвища посада [12].

Плюси:

- затребуваність ринку праці;
- творча робота, оскільки кожна завдання унікальна, й у її вирішення є кілька способів;
- можливість поєднання з навчанням;
- не завжди потрібен диплом;
- можливість стати спеціалістом у молодому віці;
- свобода у прийнятті рішень (як вирішувати те чи інше завдання, вирішує сам web-програміст);

- можливість віддаленої роботи.

Мінуси:

- ненормований робочий день;
- сидяча робота;
- велике навантаження на зір, що може призвести до його погіршення;
- іноді доводиться працювати «у стіл» і переробляти, оскільки вимоги змінюються на ходу;
- необхідно постійно навчатися, тому що все швидко застаріває – це мінус тільки для тих, хто не любить навчатися **[18]**.

1.4 Мови програмування для розробки web-застосунків

Перші кроки з вивчення веб-розробки рекомендується починати з освоєння основ HTML. Знаючи його, можна працювати з інтерфейсами користувача. Також буде видно перші результати написання коду. Неможливо освоїти розробку веб-сайтів, не знаючи HTML.

Наступним кроком після вивчення мови розмітки буде JS. JavaScript є мовою програмування для веб-розробки, яку використовують усі сучасні браузери. Майже будь-який веб-проект має код, написаний на JS. Останнім часом ця мова набирає популярності на інших платформах, включаючи сервери та пристрої.

Знаючи HTML та JS, можна приступати до вивчення CSS. Без цього неможливо настроїти зовнішній вигляд сторінок. Є безліч рішень та підручників з вивчення CSS. Але щоб досягти справжнього успіху, однієї теорії недостатньо. Для закріплення знань потрібна практика. У мережі можна знайти різні репозиторії для роботи з кодом.

Існує безліч мов програмування, які можуть бути використані для розробки web-додатків. Деякі з них більш популярні та поширені серед

розробників. Основні мови програмування для розробки web-додатків на сьогоднішній день:

- JavaScript
- PHP
- Python
- Ruby
- Java

Ці мови програмування є лише кількома з багатьох інших, які можуть бути використані для розробки веб-додатків.

Висновок до розділу 1:

Перщий розділ дав зрозуміти що, аналіз існуючого інструментарію розробки web-додатків дозволяє визначити найбільш підходящі технології створення високоякісних і ефективних додатків. При розробці web-додатків важливо враховувати потреби користувачів та забезпечувати їх зручність та комфорт.

Існує безліч інструментів та технологій, які можуть використовуватись для розробки веб-додатків, таких як фреймворки, бібліотеки, мови програмування та інші. Вибір інструментарію залежить від вимог проекту та відповідальності розробника за результат.

При аналізі існуючого інструментарію важливо враховувати такі фактори, як продуктивність, швидкість розробки, легкість використання та підтримка. Найбільш популярні технології мають велику спільноту розробників, що забезпечує розвиток та підтримку цих технологій.

Однією з основних вимог до інструментарію є можливість інтеграції з іншими технологіями та розширення функціональності програми. Важливо вибрати технології, що дозволяють розширювати функціональні можливості програми та забезпечувати її стабільність та безпеку.

Таким чином, аналіз існуючого інструментарію дозволяє вибрати найбільш підходящі технології для розробки web-додатків та забезпечити їх ефективність та високу якість.

РОЗДІЛ 2. ВИБІР ЗАСОБІВ СТВОРЕННЯ WEB-ЗАСТОСУНКУ ДЛЯ ПРОГНОЗУВАННЯ ПОГОДИ

2.1 Формулювання вимог до web-застосунку прогнозування погоди

Веб-програми для прогнозування погоди є корисними інструментами для людей, які планують свою діяльність на день або тиждень наперед. Завдяки таким програмам користувачі можуть дізнатися погодні умови у своєму регіоні та краще спланувати свій робочий та вільний час.

Прогнози погоди дозволяють користувачам отримувати актуальну інформацію про температуру, вітер, опади та інші погодні параметри на різні періоди. Користувачі можуть отримувати прогнози на найближчі кілька годин, днів або навіть тиждень наперед, що дозволяє їм краще планувати свою діяльність.

Цікаво, що веб-програми можуть бути корисними не тільки для бізнесу. Некомерційні організації теж часто потребують веб-додатку. Так, найпопулярніші останнім часом електронні щоденники – яскравий приклад веб-додатків некомерційного характеру. З допомогою електронних щоденників взаємодія між школою, кожним окремим учнем та її батьками стає прозорішим.

Веб-програми для прогнозування погоди також корисні для людей з особливими мете чутливими станами. Наприклад, люди, які страждають на алергію, можуть дізнатися рівень вмісту пилку в повітрі і відповідним чином спланувати свої дії **[1]**.

У зв'язку зі зміною клімату веб-програми для прогнозування погоди стають все більш популярними та важливими для людей. У зв'язку зі зміною клімату веб-програми для прогнозування погоди стають все більш популярними та важливими для людей. Вони дозволяють людям дізнатися погодні умови в різних регіонах та в різний час доби та підготуватися до можливих погодних збурень (рис. 2.1).

Рисунок 2.1 – Випадок погодних збурень

Наявність подібних властивостей дозволяє використовувати веб-додатки та веб-системи максимально ефективно та зручно. Наприклад, завдяки властивості масштабованості можна без внесення кардинальних змін розширювати веб-систему для роботи з числом користувачів, що постійно зростає, додавати в неї нові функції. У свою чергу завдяки зручному розгортанню процес переходу компанії на роботу з тим чи іншим веб-додатком займає мінімум часу, а в умовах великого завантаження це може бути важливо [10].

2.2 Опис принципу роботи web-застосунку

Прогнозування погоди – це процес складання науково та технічно обґрунтованих припущень про майбутній стан приземного шару атмосфери у певному місці. Упродовж тисячоліть люди намагалися передбачити погоду, але офіційне прогнозування з'явилося лише у 19 столітті. Прогнозування погоди включає збирання кількісних даних про поточні атмосферні умови та наукове розуміння атмосферних процесів (рис. 2.2).

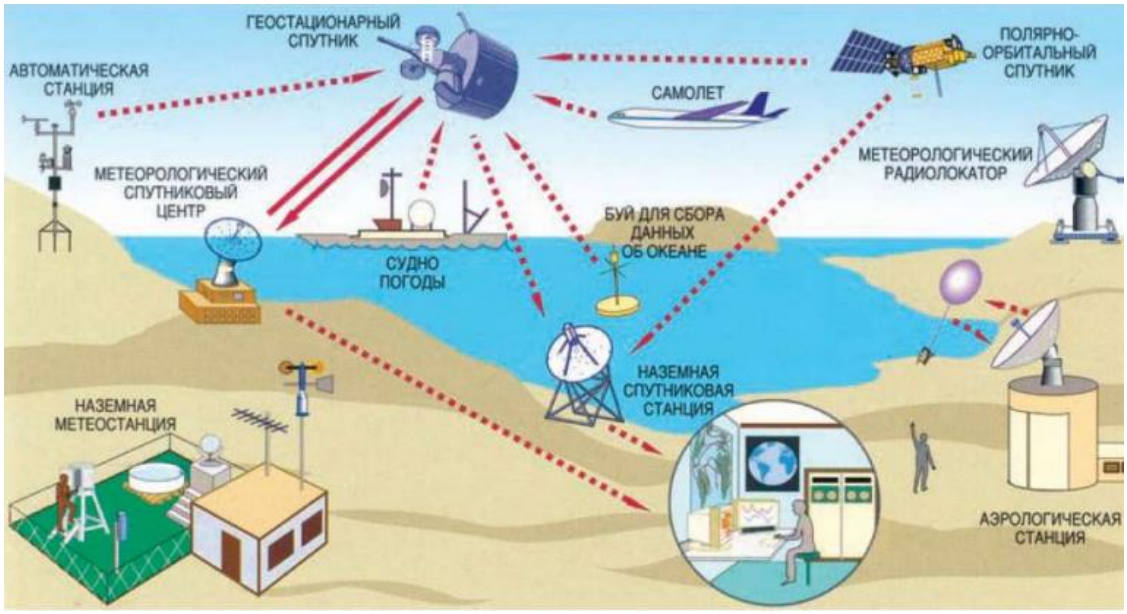


Рисунок 2.2 – Дослідження погодних явищ

Раніше прогнози в основному ґрунтувалися на атмосферному тиску, поточних погодних умовах та стані неба, але тепер використовуються моделі прогнозування. Участь людини необхідна вибору найбільш підходящої моделі прогнозу, де ґрунтуватимуться майбутні прогнози. Люди повинні вміти вибирати шаблон моделі, розуміти взаємозв'язок між віддаленими подіями та знати принципи та характеристики обраної моделі. Складна природа атмосфери, необхідність у потужних комп'ютерах на вирішення рівнянь, що описують атмосферу, наявність помилок при вимірі початкових умов і неповне розуміння атмосферних процесів знижують точність прогнозів. Чим більша різниця між поточною датою та датою, на яку складається прогноз (діапазон прогнозу), тим нижча точність. Використання декількох моделей для отримання результатів зменшує помилки та дає найімовірніший результат [13].

Прогнози погоди використовуються багатьма людьми. Штормові попередження – важливі прогнози, оскільки вони використовуються для захисту життя та майна. Прогнози температури та опадів важливі для сільського господарства. Прогнози температури необхідні працівникам мереж централізованого теплопостачання для прогнозування попиту тепло.

Щодня люди використовують прогнози погоди, щоб вирішити, що вдягнути на день. Прогнози дощу, снігу та сильного вітру використовуються для планування роботи та відпочинку на природі.

До поширених погодних явищ на Землі відносяться вітер, хмари, опади (дощ, сніг тощо), туман, грози, пильні бурі та хуртовини. Рідко трапляються й такі стихійні лиха, як торнадо та урагани. Багато погодні явища відбуваються в тропосфері (нижній частині атмосфери).

Кут падіння сонячного світла залежить від широти та відстані від океану, що призводить до відмінностей у фізичних властивостях повітряних мас. Висотні струменеві течії існують через велику різницю температур між арктичним і тропічним повітрям. Барикадні форми рельєфу, такі як субтропічні циклони у середніх широтах, формуються хвилями у зонах висотних струминних течій. Через нахил земної осі до площини орбіти кут падіння сонячних променів змінюється від періоду до періоду. У середньому річна температура лежить на поверхні Землі коливається не більше ±40°C. Зміни орбіти Землі протягом сотень тисяч років впливають на кількість та розподіл сонячної енергії на планеті та визначають довгостроковий клімат.

Різниця температур лежить на поверхні Землі викликає різницю у атмосферному тиску. Нагріте повітря розширюється, зменшуючи тиск та щільність повітря. Створюється горизонтальний градієнт тиску, повітря рухається у бік нижчого тиску, породжуючи вітри. Сили Коріоліса призводять до деякого зміщення руху повітря (праворуч у північній півкулі та ліворуч у південній півкулі). Приклад простої погодної системи є прибережні вітри; прикладом складної системи є басейн Хедлі.

Атмосфера - це складна система, тому невеликі зміни в одній частині можуть вплинути на всю систему. Протягом усієї історії людства завжди робилися спроби контролювати погоду. Доведено, що діяльність людини, така як сільське господарство та промисловість, у тій чи іншій мірі впливає на погоду. Прогнозування погоди – це процес складання науково та технічно

обґрунтованих припущень про майбутній стан атмосфери у певній точці чи регіоні Землі.

Вивчення погоди інших планетах корисно розуміння принципів зміни погоди Землі. Велика червона пляма Юпітера, добре відомий об'єкт вивчення в Сонячній системі, є антициклонической бурею, яка триває не менше 300 років. Проте погода не обмежується планетарними тілами. Корона Сонця постійно губиться у просторі, створюючи, по суті, дуже тонку атмосферу у всій Сонячній системі. Рух частинок, що викидаються Сонцем, відомий як сонячний вітер.

Синоптична карта - це географічна карта, де умовними позначеннями показані спостереження багатьох метеостанцій. Такі карти дають чітке уявлення про погодні умови в певний час (рис. 2.3).

Шляхом накопичення таких карт можна визначити напрямок руху повітряних мас, розвиток циклонів, рух фронтів і т.д. Аналізуючи синоптичні карти, можна прогнозувати зміни погоди. Можна простежити міграцію та еволюцію атмосферних збурень, переміщення, деформацію та взаємодію повітряних мас та інші зміни стану атмосфери. Останні десятиліття синоптична інформація збагатилася з допомогою аеронавігаційних спостережень. В останні роки також використовується супутникова інформація про стан океанів та суші, де немає метеорологічних станцій. Супутникове знімання хмарних систем дозволяє виявити зародження тропічних циклонів над океанами [1].

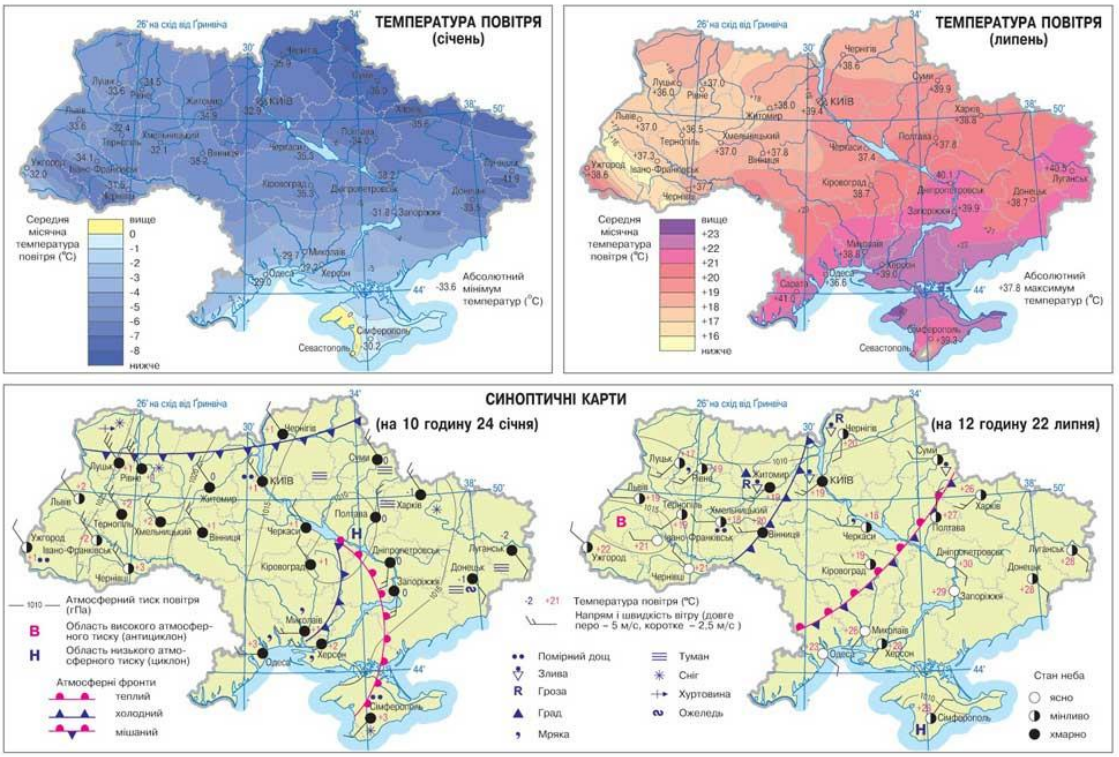


Рисунок 2.3 – Синоптичні карти

Web-додаток прогнозування погоди працює на основі збору та обробки погодних даних із різних джерел, таких як метеорологічні станції, супутники, радари та інші джерела.

Принцип роботи web-програми прогнозування погоди полягає у відображенні актуальних та прогнозних погодних даних на графічному інтерфейсі користувача. Для цього додаток використовує складні алгоритми прогнозування погоди, що базуються на зібраних погодних даних та статистичних методах аналізу.

Зазвичай web-програми прогнозування погоди мають інтерфейс, що складається з різних блоків, що відображають різні погодні параметри. Наприклад, це можуть бути блоки з інформацією про температуру, вітер, опади, рівень вологості та інші показники.

Додаток також може використовувати геопозицію користувача для відображення погодних умов у його регіоні. Користувач може вибрати іншу локацію, якщо він планує поїздку або хоче дізнатися погоду в іншому регіоні.

Інформація про погоду може бути представлена у формі графіків, діаграм, карти та інших візуальних елементів, що робить додаток зручним та зрозумілим для користувача.

Оновлення погодних даних зазвичай відбувається автоматично з певною періодичністю, що дозволяє користувачам отримувати актуальну інформацію про погоду в реальному часі [1].

2.3 Характеристика інструментів для розробки задачі

Створення веб-додатків дозволяє вирішувати різні завдання комерційних компаній. Так, наприклад, за допомогою веб-додатків можна вести облік роботи всіх співробітників, навіть якщо компанія має мережу віддалених філій, здійснювати облік вантажних чи пасажирських перевезень, здійснювати моніторинг діяльності компанії, керувати роботою персоналу, а також нараховувати заробітну плату.

Найкраще та найгірше у веб-розробці – це те, що все постійно змінюється. Хоча це цікаво, це також означає, що веб-розробники повинні завжди активно вивчати нові методи або мови програмування, адаптуватися до змін і бути готовими прийняти нові виклики. А це означає вирішувати такі завдання, як адаптація існуючих середовищ для відповідності бізнес-вимог, тестування веб-сайту для виявлення технічних проблем або оптимізація та масштабування сайту для кращої роботи з внутрішньою інфраструктурою. У своєму проекті було вирішено скласти повний список інструментів та ресурсів для веб-розробки, які допоможуть вам бути продуктивнішими. Основними з яких буде мови програмування Js/Css/Html та середовище розробки Visual Code [8].

2.3.1 Мова програмування JavaScript

JavaScript - це мова, яка спочатку була створена для "оживлення" веб-сторінок.

Програми цією мовою називаються скриптами. Сценарії можуть бути вбудовані в HTML і автоматично виконуються під час завантаження веб-сторінки (рис. 2.4).

Рисунок 2.4 – Мова програмування Javascript

Сценарії поширюються та виконуються як звичайний текст. Для їх запуску не потрібна спеціальна підготовка або компіляція. У цьому полягає різниця між JavaScript та іншою мовою Java.

Коли JavaScript був створений, він мав іншу назву: 'LiveScript'. Однак у той час була дуже популярна мова Java, і, ймовірно, було вирішено, що буде корисно позиціонувати JavaScript як молодшого брата Java.

З часом JavaScript став повністю незалежною мовою зі своєю власною специфікацією ECMAScript і тепер не має нічого спільного з Java.

Сьогодні JavaScript може виконуватися не тільки браузерами, а й серверами та іншими пристроями зі спеціальними програмами, які

називаються "движками" JavaScript. Браузери мають власні движки, іноді звані "віртуальними машинами JavaScript". Різні двигуни мають різні "кодові імена".

Наприклад:

- V8 - використовується у браузерах Chrome
- Opera та Edge.
- SpiderMonkey використовується у Firefox.
- Chakra в IE
- JavaScriptCore
- Nitro
- SquirrelFish в Safari.

Ці назви часто використовуються в статтях розробників, тому їх корисно знати. Також можемо їх використати. Наприклад, якщо "функції X підтримуються в V8", то "X", швидше за все, працюватиме в Chrome, Opera та Edge.

Двигуни складні. Однак, основи легко зрозуміти.

1. Двигун (вбудований у браузер) читає ("розбирає") текст сценарію.
2. Потім він перетворює скрипт на машинну мову ("компілює").
3. Потім код машинною мовою виконується і працює досить швидко.

Механізм виконує оптимізацію всіх етапах. Більш того, під час виконання скомпільованого сценарію він аналізує дані, що проходять через сценарій, та застосовує оптимізацію машинного коду на основі цих знань. В результаті скрипти виконуються дуже швидко.

Сучасний JavaScript - це "безпечна" мова програмування. Він не надає низькорівневого доступу до пам'яті або процесора, оскільки спочатку був розроблений для браузерів, які цього не потребували.

Можливості JavaScript сильно залежать від середовища, в якому він працює. Наприклад, Node.JS підтримує читання та запис довільних файлів, мережеві запити тощо.

Все, що пов'язано з маніпулюванням веб-сторінками та взаємодією з користувачами та веб-серверами, може бути зроблено за допомогою JavaScript у браузері.

Наприклад, у браузері JavaScript може:

- Додавати новий HTML-код на сторінку, змінювати існуючий контент, змінювати стилі.
- Реагувати на дії користувача (клацання миші, переміщення вказівника, натискання клавіш).
- Надсилати мережеві запити на віддалені сервери, завантажувати та завантажувати файли (технології AJAX та COMET).
- Отримувати та встановлювати cookies, ставити запитання відвідувачам та виводити повідомлення.
- Зберігання даних за клієнта (" локальне сховище ").
- Не менше трьох спеціалізацій з JavaScript:
- Повна інтеграція з HTML/CSS.
- Простота, доведена до досконалості.
- Підтримується всіма основними браузерами та включений за замовчуванням.

JavaScript - єдина браузерна технологія, яка поєднує всі три спеціалізації. Саме це робить JavaScript особливим. Саме тому він є найпоширенішим інструментом для створення інтерфейсів у браузерах. Хоча, звичайно, JavaScript можна використовувати для створення програм не тільки в браузері, а й на серверах, мобільних пристроях і т.д.

Синтаксис JavaScript не підходить для всіх потреб. Різним людям потрібна різна функціональність.

Це не дивно, оскільки проекти бувають різні та вимоги до них теж. Наприклад, існує низка нових мов, які транслюються (перетворюються) на JavaScript перед запуском у браузері.

Сучасні інструменти роблять транспілінг дуже швидким і прозорим, тому розробники можуть писати код іншими мовами і автоматично перетворювати його на JavaScript "під капотом".

Приклади таких мов:

- CoffeeScript додає "синтаксичний цукор" JavaScript, що дозволяє писати чистий, лаконічний код за рахунок більш короткого синтаксису. Зазвичай віддають перевагу програмістам на Ruby.
- TypeScript фокусується на додаванні "суворої типізації" для полегшення розробки та супроводу великих, складних систем. Він був розроблений компанією Microsoft.
- Flow також додає типізацію, але в інший спосіб; розроблений компанією Facebook.
- Dart виділяється тим, що має власний двигун, який працює поза браузером (наприклад, у мобільних додатках). Спочатку він був запропонований Google як альтернатива JavaScript, але тепер для роботи на ньому потрібне транспонування, як і вищепереліченими мовами.
- Brython переводить Python у JavaScript, дозволяючи писати програми на чистому Python без JavaScript [1].

2.3.2 Мова для структури web – сайту HTML

HTML (HyperText Markup Language) – це найголовніший будівельний блок Інтернету. Він визначає зміст та структуру веб-контенту; технології, відмінні від HTML, зазвичай використовуються для опису зовнішнього

вигляду/виразу (CSS) або функціональності/дії (JavaScript) веб-сторінки (рис. 2.5).

Гіпертекст стосується посилань між веб-сторінками, як усередині веб-сайтів, так і між ними. Посилання є основним елементом Інтернету.

Рисунок 2.5 – Мова HyperText Markup Language

Завантажуючи контент в Інтернет та розміщуючи посилання на сторінки, створені іншими, кожен є активним учасником всесвітньої павутини.

Мова гіпертекстової розмітки (HTML) була розроблена британським ученим Тімом Бернерсом-Лі в ЦЕРНі в Женеві, Швейцарія, приблизно в 1986-1991 роках. HTML була задумана як мова для обміну науково-технічними документами та для фахівців з верстки. Створений для використання нефахівцями, HTML успішно вирішує проблеми складності SGML, визначаючи невеликий набір дескрипторів, які є структурними та семантичними елементами. Дескриптори, які часто називають "тегами", дозволяють легко створювати відносно прості, але красиво відформатовані документи за допомогою HTML. Крім спрощення структури документів,

HTML була введена підтримка гіпертексту. Мультимедійні функції було додано пізніше.

Першим загальнодоступним описом HTML був документ "Теги HTML", вперше згаданий в Інтернеті Тімом Бернерсом-Лі наприкінці 1991. У ньому описано 18 елементів, що становлять початковий, відносно простий дизайн HTML. За винятком тегів гіперпосилань, на нього сильно вплинув SGMLguid, внутрішній формат документів, що базується на стандартній узагальненій мові розмітки CERN (SGML). Одинадцять із цих елементів, як і раніше, присутні в HTML.

HTML спочатку замислювався та розроблявся як засіб структурування та форматування документів без прив'язки до засобів відтворення (презентації). В ідеалі текст із розміткою HTML має бути здатним відтворюватися без спотворення стилю та структури на пристроях з різними технічними можливостями (кольорові екрани сучасних комп'ютерів, монохромні екрани ноутбуків, обмежені екрани мобільних телефонів та пристроїв, програми для аудіовідтворення тексту). Проте сучасне використання HTML далеко від його початкового призначення. Наприклад, тег<table>використовується для створення таблиць у документі, але іноді його застосовують для оформлення розташування елементів на сторінці. Згодом основна ідея про те, що HTML не залежить від платформи, була принесена в жертву сучасним потребам мультимедіа та графічного дизайну [1].

2.3.3 Мова для реалізації зовнішнього вигляду CSS

CSS (Cascading Style Sheets) - це код, який використовується для стилізації веб-сторінок; розуміння основ CSS дозволяє розпочати роботу (рис. 2.6).

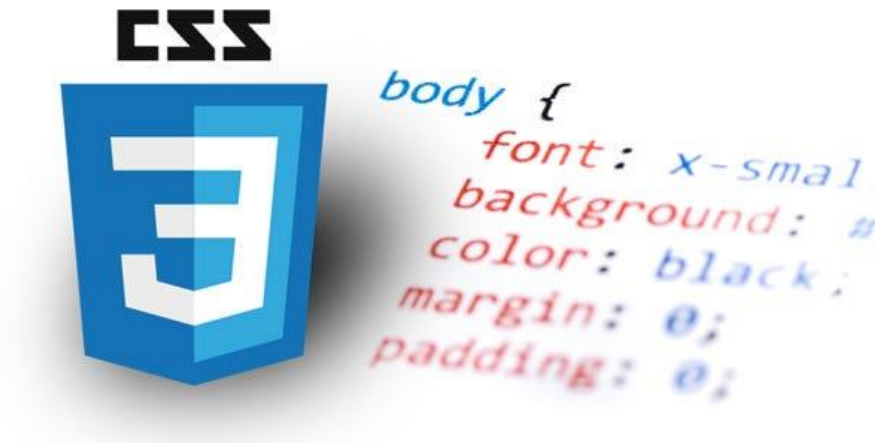


Рисунок 2.6 – Код Cascading Style Sheets

Як і HTML, CSS практично не є мовою програмування. Він не є мовою розмітки, не мовою таблиць стилів. Це означає, що можна вибірково застосовувати стилі до елементів HTML-документа.

CSS використовується авторами веб-сторінок для налаштування зовнішнього вигляду веб-сторінки, включаючи кольори, шрифти, стилі, макети блоків тощо. Мета розробки CSS - відокремити опис логічної структури веб-сторінки мовою розмітки, такому як HTML, від опису зовнішнього вигляду веб-сторінки CSS. Мета CSS - розділити та відокремити опис логічної структури веб-сторінки мовою розмітки, такою як HTML, від опису зовнішнього вигляду веб-сторінки в CSS. Такий поділ покращує доступність, дозволяє гнучкіше контролювати спосіб подання документів і зменшує складність і повторення структурного вмісту [6].

Крім того, CSS дозволяє відображати один і той же документ у різних стилях та режимах виведення, таких як екранний дисплей, друкований дисплей, голосове читання (за допомогою спеціального голосового браузера або програми читання з екрана) та виведення через пристрої Брайля (рис. 2.7).

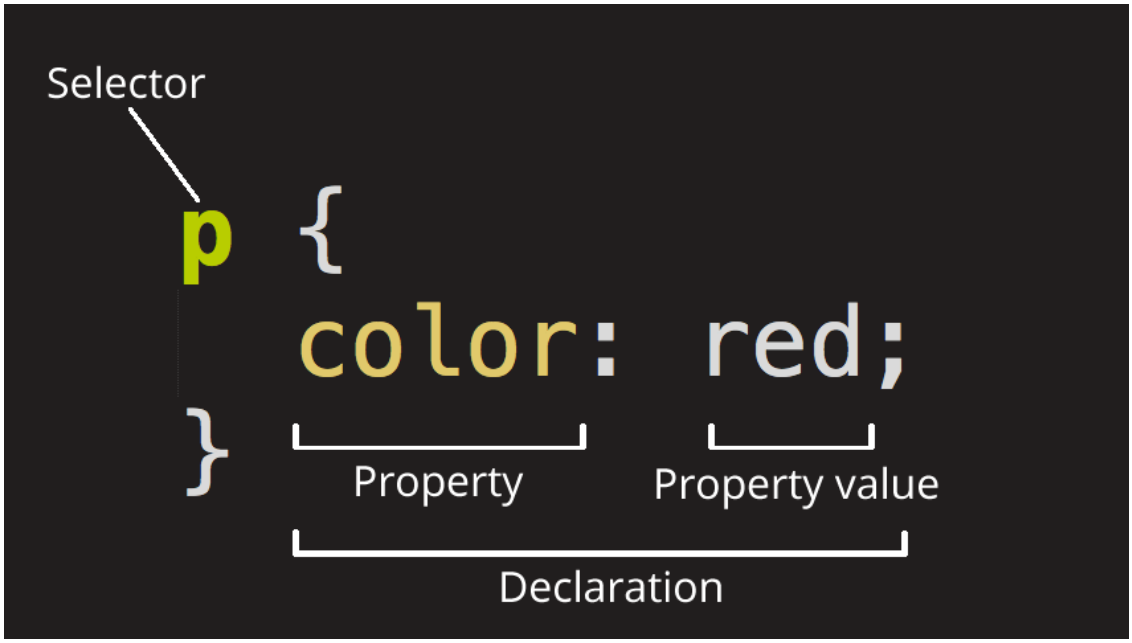


Рисунок 2.7 – Вигляд селекторів

Правила CSS можуть бути розміщені або у веб-документі, який описує зовнішній вигляд, або у зовнішньому файлі з розширенням .css. CSS - це текстовий файл, що містить список правил CSS та коментарі до них.

До появи CSS веб-сторінки розроблялися з використанням лише HTML безпосередньо у вмісті документа. Однак з появою CSS стало можливим радикально розділити зміст та подання документа. Це нововведення дозволило легко застосовувати єдиний стиль оформлення до великої кількості однакових документів та швидко змінювати їх дизайн [2].

Переваги.

- Декілька варіантів оформлення сторінок для різних пристроїв перегляду. Наприклад, оформлення сторінки з великою шириною на екрані, без меню під час друку, так, щоб меню йшло за вмістом на КПК та мобільних телефонах.
- Скорочення часу завантаження сторінки за рахунок перенесення правил подання до окремого файлу CSS. У цьому випадку браузер зчитує лише структуру документа та дані, що зберігаються на

сторінці; подання цих даних завантажується браузером лише один раз і може бути кешований.

- Легкість наступних змін дизайну. Не потрібно редагувати всі сторінки, потрібно змінити лише файл CSS.
- Більш широкі можливості для дизайну. Наприклад, CSS-макети можна використовувати для створення текстових блоків, що пропускають інший текст (наприклад, меню), або для того, щоб меню завжди було видно під час прокручування сторінки.

Недоліки.

- Одні й ті ж дані CSS відображатимуть різні макети у браузерах з різною інтерпретацією (особливо у застарілих браузерах).
- Часта необхідність вносити складні та непривабливі зміни до HTML-тегів, пов'язаних із селекторами CSS, а не тільки в фактичний окремий файл CSS, що може зробити єдиний файл стилів менш придатним для використання і значно збільшує час редагування та тестування. [4]

2.3.4 Середовище програмування VisualCode

Visual Studio Code (VS Code) – це текстовий редактор, розроблений компанією Microsoft для Windows, Linux та MacOS. Він позиціонується як "легкий" редактор коду для кросплатформної розробки веб- та хмарних програм. Він включає відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та інструменти рефакторингу. Він пропонує широкі можливості налаштування, включаючи теми користувача, поєднання клавіш і файли конфігурації. Вона розробляється як програмне забезпечення з відкритим вихідним кодом і розповсюджується безкоштовно, але готові зборки розповсюджуються під власною ліцензією (рис. 2.8).

Visual Studio Code заснований на Electron та реалізований через веб-редактор Monaco, розроблений для Visual Studio Online.

Visual Studio Code був анонсований компанією Microsoft на конференції Build 29 квітня 2015 і незабаром була випущена бета-версія.

18 листопада 2015 Visual Studio Code була випущена під ліцензією MIT, а вихідний код був опублікований на GitHub. Було оголошено підтримку розширень.

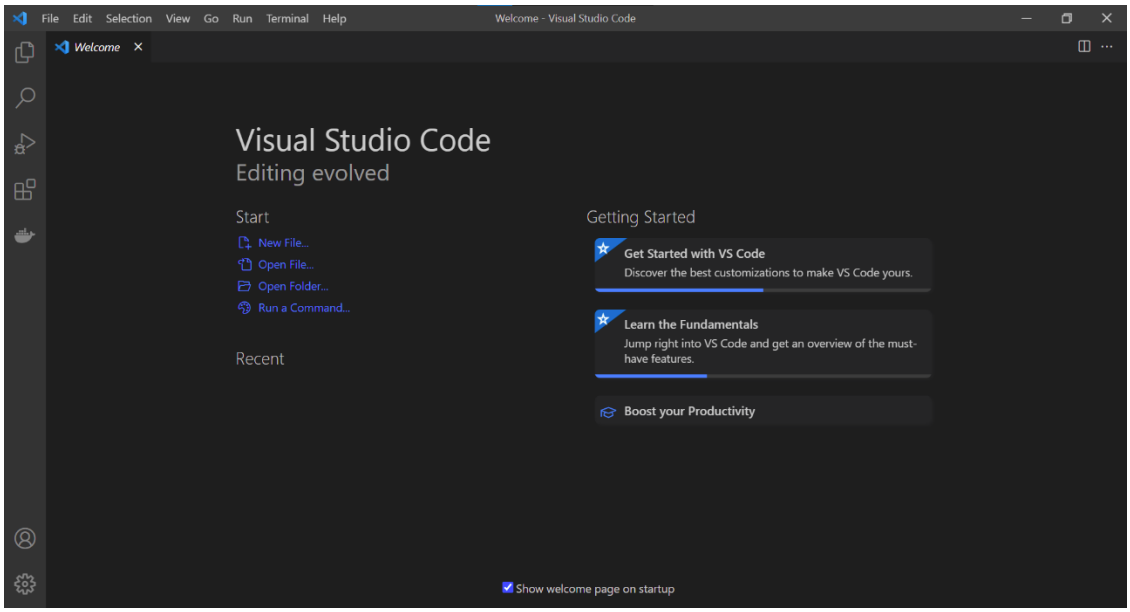


Рисунок 2.8 – Основна сторінка середовища розробки

14 квітня 2016 року Visual Studio Code закрила бета-тестування.

Visual Studio Code – це редактор вихідного коду. Він має багатомовний інтерфейс користувача і підтримує кілька мов програмування, підсвічування синтаксису, IntelliSense, рефакторинг, налагодження, навігацію за кодом, підтримку Git і т.д. Багато можливостей Visual Studio Code недоступні через графічний інтерфейс, оскільки вони часто використовуються через панелі команд або JSON файли (наприклад, налаштування користувача). Палітра команд схожа на командний рядок, який може бути викликаний за допомогою клавіш.

VS Code також дозволяє користувачеві змінити кодову сторінку, символ перекладу рядка та мову програмування поточного документа під час збереження документа.

На 2018 рік Visual Studio Code має розширення Python з відкритим вихідним кодом. Це надає розробникам широкі можливості для редагування, налагодження та тестування коду.

VS Code також підтримує редагування та запуск файлів типу Jupyter Notebook безпосередньо "з коробки" у візуальному режимі та режимі редагування вихідного коду без необхідності встановлення зовнішніх модулів.

Станом на березень 2019 року тисячі розширень тільки в категорії "мови програмування" можуть бути завантажені та встановлені з вбудованого інтерфейсу користувача.

Розширення також полегшують доступ до таких програм, як Docker та Git. У розширеннях можна знайти засоби написання коду, теми редактора та підтримку синтаксису для окремих мов [1].

2.3.5 Середовище проектування Proteus

Proteus Design Suite – пакет програм для автоматизованого проектування (САПР) електронних схем. Розробка компанії Labcenter Electronics (Велика Британія).

Пакет є системою моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice. Відмінною рисою пакету PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP та ін. Причому в Proteus повністю реалізована концепція наскрізного проектування, коли інженер змінює щось у логіці роботи схемотехніки і програмний пакет тут же «підхоплює» дані зміни в системі трасування. Бібліотека компонентів містить довідкові дані. Додатково до пакету PROTEUS VSM входить система

проектування друкованих плат. Пакет Proteus складається з двох частин, двох підпрограм: ISIS – програма синтезу та моделювання безпосередньо електронних схем та ARES – програма розробки друкованих плат. Разом із програмою встановлюється набір демонстраційних проектів для ознайомлення (рис. 2.9).

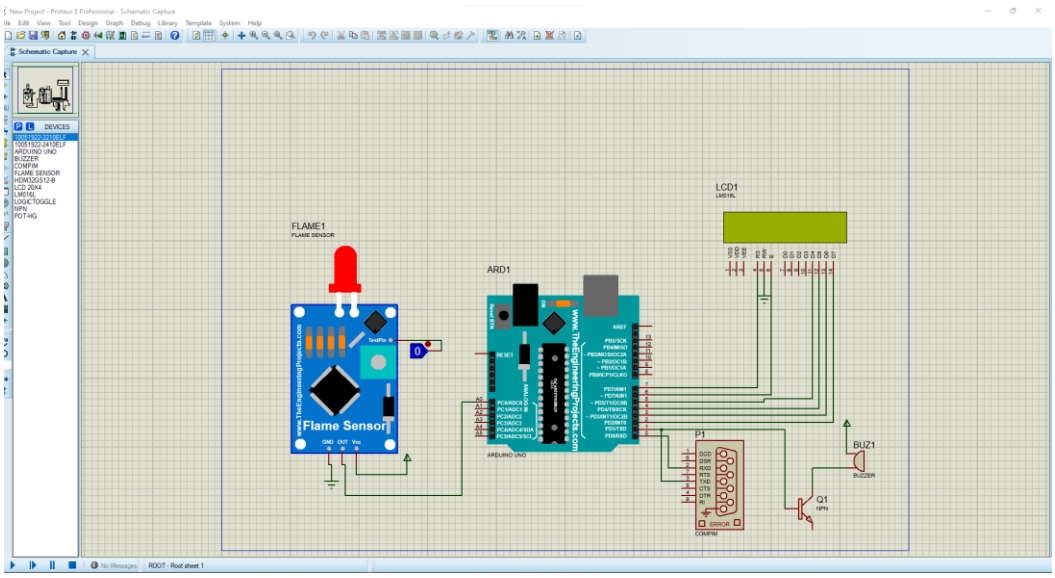


Рисунок 2.9 – Головне меню програми

Також до складу восьмої версії входить середовище розробки VSM Studio, що дозволяє швидко написати програму для мікроконтролера, що використовується в проекті, та скомпілювати [1].

2.3.6 Мова для програмування схеми на Arduino

Arduino — торгова марка апаратно-програмних засобів побудови та прототипування простих систем, моделей та експериментів у галузі електроніки, автоматики, автоматизації процесів та робототехніки.

Програмна частина складається з безкоштовної програмної оболонки (IDE) для написання програм, їх компіляції та програмування апаратури. Апаратна частина є набором змонтованих друкованих плат, що продаються як офіційним виробником, так і сторонніми виробниками. Цілком відкрита

архітектура системи дозволяє вільно копіювати або доповнювати лінійку продукції Arduino.

Використовується як для створення автономних об'єктів, так і підключення до програмного забезпечення через дротові та бездротові інтерфейси. Підходить для початківців з мінімальним вхідним порогом знань у галузі розробки електроніки та програмування [1].

2.4 Архітектура web–застосунку

Архітектура веб-програми визначає розташування всіх компонентів веб-програми і показує взаємодію між різними компонентами програми, сторонніми системами проміжного ПЗ, веб-сервісами та базами даних. Вона дає уявлення про взаємодію між кількома програмами, які працюють разом для надання послуг кінцевим користувачам (рис. 2.10).

Архітектура програмного забезпечення включає всі високорівневі компоненти системи і взаємодії всередині них.

Проектування програмного забезпечення - це проектування лише на рівні коду, метою якого є розподіл бізнес-логіки програми з різних модулів, кожен із яких має певну мету. Воно допомагає побудувати та керувати бізнес-логікою програми.

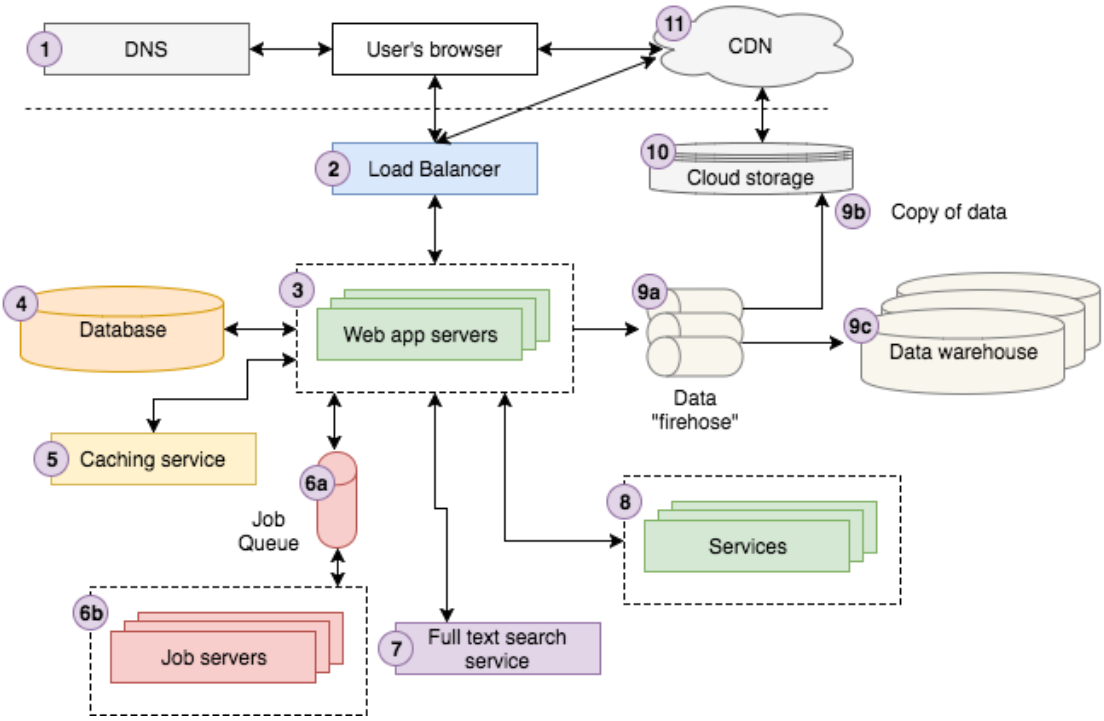


Рисунок 2.10 – Блок-схема повноцінної веб-сторінки

Існує лише три моделі компонентів веб-застосунків. Вони тісно пов'язані з кількістю сервісів та баз даних, які використовуються у веб-додатку. До них відносяться:

- Один веб-сервер; одна база даних;
- Декілька веб-серверів, одна база даних;;
- Декілька веб-серверів, кілька баз даних;
- Сервіси програми;

Три так звані монолітні моделі є такими, тому що сервери жорсткі та стабільні. Навпаки, послуги додатків (мікросервіси, безсерверні), як правило, більш гнучкі, тому що вони спрощують оновлення та масштабування. Застосовуючи цю модель, веб-сервер можна розділити на дрібніші частини, такі як сервіси в мікросервісах та функції в serverless. Це полегшує зміну чи розширення будь-якої з цих частин незалежно одна від одної.

Основні критерії для побудови надійної архітектури додатків:

- Ефективність;
- гнучкість;
- Можливість повторного використання;
- Простота тестування;
- Послідовне та успішне вирішення проблем
- Структурований код, що легко розуміється; і
- Масштабованість у процесі розробки;
- Швидкий час відгуку;
- Відсутність збоїв; Відсутність збоїв; Відсутність збоїв; Відсутність збоїв;
- Відсутність єдиної точки відмови; і
- Простота;
- Опора на перевірені стандарти безпеки.

Архітектура веб-програми для прогнозування погоди може бути заснована на архітектурі клієнт-сервер. На стороні сервера знаходяться програмні модулі, які отримують дані про погоду з різних джерел, обробляють їх та генерують прогнози. Дані можуть збиратися з різних глибинних датчиків та інших веб-ресурсів, які надають інформацію про погоду.

На стороні клієнта користувачі можуть взаємодіяти з веб-програмою за допомогою веб-браузера. Клієнтська частина може включати інтерфейс користувача, який дозволяє користувачам вводити параметри прогнозу погоди, переглядати результати прогнозу і зберігати налаштування.

Крім того, веб-додаток може використовувати базу даних для зберігання інформації про прогноз погоди та інших даних, що використовуються для обробки даних. Для забезпечення безпеки та захисту від несанкціонованого доступу можуть використовуватись різні методи, такі як автентифікація та авторизація користувачів [5].

Така архітектура дозволяє створювати веб-програми для прогнозування погоди, доступ до яких можливий з будь-якого пристрою, підключеного до Інтернету. Використання різних інструментів та технологій також може забезпечити високу продуктивність та швидкість обробки даних, що важливо для надання користувачам швидких та точних прогнозів погоди.

Для веб-програми прогнозування погоди архітектура може містити кілька ключових компонентів

- Front-end - клієнтська частина програми, що відповідає за взаємодію з користувачем та відображення інформації на екрані.
- Back-end - серверна частина програми, що відповідає за обробку запитів користувача, зберігання даних та підготовку відповідей.
- API - інтерфейс програмування, що забезпечує взаємодію між зовнішніми та внутрішніми компонентами.
- База даних - система для зберігання та організації даних про погоду із зовнішніх джерел.
- Системи прогнозування - алгоритми та моделі для обробки погодних даних та прогнозування майбутніх погодних умов.
- Зовнішні джерела даних - послуги або API, що надають доступ до фактичних погодних даних, які використовуються для розрахунку прогнозів.

Архітектура веб-програми зазвичай забезпечує поділ обов'язків між різними компонентами для забезпечення ефективності та надійності системи. Вона також гарантує, що система може бути масштабована чи розширена у майбутньому.

2.5 Обґрунтування системних та програмних вимог

Щоб розробити веб-програму для прогнозування погоди, необхідно визначити системні та програмні вимоги. Системні вимоги відносяться до апаратного забезпечення, на якому працюватиме програма, в той час як програмні вимоги відносяться до необхідного програмного забезпечення та технології.

Серверна частина веб-програми повинна працювати на потужному сервері з достатньою оперативною пам'яттю та потужністю процесора. Також потрібне постійне підключення до Інтернету, щоб можна було отримувати актуальні дані про погоду.

Для створення програмної частини необхідно знати мови програмування, такі як JavaScript, HTML і CSS, а також різні фреймворки та бібліотеки для розробки веб-додатків. Також необхідні базові знання з прогнозування погоди та аналізу метеорологічних даних.

Для забезпечення безпеки даних користувача повинні бути реалізовані такі заходи безпеки, як шифрування і перевірка даних, що вводяться.

Усі вимоги до системи та програмного забезпечення повинні бути детально обґрунтовані з урахуванням потреб користувачів та бізнесу.

Крім системних та програмних вимог, необхідно визначити функціональні та нефункціональні вимоги до веб-додатку. Функціональні вимоги визначають, які можливості та функції мають бути застосовані, наприклад, можливість відображення поточної погоди та прогнозів на різні дні, або графіків, що показують зміни погоди.

Нефункціональні вимоги визначають, якими мають бути функціональність та якість веб-програми, наприклад, швидкість, стабільність, безпека, доступність для користувачів з різними платформами та пристроями, зручний інтерфейс. Необхідно також враховувати, що прогнозування погоди – складний процес, і точність прогнозу залежить від

багатьох факторів, включаючи наявність свіжих та точних погодних даних, якість алгоритмів аналізу та досвід фахівців. Тому досить складно розробити веб-додаток, який забезпечує 100% точний та надійний прогноз погоди. Тому при розробці веб-програми для прогнозування погоди необхідно враховувати системні, програмні, функціональні та нефункціональні вимоги, а також складність самого процесу прогнозування погоди. Всі ці фактори необхідно враховувати при розробці та тестуванні, щоб забезпечити якість та ефективність програми.

Висновок до розділу 2:

З результатів проведеного аналізу різних інструментів для розробки веб-застосунків для прогнозування погоди можна зробити висновок, що вибір відповідного інструменту залежить від конкретних потреб користувачів та розробників. При виборі відповідного інструмента слід враховувати такі фактори, як масштабованість системи, продуктивність, масштабованість та підтримка спільноти розробників. Для розробки більшості погодних веб-застосунків найкращим вибором можуть бути відкриті та безкоштовні інструменти, такі як HTML, CSS, JavaScript та бібліотеки, такі як jQuery, React та Bootstrap. Спеціалізовані фреймворки та платформи, такі як Django, Flask, Node.js та AWS, також можуть бути використані залежно від вимог до функціональності та продуктивності.

Таким чином, вибір інструментів для створення веб-програмної програми для прогнозування погоди повинен здійснюватися відповідно до конкретних потреб і вимог системи і може бути зроблений шляхом ретельного аналізу та порівняння різних інструментів**.**

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB–ЗАСТОСУНКУ ДЛЯ ПРОГНОЗУВАННЯ ПОГОДИ

3.1 Підготовка робочого місця до роботи і завантаження програмного забеспечення

Для початку роботи треба завантажити з офіційного сайту середовище програмування Visual code, його описували у другому розділі (рис. 3.1).

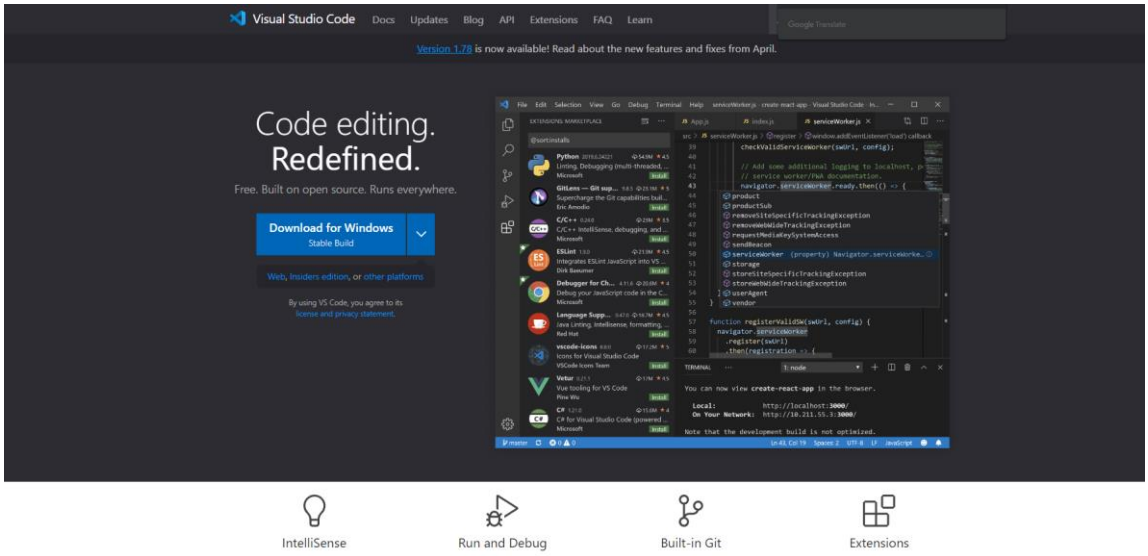


Рисунок 3.1 – Офіційний сайт завантаження

В результаті у нас з’явиться програма де зможемо компілювати свій код для виводу результатів. Головну сторінку було зображено у другому розділі.

Створюємо папку із власним проектом де будуть знаходитися всі наші файли які будемо використовувати, папка с моїм проектом буде названа “WetherApp” (рис. 3.2, 3.3).

Обираємо шрифти які потрібні для нашого проекту, завантаження їх здійснюється з браузера, мій проект використовує “symbol-routed”. Зображення для мого застосунку було знайдено в інтернеті.

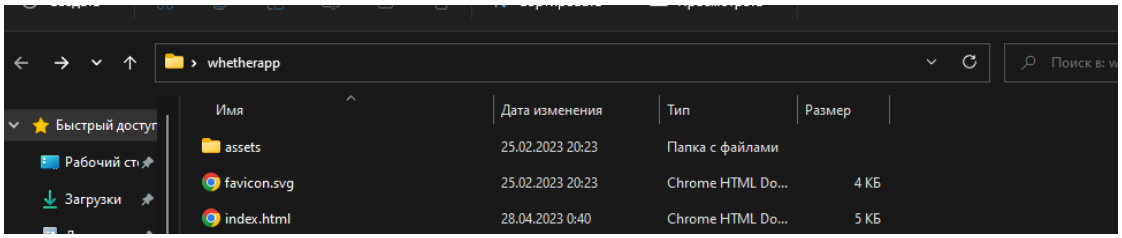


Рисунок 3.2 – Директорія проекту

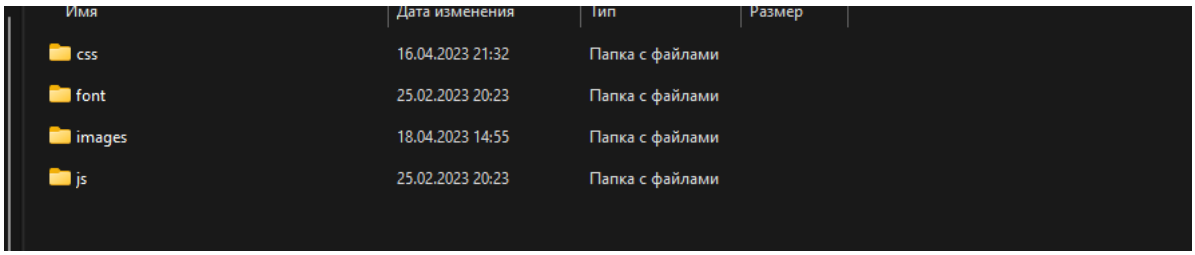


Рисунок 3.3 – Друга частина папки проекта

Для свого проекту було використано такі файли:

Index.html – основна сторінка web – застосунка.

- Favicon.html – сторінка де було взяти іконка для зображення вкладки сайту.

  Stylep.css – файл для формування зовнішнього вигляду та стилей
- Font – папка с шрифтами для нашого застосунка

  images – зображення для коректнішої орієнтації
- App/Route/Api/Module…js – чотири файла javasctript інформація про які буде пізніше.

3.2 Створення основної структури та інтерфейсу web–застосунку

3.2.1 Структурне створення фундамента web – застосунка через HyperText Markup Language

Почнемо с файлу index.html, спочатку було написано шаблонні дані, які потрібні браузеру для коректної роботи (рис. 3.4).

Рисунок 3.4 – Зовнішній вигляд шаблонної частини коду

Основна наша інформація та код буде знаходитися в селекторі “head” та “body”. Селектор “head”, що перекладається як голова містить в собі всю організаційну інформацію в ньому з’єднується всі організаційні моменті

застосунком. (рис. 3.5).

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <!--
    meta tags
  -->
  <title>weather</title>
  <meta name="title" content="weather">
  <meta name="description" content="weather is a weather app made by Rich Rayse Bki-19">

  <!--
    favicon
  -->
  <link rel="shortcut icon" href="./favicon.svg" type="image/svg+xml">

  <!--
    google font
  -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Nunito+Sans:wght@400;600&display=swap" rel="stylesheet">

  <!--
    custom css link
  -->
  <link rel="stylesheet" href="./assets/css/stylep.css">

  <!--
    custom js link
  -->
  <script src="./assets/js/app.js" type="module"></script>
  <script src="./assets/js/route.js" type="module"></script>
</head>
```

Рисунок 3.5 – Вміст тега “head”

Вище наведений приклад мого проекту, де було додано:

-Title – назва вкладки застосунка. <meta name="title" content="weather"> - вказує заголовок сторінки, який відображатиметься у вкладці браузера та в результатах пошуку, <meta name="description" - визначає короткий опис сторінки, який буде відображатися в результатах пошуку, під заголовком. В даному випадку, заголовок сторінки встановлений як "weather", а опис - "weather це додаток для прогнозу погоди, створений Rich Rayse".

- Link favicon – з’єднуємо фавікон.

-Google fonts – гугл шрифти.

- Link css/Script – файл css та js.

Для більш впевненої орієнтації по коду було написано коментарі які виділені зеленим кольором, так можна побачити і зрозуміти який блок коду до чого відноситься, це допоможе не тільки мені не заплутатися, но і замовнику у майбутньому.

Тэг “body” в ньому буде весь код про сайт, в усіх випадках web – застосунки мають три контейнера основних це “header, main, footer” шапка, основа та підвал.

Шапка в моєму проекті виглядає наступним чином (рис. 3.6).

```
<!--
  #HEADER
-->

<header class="header">
  <div class="container">

    <a href="https://i.pinimg.com/564x/e7/5e/d5/e75ed5470490054f0df918d2fceb7079.jpg" class="logo">
      <img src="./assets/images/8" width="364" height="58" alt="logo">
    </a>

    <div class="search-view" active data-search-view>

      <div class="search-wrapper">
        <input type="search" name="search" placeholder="Search city..." autocomplete="off" class="search-field"
          data-search-field>

        <span class="m-icon leading-icon">search</span>

        <button class="icon-btn leading-icon has-state" aria-label="close search" data-search-toggler>
          <span class="m-icon">arrow_back</span>
        </button>
      </div>

      <div class="search-result" data-search-result></div>

    </div>

    <div class="header-actions">

      <button class="icon-btn has-state" aria-label="open search" data-search-toggler>
        <span class="m-icon icon">search</span>
      </button>

      <a href="#/current-location" class="btn-primary has-state" data-current-location-btn>
        <span class="m-icon">my_location</span>

        <span class="span">Current Location</span>
      </a>

    </div>

  </div>
</header>
```

Рисунок 3.6 – Вміст тега “header”

: контейнер для розміщення вмісту в центрі сторінки з фіксованою шириною

<a href="https://i.pinimg.com/564x/e7/5e/d5/e75ed5470490054f0df918d2fceb7079.jpg" class="logo">: посилання-логотип із зображенням

<div class="search-view" active data-search-view>: область пошуку, яка має атрибути active та data-search-view

: обгортка для поля пошуку та кнопки

<input type="search" name="search" placeholder="Search city..." autocomplete="off" class="search-field" data-search-field>: поле для введення тексту пошуку з атрибутами autocomplete та data -search-field

search: іконка лупи зліва від поля пошуку

<button class="icon-btn leading-icon has-state" aria-label="close search" data-search-toggler>: кнопка для закриття області пошуку

arrow_back: іконка стрілки "назад" у кнопці закриття пошуку

<div class="search-result" data-search-result> : контейнер для відображення результатів пошуку

: область дій у заголовку

<button class="icon-btn has-state" aria-label="open search" data-search-toggler>: кнопка для відкриття області пошуку

<a href="#/current-location" class="btn-primary has-state" data-current-location-btn>: посилання на поточне розташування з атрибутами class та data-current-location-btn

my_location: іконка геолокації зліва від посилання на поточне розташування

Current Location: текст "Current Location" поруч із посиланням на поточне розташування.

Результатом було отримано шапку, яку потім будемо стилізувати (рис. 3.7).

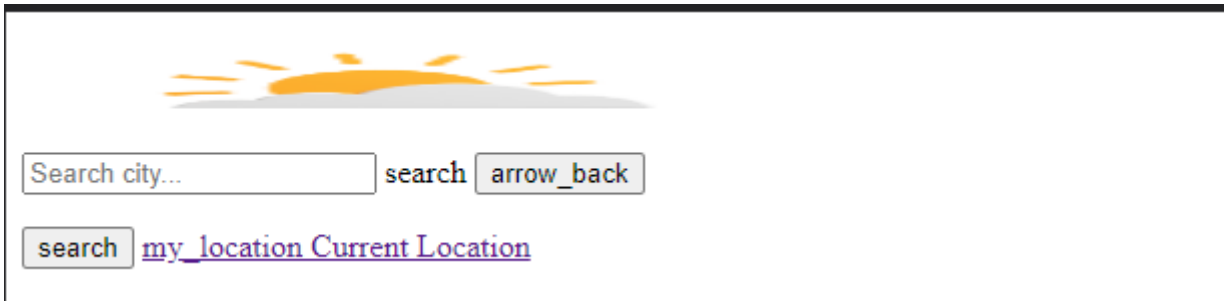


Рисунок 3.7 – Результат тега “header”

Переходячи до тега “body” де знаходиться основний вміст застосунка тобто головний контейнер та наш підвал “footer” (рис 3.9).

```
<main>
  <article class="container" data-container>

    <div class="content-left">
      <!--
        - #CURRENT WEATHER
      -->

      <section class="section current-weather" aria-label="current weather" data-current-weather></section>

      <!--
        - #FORECAST
      -->

      <section class="section forecast" aria-labelledby="forecast-label" data-5-day-forecast></section>
    </div>

    <div class="content-right">
      <!--
        - #HIGHLIGHTS
      -->

      <section class="section highlights" aria-labelledby="highlights-label" data-highlights></section>

      <!--
        - #HOURLY FORECAST
      -->

      <section class="section hourly-forecast" aria-label="hourly forecast" data-hourly-forecast></section>
```

Рисунок 3.8 – Вміст тега “main”

```
    <footer class="footer">
      <p class="body-3">
        Copyright 2023 By Rich Rayse.
      </p>

      <p class="body-3">
        Powered By <a href="https://openweathermap.org/api" title="Free OpenWeather Api" target="_blank"
          rel="noopener">
          <img src="./assets/images/openweather.png" width="150" height="30" loading="lazy" alt="OpenWeather">
        </a>
      </p>
    </footer>
  </div>

  <div class="loading" data-loading></div>



</article>
</main>
```

Рисунок 3.9 – Тег “footer”

Елемент <main> є головним контейнером сторінки, який містить два дочірні елементи: та (рис 3.8).

 містить дві секції:

Перша секція представляє поточну погоду і містить атрибути aria-label і data-current-weather.

Інша секція <section class="section forecast" aria-labelledby="forecast-label" data-5-day-forecast></section> представляє прогноз погоди на 5 днів і містить атрибути aria-labelledby і data-5-day- forecast.

 містить три секції:

- Перша секція <section class="section highlights" aria-labelledby="highlights-label" data-highlights></section> містить підзаголовок "highlights" і дані, пов'язані зі значними аспектами погоди, такими як температура, вітер , та містить атрибут data-highlights.
- Інша секція - прогноз погоди на годину вперед і містить атрибут aria-label і data-hourly-forecast.

- Третя секція <footer class="footer"> містить копірайт та логотип OpenWeather із посиланням на веб-сайт OpenWeather та містить елемент <p> з класом "body-3".

Також код містить додатковий елемент <div class="loading" data-loading> , який, використовується для відображення індикатора завантаження або прогрес-бару.

Останнім йде блок коду, який показує нам помилку 404, якщо здійснилась невдала переадресація (рис 3.10).

```
<!--
  - #404 SECTION
-->

<section class="error-content" data-error-content>

  <h2 class="heading">404</h2>

  <p class="body-1">Page not found!</p>

  <a href="#/weather?lat=51.5073219&lon=-0.1276474" class="btn-primary">
    <span class="span">Go Home</span>
  </a>

</section>

/body>

/html>
```

Рисунок 3.10 – 404 Section

Деякий html код буде використано в js файлі для зміни данних при запиті другої країни (рис 3.11).

Рисунок 3.11 – Зовнішний вигляд сайту без стилей

3.2.2 Розробка стилей для html файлу за домогою Cascading Style Sheets

Css – домопоможе нам стилізувати наш файл html, розробити для нього сприятливий для ока та гарний зовнішній вигляд, а також адаптувати під мобільні пристрої або планшети. За допомогою css також було розроблено

анімацію кола, яке обертається в циклі, якщо щось завантажується або помилка.

Css файл мого проекту вийшов на 1000+ строків, тому розповідатися буде про головне. Для початку було написано деякі константи для більш зручнішного користування (рис 3.12).

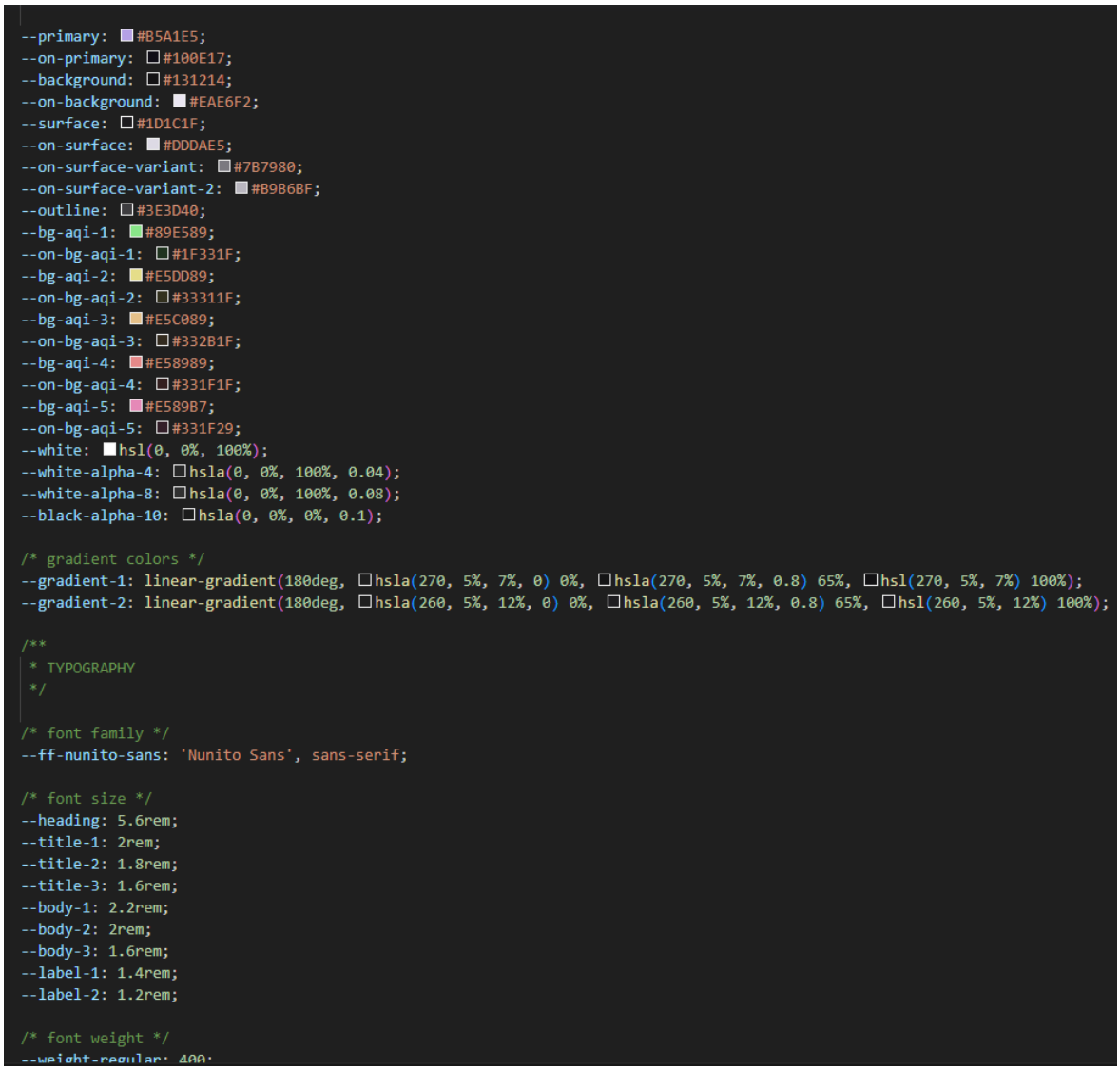


Рисунок 3.12 – Потрібні для проекту константи

За допомогою них можливо не вводячи код кольору або іншого просто вказати назву константи.

Основною опцією йде обнуління, яке допоможе скинути всі отступи та стилі які йдуть по замовчуванню в барузерах (рис 3.13).

```css
*,
*::before,
*::after {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

li { list-style: none; }

a,
img,
span,
input,
button { display: block; }

a {
  color: inherit;
  text-decoration: none;
}

img { height: auto; }

input,
button {
  background: none;
  border: none;
  color: inherit;
  font: inherit;
}

input { width: 100%; }

button { cursor: pointer; }

sub { vertical-align: baseline; }

sup { vertical-align: top; }

sub, sup { font-size: 0.75em; }

html {
  font-family: var(--ff-nunito-sans);
  font-size: 10px;
  scroll-behavior: smooth;
}
```

Рисунок 3.13 – Код обнуління

* - означає всі елементи на сторінці.

*::before і *::after - позначають псевдоелементи, додані перед та після кожного елемента відповідно.

margin: 0; та padding: 0; - встановлюють відступи та внутрішні відступи всіх елементів на сторінці в нуль.

box-sizing: border-box; - встановлює тип розміру для всіх елементів на сторінці в border-box, який включає межі і відступи в загальну ширину і висоту елемента.

li { list-style: none; } – видаляє маркери списків у всіх елементів списку.

a, img, span, input, button { display: block; } - встановлює властивість display на block для всіх посилань, зображень, текстових блоків, полів уведення та кнопок.

a { color: inherit; text-decoration: none; } - встановлює колір посилань на успадкований колір та видаляє підкреслення посилань.

img {height: auto; } - встановлює автоматичну висоту зображення залежно від ширини.

input, button { background: none; border: none; color: inherit; font: inherit; } - видаляє фон та межі біля полів введення та кнопок, встановлює колір та шрифт на успадковані значення.

input {width: 100%; } – встановлює ширину поля введення на 100%.

button { cursor: pointer; } - встановлює курсор при наведенні на кнопку на pointer (покажчик).

sub { vertical-align: baseline; } та sup { vertical-align: top; } - встановлюють вертикальне вирівнювання надрядкових та підрядкових символів на базову лінію та верхню межу відповідно.

sub, sup { font-size: 0.75em; } - зменшує розмір шрифту надрядкових та підрядкових символів на 0.75 щодо батьківського елемента.

Далі потрібно встановити основні стилі для нашого сайту для робимо його під себе, або як потрібно користувачеві.

```css
body {
  background: url(/assets/images/mmmmm-nnnnnn-bbb-vvv-cccc.jpg) 0 0 /100% no-r
  background-repeat: no-repeat;
  background-size: cover;
  color: rgb(255, 255, 255);
  font-size: var(--body-3);
  overflow: hidden;
}

:focus-visible {
  outline: 2px solid var(--white);
  outline-offset: 2px;
}

::selection { background-color: var(--white-alpha-8); }

::-webkit-scrollbar {
  width: 6px;
  height: 6px; /* for horizontal scrollbar */
}

::-webkit-scrollbar-thumb {
  background-color: var(--white-alpha-8);
  border-radius: var(--radius-pill);
}
```

Рисунок 3.14 – Основний блок “body” та “scroll”

Тут як приклад було встановлено задню картинку, колір тексту та скролл, це прокрутчик, який допомогає нам побачити контент, який не вміщується в наш єкран (рис 3.14).

:focus-visible встановлює стиль для фокусу на елементі під час використання клавіатури, в той час як ::selection задає стиль виділеного тексту сторінці.

::webkit-scrollbar задає стиль для смуг прокручування веб-сторінки в Safari та інших браузерах на основі WebKit, а ::webkit-scrollbar-thumb встановлює стиль для смуги прокручування.

Весь контент у нас буде в контейнері який буде не більше 1600px (рис 3.15).

```css
.container {
  max-width: 1600px;
  width: 100%;
  margin-inline: auto;
  padding: 16px;
}
```

Рисунок 3.15 – Код основного контейнера

Основні класи які були використані в моєму проекті:

- box-sizing

Застосовується для зміни алгоритму розрахунку ширини та висоти елемента.

Властивість успадковується.

content-box - Основується на стандартах CSS, при цьому властивості width і height задають ширину і висоту контенту і не включають значення відступів, полів і кордонів.

border-box - Властивості width і height включають значення полів і кордонів, але не відступів (margin). Ця модель використовується браузером Internet Exporer у режимі несумісності.

padding-box - Властивості width і height включають значення полів, але не відступів (margin) і кордонів (border).

- Padding - внутрішній відступ елемента

При вказівці поля у відсотках значення вважається від ширини батька елемента.

1. Властивість не успадковується.
2. padding: з_всіх_сторон;
3. padding: зверху праворуч знизу ліворуч;
4. padding: зверху_знизу справа_ліворуч;
5. padding: зверху праворуч зліва знизу;
6. Відступ зверху та знизу не діє на термінові теги.

- Margin - зовнішній відступ елемента при вказівці поля у відсотках значення вважається від ширини батька елемента:
    1. Властивість не успадковується.
    2. Значення може бути як позитивним, і негативним числом.
    3. margin: з_всіх_сторон;
    4. margin: зверху праворуч знизу ліворуч;
    5. margin: зверху_знизу справа_ліворуч;
    6. margin: зверху праворуч зліва знизу;
    7. Відступ зверху та знизу не діє на термінові теги.
- Width - встановлює ширину блокових тегів та деяких рядкових (наприклад, img), властивість не успадковується
    1. width:100px;
    2. width:10%.

Max-width - встановлює максимальну ширину блокових тегів та деяких рядкових (наприклад, img). Min-width - встановлює мінімальну ширину блокових тегів та деяких рядкових (наприклад, img). Height - встановлює висоту блокових тегів та деяких рядкових (наприклад, img):

1. Властивість не успадковується.
2. height:100px;
3. height:10%;
4. min-height
5. Властивість не успадковується.
6. max-height
7. Властивість не успадковується.

Overflow - керує відображенням вмісту блокового елемента. Visible - відображається весь вміст елемента, навіть за межами встановленої висоти та ширини.

hidden - Відображається лише область всередині елемента, решта буде прихована. scroll - завжди додаються смуги прокручування. Auto - смуги прокручування додаються лише за необхідності.

- display:

Багатоцільова властивість, яка визначає, як елемент повинен бути показаний у документі. Властивість не успадковується.

block – елемент показується як блоковий. Застосування цього значення для вбудованих елементів, наприклад тега , змушує його вести подібно до блоків - відбувається перенесення рядків на початку і в кінці вмісту.

inline - Елемент відображається як інтегрований. Використання блокових тегів, таких як та <p>, автоматично створює перенос і показує вміст цих тегів з нового рядка. Значення inline скасовує цю особливість, тому вміст блокових елементів починається з місця, де закінчився попередній елемент.

inline - block - Це значення генерує блоковий елемент, який обтікається іншими елементами веб-сторінки подібно до вбудованого елемента. Фактично такий елемент за своєю дією схожий на елементи, що вбудовуються (на зразок тега <img>). При цьому його внутрішня частина форматується як блоковий елемент, а сам елемент як вбудований.

none - тимчасово видаляє елемент із документа. Займане ним місце не резервується і веб-сторінка формується так, як елемента і не було.

Також одним із важливих елементів є медіа запрос, який адаптує наш застосунок під пристрої з меншою роздільною здатністю.

```
 */

@media (min-width: 768px) {

  /**
   * REUSED STYLE
   */

  .container { padding: 24px; }

  .title-1 { --title-1: 2.4rem; }

  .section > .title-2 { margin-block-end: 16px; }

  .card-lg { padding: 24px; }


  .card-sm {
    padding: 20px;
    display: grid;
    grid-template-rows: min-content 1fr;
  }

  .badge {
    top: 20px;
    right: 20px;
  }
```

Рисунок 3.16 – Адаптування під 728 пікселів та менше

Так можна змінити контейнери та стилі під меншу роздільну здатність (рис 3.16).

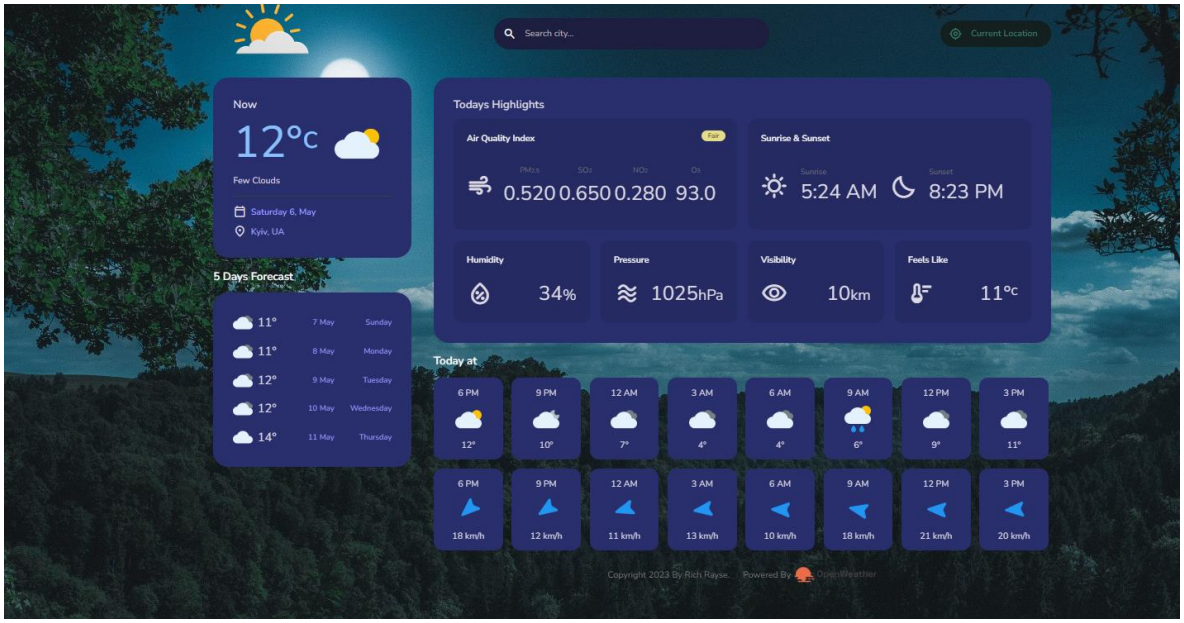


Рисунок 3.17 – Приклад застосунка при повній роздільній здатності (рис 3.17).

Так застосунок буде виглядати при адаптуванні (рис 3.18).

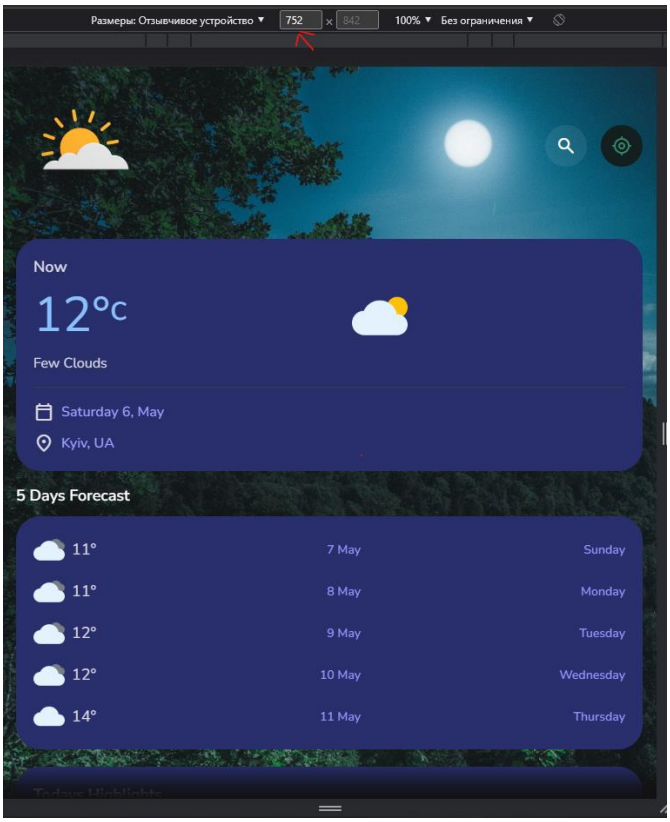
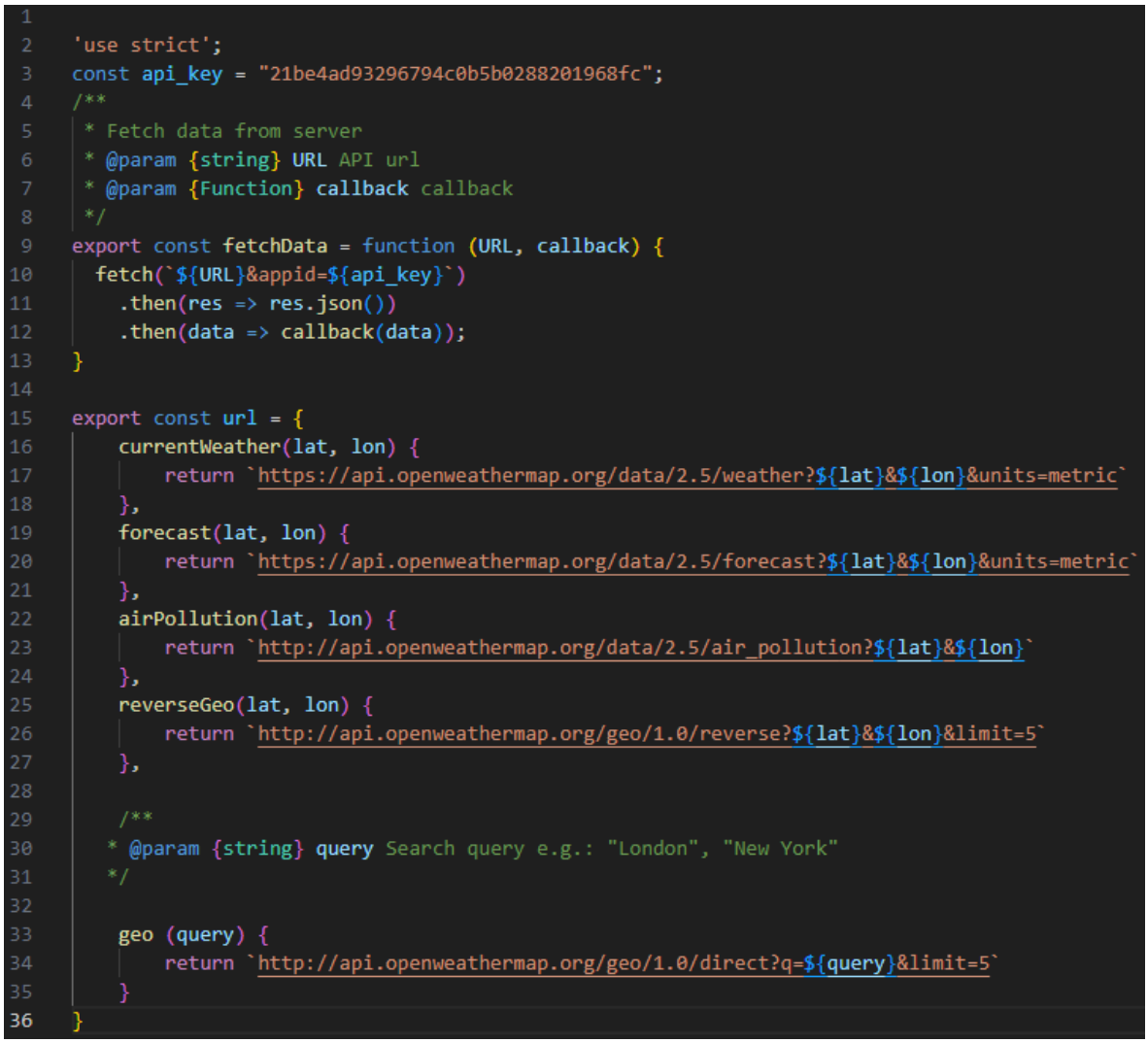


Рисунок 3.18 – Зміна при 728 пікселів та менше

3.3 Розробка логічної частини на JavaScript

Перший файл, який пов’язує наш застосунок с данними які надходять і оновлюються це Api.js (рис 3.19).



```
'use strict';
const api_key = "21be4ad93296794c0b5b0288201968fc";
/**
 * Fetch data from server
 * @param {string} URL API url
 * @param {Function} callback callback
 */
export const fetchData = function (URL, callback) {
  fetch(`${URL}&appid=${api_key}`)
    .then(res => res.json())
    .then(data => callback(data));
}

export const url = {
    currentWeather(lat, lon) {
        return `https://api.openweathermap.org/data/2.5/weather?${lat}&${lon}&units=metric`
    },
    forecast(lat, lon) {
        return `https://api.openweathermap.org/data/2.5/forecast?${lat}&${lon}&units=metric`
    },
    airPollution(lat, lon) {
        return `http://api.openweathermap.org/data/2.5/air_pollution?${lat}&${lon}`
    },
    reverseGeo(lat, lon) {
        return `http://api.openweathermap.org/geo/1.0/reverse?${lat}&${lon}&limit=5`
    },

    /**
   * @param {string} query Search query e.g.: "London", "New York"
   */

    geo (query) {
        return `http://api.openweathermap.org/geo/1.0/direct?q=${query}&limit=5`
    }
}
```

Рисунок 3.19 – Api.js

api_key - це ключ API, який використовується для отримання доступу до даних.

fetchData - функція для отримання даних з API, яка використовує fetch для звернення до сервера OpenWeatherMap, передає ключ API та функціонує зворотний виклик виклику, який буде виконано з отриманими даними в якості параметра.

url - це об'єкт, який містить різні URL-адреси для різних запитів API, таких як поточна погода, прогноз, забруднення повітря тощо.

Методи currentWeather, weather, airPollution, reverseGeo та geo об'єкта url повертають URL-адреси для різних типів запитів до API OpenWeatherMap на основі переданих їм параметрів, наприклад, широти та довго або запиту на ім'я місця (у випадку методу geo).

Отже, цей код дозволяє легко виконувати запити до API OpenWeatherMap, використовуючи методи url об’єкта та функцію fetchData.

Після створення арі потрібно ще декілька файлів для зміни інформації застосунка згідно данних, які надходять, а саме:

- Route.js
- Module.js
- App.js

Route.js - складається з кількох функцій та констант, які допомагають отримати різноманітну інформацію про погоду.

weekDayNames та monthNames: це масиви, які містять назви днів тижня та місяців. getDate: це функція, яка перетворює дату з формату Unix у рядок у визначеному форматі. Вона приймає два параметри: dateUnix, що представляє дату у форматі Unix (кількість секунд, що минули з 1 січня 1970 року), та timezone, що представляє різницю між часовим поясом, в якому розташований користувач, та UTC, також у секундах. Функція повертає рядок з назвою дня тижня, датою та місяцем.

getTime та getHours: ці функції також перетворюють дату з формату Unix у рядок, але вони повертають рядки з часом замість дати. getTime повертає час у форматі "години:хвилини AM/PM", а getHours повертає час у форматі "години AM/PM". mps_to_kmh: це функція, яка приймає значення швидкості у метрах на секунду та повертає еквівалентне значення у кілометрах на годину.

aqiText: це об'єкт, який містить різні ступені якості повітря та повідомлення, які пояснюють, що означає кожен рівень.

url: це об'єкт, який містить функції, які повертають різні URL-адреси для запитів до API OpenWeatherMap. Ці функції приймають координати місця або рядок запиту для гео-кодування і повертають відповідний URL-адрес для запиту до API.

fetchData: це функція, яка виконує (рис 3.20).

```
 */
export const getDate = function (dateUnix, timezone) {
  const date = new Date((dateUnix + timezone) * 1000);
  const weekDayName = weekDayNames[date.getUTCDay()];
  const monthName = monthNames[date.getUTCMonth()];

  return `${weekDayName} ${date.getUTCDate()}, ${monthName}`;
}

/**
 * @param {number} timeUnix Unix date in seconds
 * @param {number} timezone Timezone shift from UTC in seconds
 * @returns {string} Time string. formate: "HH:MM AM/PM"
 */
export const getTime = function (timeUnix, timezone) {
  const date = new Date((timeUnix + timezone) * 1000);
  const hours = date.getUTCHours();
  const minutes = date.getUTCMinutes();
  const period = hours >= 12 ? "PM" : "AM";

  return `${hours % 12 || 12}:${minutes} ${period}`;
}

/**
 * @param {number} timeUnix Unix date in seconds
 * @param {number} timezone Timezone shift from UTC in seconds
 * @returns {string} Time string. formate: "HH AM/PM"
 */
export const getHours = function (timeUnix, timezone) {
  const date = new Date((timeUnix + timezone) * 1000);
  const hours = date.getUTCHours();
  const period = hours >= 12 ? "PM" : "AM";

  return `${hours % 12 || 12} ${period}`;
}

/**
 * @param {number} mps Metter per seconds
 * @returns {number} Kilometer per hours
 */
```

Рисунок 3.20 – Алгоритм із функцій Module.js

App.js та Route.js – пов’язані між собою та виконують алгоритм із зміною таблиць в залежності з надходящею інформацією. Route Модуль містить дві функції, які дозволяють отримувати погоду на основі різних параметрів:

1. CurrentLocation(): використовує місцезнаходження користувача для отримання погоди на поточному місці розташування;
2. SearchedLocation(query): отримує погоду на основі переданого параметру запиту (query), який містить значення широти (lat) та довготи (lon) місця розташування.

Також модуль містить об'єкт routes, який містить маршрути (routes) та функції, які їм відповідають. Функція checkHash() отримує поточний маршрут з URL-адреси та запускає відповідну функцію з масиву маршрутів.

Цей модуль дозволяє користувачам отримувати погоду на основі місцезнаходження або запитів, а також маршрутизує запити на відповідні функції для обробки даних.

Останнім йде App.js, в ньому JavaScript код, який використовується для розробки функціональної частини погодного веб-додатка.

Приклад роботи веб-додатка можна описати наступним чином: користувач може виконувати пошук погоди за містом, або ж відображати погодні умови у поточному місці розташування. На сторінці відображається інформація про погоду, така як поточна температура, іконка погоди, опис погоди, дата та географічні координати місця, де відображається погода.

Детальніший опис того, що робить кожен рядок коду:

- use strict - встановлює режим строгого режиму, який дозволяє використовувати сучасний синтаксис JavaScript, а також забороняє використання небезпечних функцій.
- import - оператор імпортування, який імпортує функції з інших модулів. В даному випадку, імпортується функція fetchData з модуля api.js та всі функції з модуля module.js.
- addEventOnElements - функція, яка приймає список DOM-елементів, тип події та зворотний виклик. Ця функція додає обробник подій до кожного елемента в списку DOM-елементів.

- toggleSearch - функція, яка додає або видаляє клас "active" до елементу DOM, який відповідає за відображення поля введення пошуку погоди.
- searchTimeout - змінна, яка зберігає таймер, який використовується для запиту на сервер, коли користувач вводить запит в поле пошуку.
- serachTimeoutDuration - змінна, яка зберігає тривалість таймера для пошуку на сервері.
- searchField - елемент DOM, який відповідає за відображення поля введення пошу

3.4 Програмування датчика dht11 для показників температури на базі Arduino

Для достовірності пронрозування погоди було створено схему на базі Arduino Uno, де основною частиною був датчик DHT11.

Датчик DHT11 – це цифровий датчик температури та вологості, що дозволяє калібрувати цифровий сигнал на виході. Складається з ємнісного датчика вологості та термістора. Також датчик містить в собі АЦП для перетворення аналогових значень вологості та температури (рис 3.21).

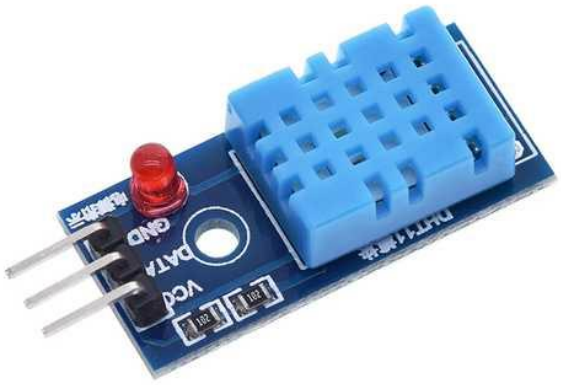


Рисунок 3.21 – Датчик DHT11

За допомогою віртуального середовища проектування Proteus, було створено повноцінну схему, яка могла зчитувати вологість та температуру навколо себе. За допомогою цього датчика та схеми, можемо довести чи правильно у нас працюють прогнозування. Треба спрогнозувати погоду в місті в якому потрібно, зняти показники температури за допомогою web – додатка та звірити показники які показує датчик. Виглядає це наступним образом (рис 3.22) [14].

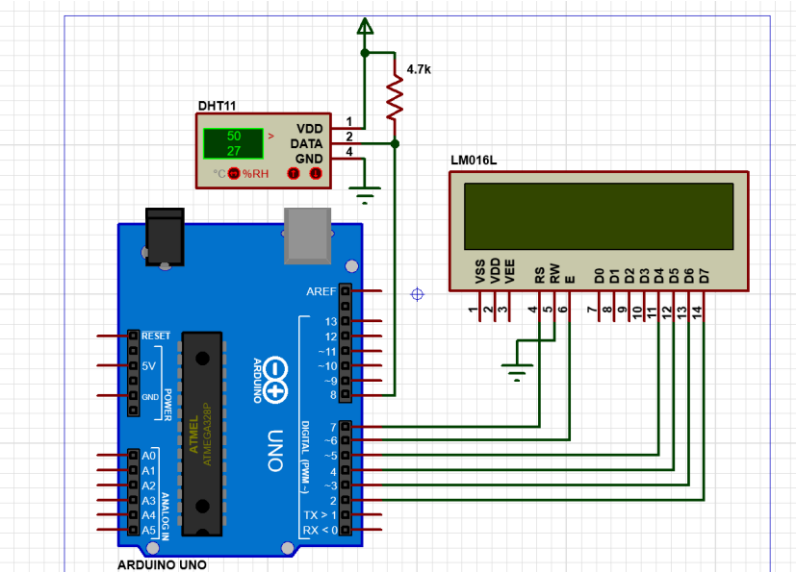


Рисунок 3.22 – Створена схема з використанням датчика

Схема знаходиться у вимкненому стані, але якщо увімкнути її, можливо буде зняти показники вологості та температури навколо.

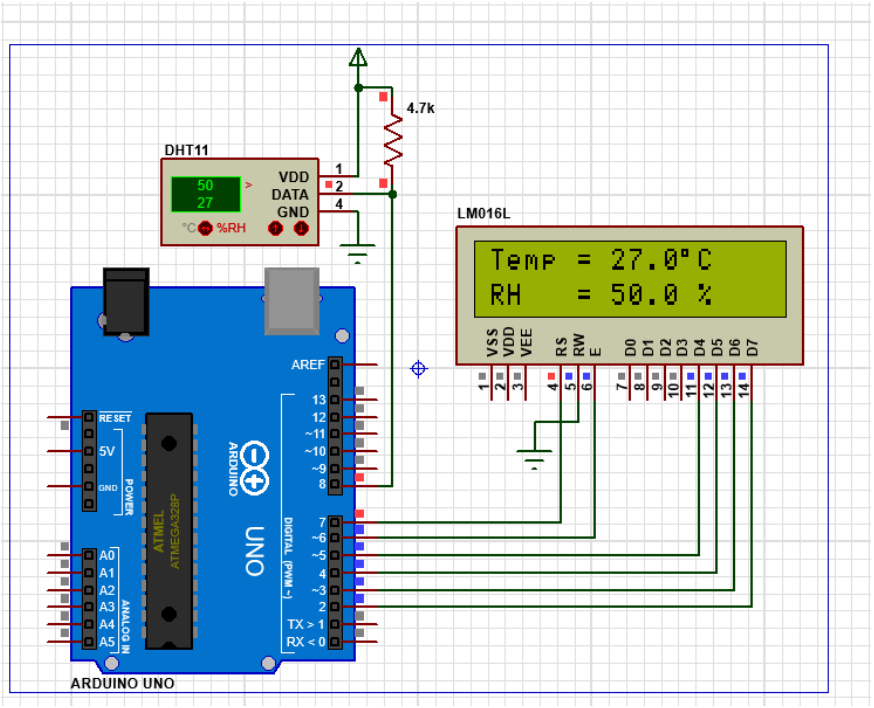


Рисунок 3.23 – Схема у робочому стані

У схемі було використано Arduino Uno, датчик температури DHT11 та дісплей для виводу інформації. Схему Arduino Uno було запрограмовано на взаємодію с датчиком наступим чином:

```
// include LCD library code
#include <LiquidCrystal.h>
// include DHT library code
#include "DHT.h"

#define DHTPIN 8            // DHT11 data pin is connected to Arduino pin 8

// LCD module connections (RS, E, D4, D5, D6, D7)
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);

#define DHTTYPE DHT11       // DHT11 sensor is used
DHT dht(DHTPIN, DHTTYPE);   // Initialize DHT library

char temperature[] = "Temp = 00.0 C  ";
char humidity[]    = "RH   = 00.0 %  ";
void setup() {
  // set up the LCD's number of columns and rows
  lcd.begin(16, 2);
  dht.begin();
}

void loop() {
  delay(1000);           // wait 1s between readings
  // Read humidity
  byte RH = dht.readHumidity();
  //Read temperature in degree Celsius
  byte Temp = dht.readTemperature();

  // Check if any reads failed and exit early (to try again)
  if (isnan(RH) || isnan(Temp)) {
    lcd.clear();
    lcd.setCursor(5, 0);
    lcd.print("Error");
    return;
  }

  temperature[7]     = Temp / 10 + 48;
  temperature[8]     = Temp % 10 + 48;
  temperature[11]    = 223;
  humidity[7]        = RH / 10 + 48;
  humidity[8]        = RH % 10 + 48;
  lcd.setCursor(0, 0);
  lcd.print(temperature);
  lcd.setCursor(0, 1);
  lcd.print(humidity);
}
```

Рисунок 3.24 – Код Arduino

Цей код призначений для взаємодії між платформою Arduino та датчиком вологості та температури DHT11. Він також включає бібліотеку LiquidCrystal для роботи із РК-дисплеєм. Для роботи з датчиком використовується бібліотека DHT. У setup() функції ініціалізуються налаштування, такі як ініціалізація РК-дисплея та датчика DHT (рис 3.24).

Функція loop() виконується нескінченно та зчитує показання температури та вологості кожну секунду (затримка 1000 мс). Якщо зчитування не вдалося, на дисплеї відображається повідомлення про помилку.

Потім відбувається обробка лічених даних, в якій значення температури і вологості конвертуються символьні рядки для виведення на РК-дисплей. Символи для виведення градусного знака та відсоткового знака кодуються в ASCII-коді.

Нарешті символьні рядки виводяться на РК-дисплей за допомогою lcd.print().

Таким чином можна переверіти погоду навколо та звіритися наскільки є неточність [15].

Висновок до розділу 3:

У результаті вивчення теми "програмна реалізація web-застосунку для прогнозування погоди" можна зробити висновок, що розробка такого застосунку є важливою задачею для багатьох галузей, включаючи сільське господарство, транспорт, будівництво та інші. Для створення відповідного web-застосунку потрібно використовувати різноманітні програмні інструменти та технології, такі як мови програмування (наприклад, JavaScript), фреймворки (наприклад, React), сервіси для отримання даних про погоду (наприклад, OpenWeatherMap) та інші.

Загалом, розробка web-застосунку для прогнозування погоди є складною задачею, яка потребує використання різних технологій та інструментів. Однак, з урахуванням зростаючого попиту на точні прогнози погоди, такий застосунок може стати дуже корисним для багатьох користувачів, які хочуть бути завжди в курсі погодних умов.

РОЗДІЛ 4 ПРАКТИЧНЕ ВИКОРИСТАННЯ WEB–ЗАСТОСУНКУ ДЛЯ ПРОГНОЗУВАННЯ ПОГОДИ

4.1 Прогнозування погоди за допомогою web–застосунку

Web – застосунок запрограмовано, тепер можна подивитися на результат виконаної роботи (рис 4.1) (рис 4.2).

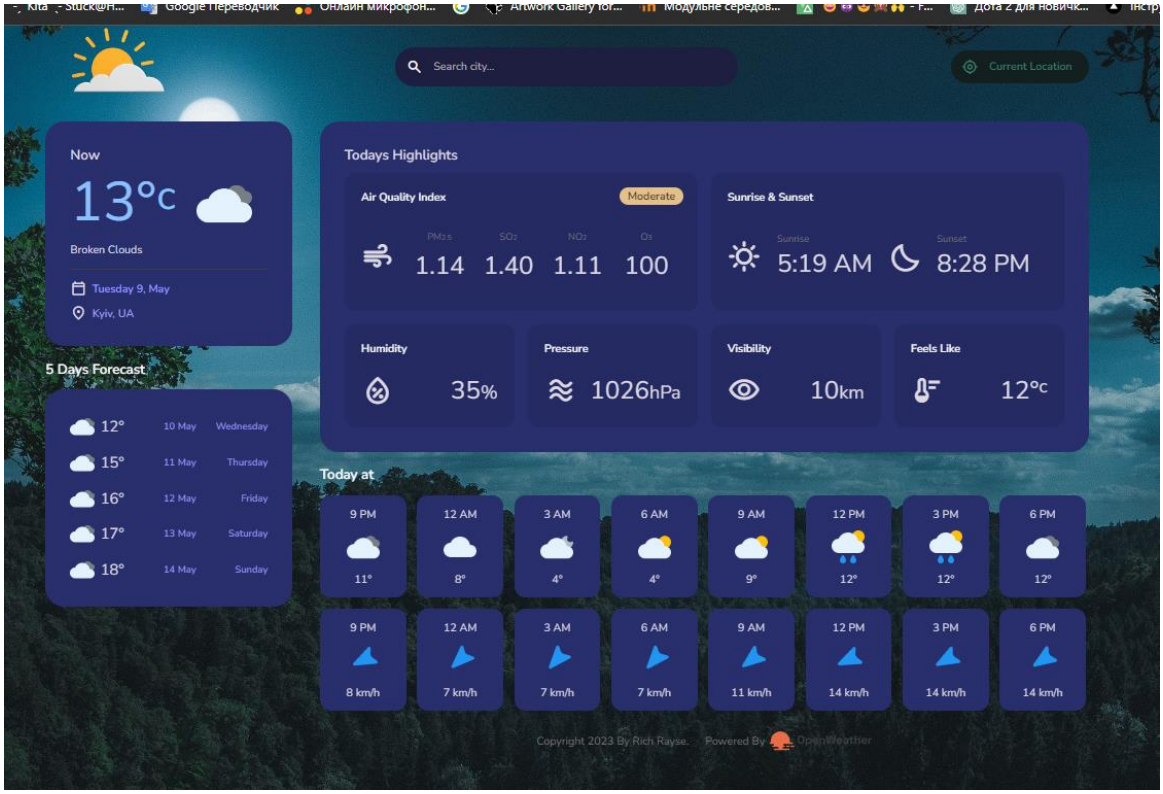


Рисунок 4.1 – Прогнозування в Києві

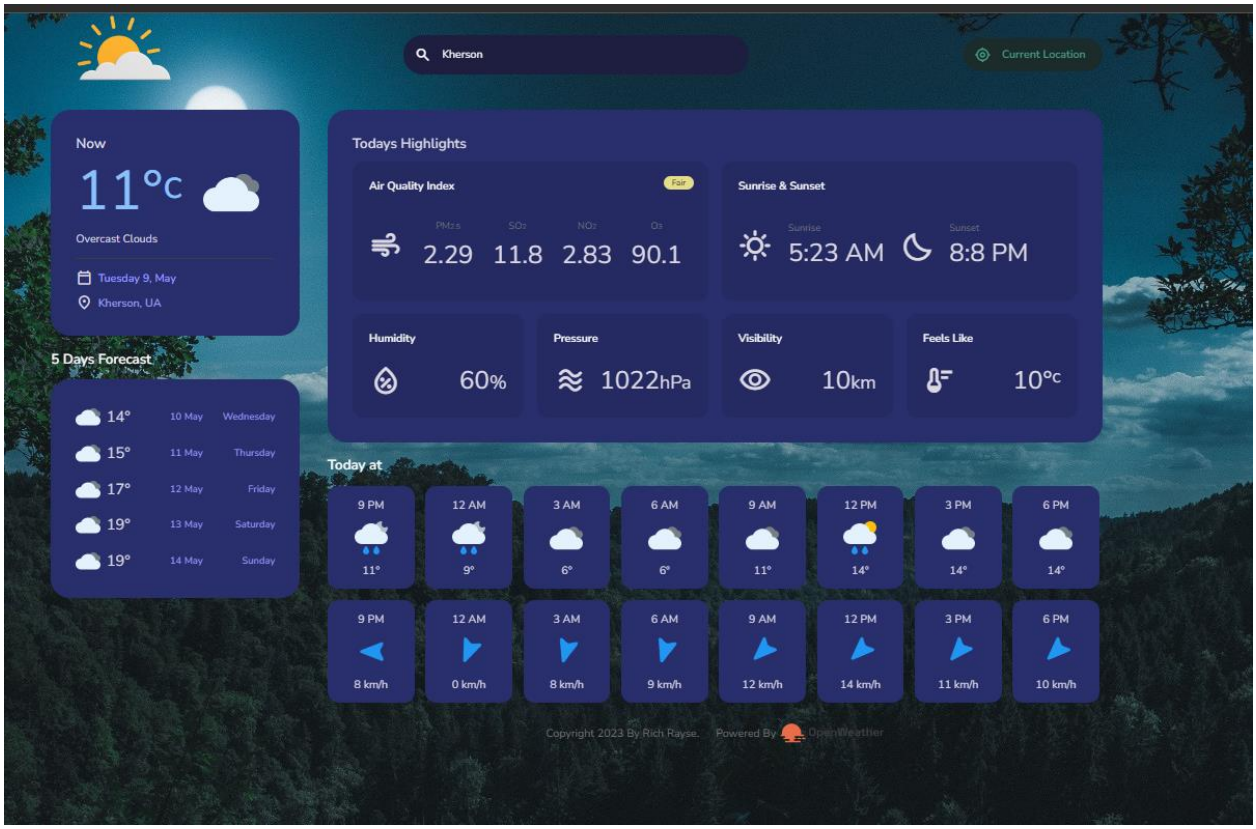


Рисунок 4.2 – Прогнозування в Херсоні

Останнім візьмем прикладом столицю Німеччини – Берлін та подивимось всі данні про це місто (рис 4.3).

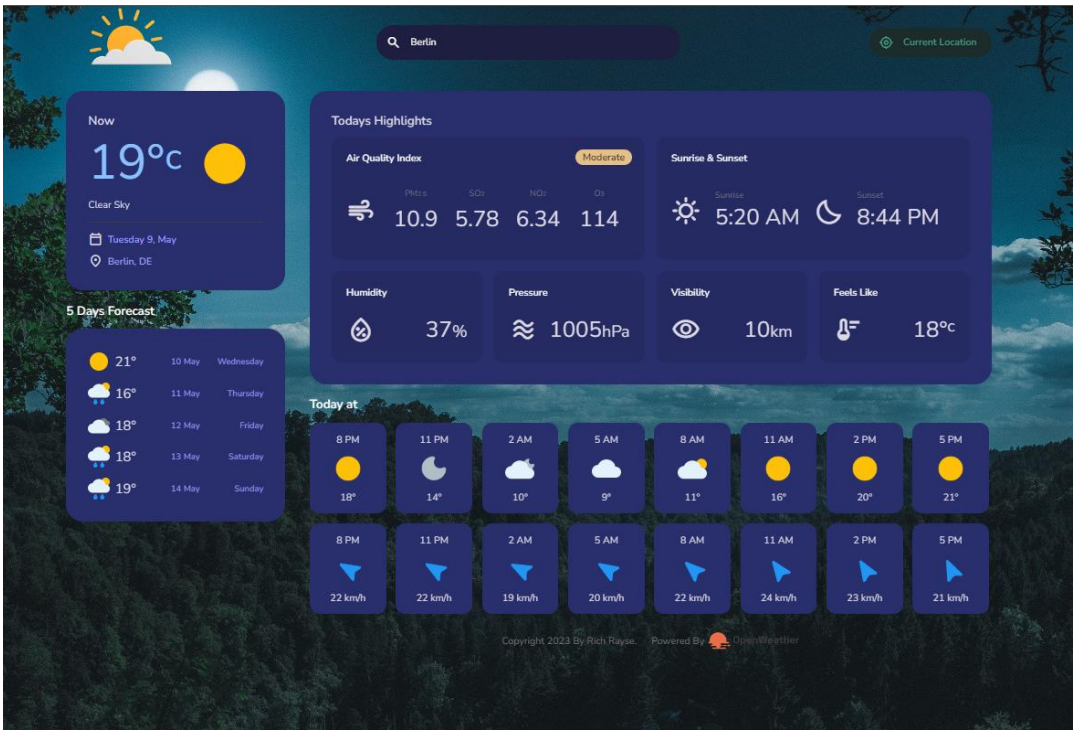


Рисунок 4.3 – Прогнозування в Берліні

Висновок по розділу 4:

Основна мета розробки web-застосунку для прогнозування погоди полягає в тому, щоб користувачі могли отримувати актуальну та точну інформацію про погоду в обраному регіоні в режимі реального часу. Для цього необхідно мати доступ до баз даних з інформацією про погоду, що забезпечується за допомогою API.

ЗАГАЛЬНІ ВИСНОВКИ

1. На основі проведених досліджень розроблено web-застосунок для прогнозування погоди, що надає щоб користувачам можливість отримувати актуальну та точну інформацію про погоду в обраному регіоні в режимі реального часу, при наявності доступу за допомогою API до баз даних з інформацією про погоду.

2. Перщий розділ дав зрозуміти що, аналіз існуючого інструментарію розробки web-додатків дозволяє визначити найбільш підходящі технології створення високоякісних і ефективних додатків. При розробці web-додатків важливо враховувати потреби користувачів та забезпечувати їх зручність та комфорт.

3. Існує безліч інструментів та технологій, які можуть використовуватись для розробки веб-додатків, таких як фреймворки, бібліотеки, мови програмування та інші. Вибір інструментарію залежить від вимог проекту та відповідальності розробника за результат.

4. Однією з основних вимог до інструментарію є можливість інтеграції з іншими технологіями та розширення функціональності програми. Важливо вибрати технології, що дозволяють розширювати функціональні можливості програми та забезпечувати її стабільність та безпеку.

5. Таким чином, аналіз існуючого інструментарію дозволяє вибрати найбільш підходящі технології для розробки web-додатків та забезпечити їх ефективність та високу якість.

6. З результатів проведеного аналізу різних інструментів для розробки веб-застосунків для прогнозування погоди можна зробити висновок, що вибір відповідного інструменту залежить від конкретних потреб користувачів та розробників. При виборі відповідного інструмента слід враховувати такі фактори, як масштабованість системи, продуктивність, масштабованість та

підтримка спільноти розробників. Для розробки більшості погодних веб-застосунків найкращим вибором можуть бути відкриті та безкоштовні інструменти, такі як HTML, CSS, JavaScript та бібліотеки, такі як jQuery, React та Bootstrap. Спеціалізовані фреймворки та платформи, такі як Django, Flask, Node.js та AWS, також можуть бути використані залежно від вимог до функціональності та продуктивності.

7. Таким чином, вибір інструментів для створення веб-програмної програми для прогнозування погоди повинен здійснюватися відповідно до конкретних потреб і вимог системи і може бути зроблений шляхом ретельного аналізу та порівняння різних інструментів**.**

8. У результаті вивчення теми "програмна реалізація web-застосунку для прогнозування погоди" можна зробити висновок, що розробка такого застосунку є важливою задачею для багатьох галузей, включаючи сільське господарство, транспорт, будівництво та інші. Для створення відповідного web-застосунку потрібно використовувати різноманітні програмні інструменти та технології, такі як мови програмування (наприклад, JavaScript), фреймворки (наприклад, React), сервіси для отримання даних про погоду (наприклад, OpenWeatherMap) та інші.

9. Загалом, розробка web-застосунку для прогнозування погоди є складною задачею, яка потребує використання різних технологій та інструментів. Однак, з урахуванням зростаючого попиту на точні прогнози погоди, такий застосунок може стати дуже корисним для багатьох користувачів, які хочуть бути завжди в курсі погодних умов.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загальна інформація - https://ua.wikipedia.org/

2. Відомості про css - https://developer.mozilla.org

3. Селектори css - https://codeguida.com/post/69

4. Принципи web - https://sites.google.com/

5. Архітектура веб-застосунків - https://codeguida.com/post/1394

6. Інформація по css - https://css.in.ua/

7. Розробка веб-застосунків - https://webstudio2u.net/ua/site-develop/641-razrabotka-veb-prilozheniy.html

8. Інструменти та веб-ресурси - https://techblog.sdstudio.top/

9. Інформація про розробку - https://internetdevels.ua/

10. Відомості про прогноз погоди - https://www.bbc.com/ukrainian

11. Веб-програмування - http://apeps.kpi.ua/web-programuvania/en

12. Чому веб-програмування - http://www.itschool.vn.ua/

13. Відомості про веб-програмування - http://kiis.knu.ua/

14. Arduino - https://simple-circuit.com/arduino-dht11-sensor-lcd-proteus/

15. Simulate Arduino - https://maker.pro/arduino/projects/how-to-simulate-arduino-projects-using-proteus

16. Класифікація веб-застосунків - https://docplayer.net/50446947-4-7-tipi-veb-storinok-klasifikaciya-veb-saytiv.html

17. Веб-сайти - https://dou.ua/lenta/articles/web-development-status-2020/

18 – Різниця між сайтом та додатком - https://webcase.com.ua/

Додаток А

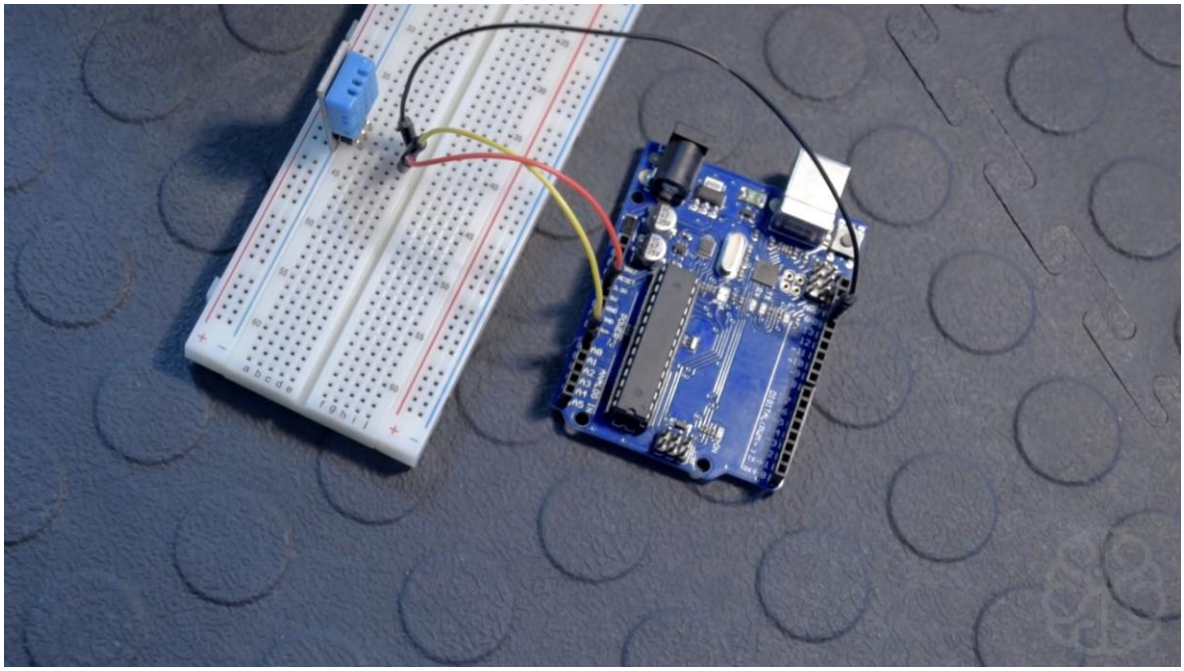


Рис. А.1. Схема на базі датчика dht11 у реальному житті

Додаток Б

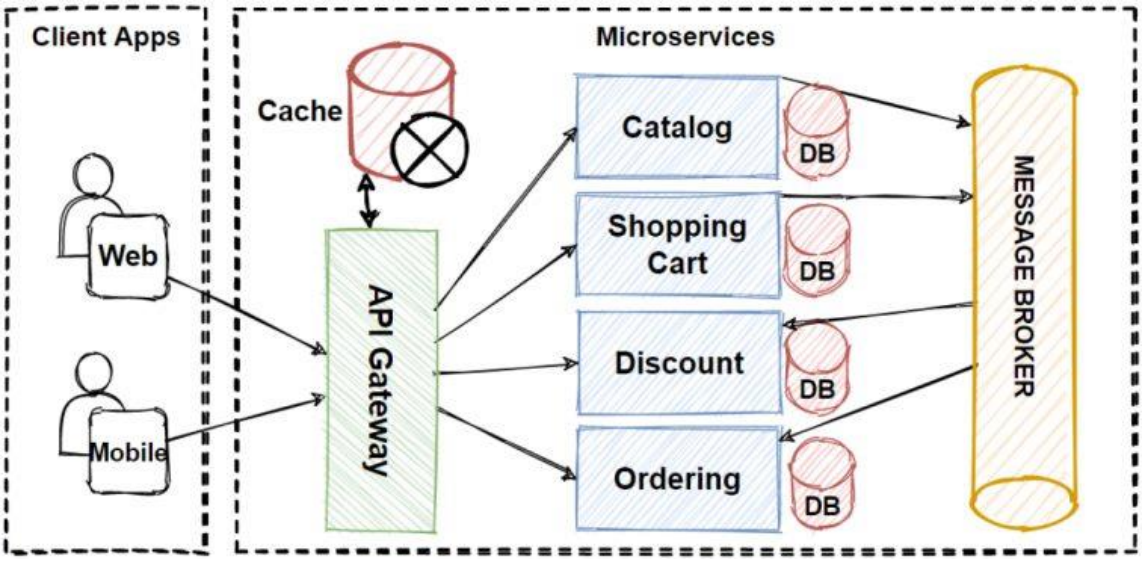


Рис. Б.1. Схема архітектури web – застосунку