МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛИТВИНЮК ВОЛОДИМИР СЕРГІЙОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____ Тетяна НЕСКОРОДЄВА
«_____» _____ 2022р.

РОЗРОБКА СИСТЕМИ ДЛЯ ПОШУКУ ТА ОБРОБКИ ДАНИХ ПРО ТЕХНІЧНІ НЕСПРАВНОСТІ АВТОМОБІЛЯ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Римар П. В., старший викладач
кафедри інформаційних технологій
Науковий консультант:
Нескородєва Т.В.,
д-р техн. наук, доцент
_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

Анотація

Литвинюк В.С. Ця магістерська робота присвячена розробці веб-додатку для пошуку інформації про недоліки та переваги автомобіля. Так, як на даний момент технології розвиваються дуже стрімко та швидко, усі намагаються полегшити процеси, які до недавнього часу займали дуже велику кількість часу та сил. Ці інновації впроваджуються у всі сфери життя людини від побутових задач до складних і неординарних процесів. Саме з ціллю полегшити процес пошуку інформації про потрібний автомобіль було створено цей веб-додаток та проведено дану наукову роботу.

У вступі наведено актуальність розробки такого веб-додатку, в яких ситуаціях він був би корисним та які механізми та інструменти було вивчено в ході роботи.

Другий розділ присвячено вибору інструментів для розробки та їх детальному опису. Також там розглянуто їх функціонал та обґрунтовано чому саме цей інструмент обрано серед аналогів. Наведено усі переваги та недоліки використання інструментів для веб-розробки та які є альтернативи.

Третій розділ містить в собі процес розробки самого веб-додатку та його тестування й налагодження. В ньому детально описано який функціонал притаманний цій системі та як розробляти подібні системи. Розписано, як розроблявся та будувався дизайн додатку. Також описано методи завдяки яким працює додаток. Загалом описано весь процес від першої до останньої стрічки написаного коду та обґрунтовано чому це виконано саме цим шляхом і які в цьому плюси та переваги.

У висновку підбитий підсумок проведеної роботи та обґрунтована значимість цієї системи. Наведено приклади ситуацій в яких цей додаток буде дуже корисним та допоможе зберегти час своїм функціоналом.

Ключові слова: інформація, автомобіль, HTML, CSS, SCSS, недолік, пошук.

Lytvynyuk V.S. This master's thesis is devoted to the development of a web application for finding information about the disadvantages and advantages of a car. Since technology is currently developing very rapidly and quickly, everyone is trying to facilitate processes that, until recently, took a lot of time and effort. These innovations are implemented in all areas of human life, from everyday tasks to complex and extraordinary processes. It was with the aim of facilitating the process of finding information about the desired car that this web application was created and this scientific work was carried out.

The introduction shows the relevance of developing such a web application, in which situations it would be useful, and which mechanisms and tools were learned during the work.

The second section is devoted to the selection of development tools and their detailed description. Their functionality is also discussed there and it is justified why this particular tool was chosen among analogues. Here are all the pros and cons of using web development tools and what the alternatives are.

The third section contains the process of developing the web application itself and its testing and debugging. It describes in detail what functionality is inherent in this system and how to develop similar systems. It is described how the design of the application was developed and built. The methods by which the application works are also described. In general, the entire process from the first to the last tape of the written code is described, and it is justified why it was done this way and what are the advantages and disadvantages.

The conclusion sums up the work done and substantiates the importance of this system. There are examples of situations in which this application will be very useful and will help save time with its functionality.

Keywords: information, car, HTML, CSS, SCSS, flaw, search.

Зміст

Вступ

Актуальність роботи: На сьогоднішній день людство прогресує із неймовірною швидкістю. Не так давно ми не мали уявлення про автомобіль, а сьогодні це повсякденність. Якщо переглянути статистику від 2014 року існують країни у яких кількість авто більша за кількість громадян. Наприклад Сан-Марино, де на 1000 осіб припадало 1263 автівки. Власний автомобіль – це дуже хороший помічник та друг, який неймовірно полегшує життя та надає дуже багато нових можливостей. Автомобільна індустрія розвивається із року в рік все швидше і швидше та якщо декілька десятиліть назад усі марки авто можна було перерахувати на пальцях, то сьогодні це неможливо, не кажучи про усі існуючі моделі авто.

Переважна більшість людей не є авто-експертами і не мають уявлення про усі тонкі моменти та підводне каміння у конструкції автомобіля. Кожне авто по своїй суті це унікальний апарат із безліччю агрегатних вузлів та механізмів. Існують автомобілі «co-platformer», це означає, що ці автомобілі побудовані на одні і тій самій платформі, але це не означає, що вони однакові чи навіть схожі. Це означає лише те, що деякі деталі ходової частини є взаємо замінними з одного авто на друге. Тому з усього вищесказаного можем зробити висновок, що кожна модель авто є унікальною.

Автомобіль – це така річ яка може використовуватись не один, не два і навіть не двадцять років. Все залежить від того, як його експлуатувати та як за ним доглядати, але існують і так звані «болячки», проблематичні місця які були такими при конструюванні і залишились при виробництві. Прикладом таких болячок може бути швидке проявлення корозії на кузові через те, що шар лакового та фарбового покриття занадто тонкий, чи кузов не проходив належної антикорозійної обробки при виготовленні автівки на заводі, але ці болячки можуть бути пов’язані не тільки із візуальними пошкодженнями, а й із електронікою чи механічною частиною. Наприклад невдалий блок управління автоматичною коробкою передач чи невдалий механізм охолодження моторного відсіку через, що може виникати перегрів мотора за певних обставин. І такі

випадки є майже в кожного авто, просто десь їх дуже мало, а десь нереально багато.

Виходячи з того, що більше ніж 90% людей які мають авто не є авто-експертами та не знаються у авто тематиці достатньо добре, вони натикаються на проблеми в ситуаціях при виборі, яке ж авто купити і на які місця звернути увагу у конкретній моделі конкретної марки автомобіля. Ці проблеми частково вирішують форуми автолюбителів де є дуже багато обговорень про автомобілі і саме там можна знайти інформацію про авто, також є медіа ресурси із оглядами від авто-експертів на конкретні автівки. Автоматизувати процес пошуку цієї інформації може дуже сильно полегшити життя людям та зберегти багато часу вільним від серфінга інтернету із метою вичитати якісь нюанси про дане авто.

Мета дослідження: Аналіз інформації та розробка системи для пошуку інформації про технічні несправності авто.

Завдання дослідження:

- Пошук та огляд аналогів
- Ознайомлення із інструментами розробки
- Ознайомлення із інструментами робити із сторонніми сервісами
- Розробка системи для пошуку інформації про технічні несправності авто
- Тестування системи
- Реліз системи

Об’єкт дослідження: Різноманітні недоліки у конструкції чи механізмах та плюси автомобіля.

Предмет дослідження: Система, що дозволяє швидше, легше та зручніше знаходити потрібну інформацію про недоліки та позитивні моменти даного автомобіля.

Практичне значення одержаних результатів: створення незалежного додатку та взаємодія із користувачем для пошуку інформації.

Структура кваліфікаційної роботи: Магістерська робота складається із вступу, трьох розділів та висновків до них, списку використаних джерел та одного додатку.

У першому розділі магістерської роботи наведено інформацію про проблематику цієї задачі, огляд аналогів та постановка задачі, яку потрібно вирішити.

У другому розділі проведено детальний опис та дослідження інструментів, що було використано для розв’язання поставленої задачі.

У третьому розділі проведено розглянуто процес розробки системи та вирішення проблема за допомогою використання інструментів з другого розділу.

РОЗДІЛ 1. ПРОБЛЕМАТИКА. ОГЛЯД АНАЛОГІВ. ПОСТАНОВКА ЗАДАЧІ.

В даному розділі буде описана проблематика, огляд вже існуючих аналогів цієї системи та постановка задачі, яку потрібно вирішити.

1.1 Проблематика

На сьогоднішній день із плином прогресу люди винаходять все більше і більше різноманітних гаджетів, пристроїв та агрегатів, але інколи вирішення або полегшення в якісь ситуації викликає інші складнощі чи проблеми. Можна провести паралель із автомобілем.

Авто – це неймовірний винахід людства так як це не аби як полегшує життя та робить його комфортнішим, але кожен автомобіль вимагає необхідного догляду, обслуговування там турботи за ним. Він вимагає час від часу проводити регламентні роботи та маніпуляції, такі як заміна мастила двигуна, коробки перемикання передач та редукторів, заміна прокладок, втулок та ущільнювачів, також такими являються фільтра салону, мастила двигуна, коробки перемикання передач та повітряного фільтра. До цих самих моментів обслуговування можна додати обслуговування гальмівної системи, а саме заміна колодок, патрубків та дисків. Це все є розхідними матеріалами та має мінятися або по регламенту, або ж по закінченню функціональної частини.

Та це не все, із чим стикається власник автомобіля при його експлуатації. На станціях технічного обслуговування здійснюється ще ремонтні роботи. Це вже зовсім інший вид робіт. Вони не є регламентними та трапляються не так часто та не періодично. Зазвичай такі роботи, якщо автомобіль зламався не по вині власника, а через брак чи заводський дефект, роблять по гарантії на офіційних дилерських центрах, за умови, що авто ще на гарантійному обслуговуванні. У кожного авто-бренду ці умови різні. Дехто надає гарантію на 5 років або 100 тисяч кілометрів пробігу, дехто більше, а дехто менше.

Після закінчення строку гарантійного обслуговування власник авто вже за власні кошти ремонтує своє авто. Іноді чек за ремонт автівки складає декілька сотень гривень, а інколи і декілька тисяч доларів США, все залежить від того, що ж саме трапилось з автомобілем та які об'єми робіт було проведено. Також цінники на окремі деталі іноді сягають половини вартості авто, особливо якщо автомобіль вже 15 років та більше їздить по дорогам загального користування.

Слід розуміти, що кожен автомобіль маю різну конструкцію та механізми. Та конструюють їх люди, яким властиво допускати якісь помилки та не бачити недоліки. Саме тому в кожного автомобіля є свої болячки чи проблематичні місця. Цими болячками може бути будь що, від лакового та фарбового покриття до двигуна чи трансмісії. Авто-експерти про них знають та знають як з ними боротися, як експлуатувати авто, щоб їх прояви були мінімальні та як їх усувати назавжди якщо це можливо, але звичайні люди не такі обізнані в цих питаннях.

Для цього люди створили різноманітні авто форуми та блоги, де роблять відео огляди на автомобілі, діляться інформацією по її експлуатації та багато чого іншого. Але коли постає питання покупки авто, було б дуже добре знати всі ці тонкості та нюанси по машині до того, як купити. Адже кожному хотілось би купити авто та отримувати від нього задоволення, а не навпаки.

Наразі для того, щоб отримати інформацію про плюси та мінуси автомобіля треба прогортати дуже багато різноманітних статей, блогів та форумів. Це займає нереально багато часу та нервів. Сервіс який представлено в цій роботі набагато полегшує життя тим людям, які хочуть дізнатися про підводні камені та плюси до покупки автомобіля, та визначитись чи взагалі їм потрібна така автівка.

1.2 Огляд аналогів

Наразі вітчизняних аналогів не існує, але давайте оглянемо дуже схожі із ідеологією сервіси.

Auto RIA

Рис. 1.1 Авто Ріа

В якості першого аналогу можна розглянути усім відомий сервіс від не менш відомої компанії Auto RIA. Цей сервіс першочергово призначений для продажу та купівлі автомобілів, але власники авто можуть писати тут відгуки та рецензії на них. Також можна проставляти у деяких категоріях оцінки від 0 до 5 та таким чином відзначати, що вам сподобалось найбільше, а що сподобалось найменше чи взагалі не сподобалось. Ось список критеріїв за якими можна виставляти оцінки:

- Керування
- Надійність
- Комфорт
- Ціна
- Дизайн

На мою особисту думку такі критерії, як Ціна та Дизайн не є об’єктивними так як із плином часу ціни на автомобілі змінюються, а старі відгуки залишаються і впливають на загальну статистику, а дизайн це взагалі суб’єктивний критерій для оцінювання.

Переваги:

- Велика база користувачів
    - Цей сервіс відомий по всій країні. Він являється по сумісності найбільшою платформою з продажу автомобілів, якою користуються усі хто хоче продати своє авто.
- Багатофункціональність
    - Вона полягає в тому, що цей сервіс надає не тільки інформативний функціонал, а також є можливість продавати/купувати автомобілі та інші транспортні засоби, від мотоцикла до човнів та літаків.
- Найбільш інтуїтивно зрозумілий дизайн
    - Дизайн виконано без непотрібної інформації. Він містить лише найбільш необхідну інформацію та користувач, який вперше зайшов на цей сервіс інтуїтивно розуміє де знаходиться та чи інша кнопка
- Можливість створювати власні підписки
    - Ця можливість звільняє вас від постійного моніторингу дошки оголошень, а самостійно буде надсилати вам повідомлення про якісь нові оголошення чи зміни в оголошенні за яким відбувається слідкування.
- Доступність
    - Цей сервіс має досить велику підтримку крос платформи. Він реалізований для будь-якої системи: Apple, Android, Web.

Недоліки:

- Відгуки не зовсім достовірні
    - Відгуки про автомобілі може залишати будь-хто, таким чином достовірність відгуків під сумнівом.
- Оцінювання
    - Ставити оцінки за критеріями також може ставити будь-хто, тому вони теж можуть бути сумнівними.

Avto.pro

Рис. 1.2 Авто Про

В якості наступного аналогу хотілося б розглянути сервіс Avto.pro. Цей сервіс більше для пошуку та купівлі автозапчастин, але також він має окрему вкладку «Форум» де є окремі гілки для обговорення певної теми. Також окрім форуму та продажу автозапчастин є ще вкладка з новинами авто світу. Сервіс має гарну онлайн підтримку та інтуїтивність навігації по сервісу.

Переваги:

- Зрозумілість дизайну
  - Дизайн максимально простий та не переповнений непотрібною інформацією. Мінімум реклами – максимум потрібного контенту.
- Велика база користувачів
  - Сервіс являється найбільшим інтернет-дистриб’ютором автозапчастин в Україні. Тому про нього знають всі хто хоч раз шукав автозапчастини для свого авто власноруч.

Недоліки:

- Об’єктивність
  - Сервіс надає можливість чатитись людям між собою у вигляді форуму, але ніхто не дає гарантії, що до достовірності інформації поданої на форумі адже далеко не всі люди авто-експерти.

- Пошук
    - Для того, щоб знайти потрібну інформацію треба витратити дуже багато часу на перегляд тем форуму та їх контенту.

1.3 Постановка задачі

Необхідно розробити систему для пошуку інформації про недоліки автомобіля. Система має бути реалізована на мові програмування javascript ES6+ із використанням сучасних фреймворків та бібліотек. Також має бути реалізована база даних для збереження інформації там її надання згідно запиту. Дизайн має бути інтуїтивно зрозумілий та простий(не напружений зайвою інформацією). Реалізувати логіку таким чином, щоб сервіс мав змогу збирати інформацію про обраний автомобіль із різних сервісів та надавати її користувачу разом з інформацією яка буде знаходитись у локальній базі даних. Кожен запит має формуватися на основі обраних варіантів із списків, які відповідають за марку, модель, кузов автомобіля.

Система включає в себе наступні ключові моменти:

1) Надання інформації що до авто
2) Підтримка усіх сучасних браузерів
3) Інтуїтивно зрозумілий дизайн
4) Надсилання запитів до інших сервісів та отримання відповідей у вигляді інформації про авто

Доступний функціонал:

1) Вибір марки, моделі та кузова автомобіля
2) Отримання потрібної інформації
3) Зв’язок із розробником

Результати не зберігаються та їх можна переглянути лише один раз після надсилання запиту

Висновки до 1 розділу.

В даному розділі була чітко сформована постановка задачі та визначена функціональність системи підтримки прийняття рішень, яку було розроблено та досліджено.

Також було оглянуто декілька подібних сервісів. Було чітко описані їх переваги та недоліки в порівнянні із конкурентами в суміжних галузях. Дана система не має прямих аналогів на українському ринку в принципі, саме тому її можна назвати унікальною. Це завдяки унікальному функціоналу, який не присутній в жодному із представлених сервісів. Цей функціонал дозволяє отримати інформацію із додатків-конкурентів і паралельно надавати частину своєї інформації.

РОЗДІЛ 2. ТЕХНОЛОГІЇ ТА МЕТОДИ

Перед написанням будь-якого коду потрібно визначитись із середовищем де це робити, так званим IDE. IDE – це Integrated Development Environment (інтегроване середовище розробки). Це комплексне рішення для розробки, яке складається із редактора коду, інструментів для складання, налагодження та автоматизації програм. Більшість із них мають вбудований компілятор, інтерпретатор коду та можливість автодоповнення коду, але деякі не містять жодного із переліченого, все залежить від спеціалізації IDE. IDE для об'єктно-орієнтованого програмування зазвичай мають вбудовані інспектор класів, інспектор об'єктів та відстеження схеми ієрархії класів для полегшення програмування об'єктно-орієнтованим шляхом.

Для веб-розробки сьогодні доступно неймовірно багато інструментів, технологій та засобів. Вони діляться на 2 категорії. Перші відповідають за структуру(скелет) сайту та допомагають краще, ефективніше та швидше зверстати структуру веб сторінки. Другі відповідають за стилі та дизайн сайту. Вони допомагають зручніше та краще організувати написання стилів для сторінок, елементів сторінки та всього іншого.

Засоби та інструменти що відповідають за логіку та функціонал. Перш за все це мови програмування, які можливо використовувати для розробки веб-сервісів та веб-додатків. Однією з найвідоміших є JavaScript. Але для неї також існує неймовірна кількість фреймворків та бібліотек для того, щоб більш оптимізовано виконувати ті чи інші задачі. Також є TypeScript, що є окремою мовою програмування, але він є наслідником JavaScript та має більш розширені можливості.

Інструментарій для роботи із даними. Сюди можна віднести, як бібліотеки для мов програмування, які надають можливість надсилати, обробляти та віддавати дані так і засоби роботи із базами даних та самі бази даних. Бази даних є багатьох видів. Наприклад реляційні, мережеві та ієрархічні та інші. В залежності від того у якому вигляді та обсязі вам потрібно тримати дані може

змінюватись і тип бази даних. Сервісів по створенню власної бази даних є неймовірна кількість, але можна її створити і власноруч.

Також не слід забувати про інструменти, які просто на просто полегшують життя програміста. Сюди можна віднести різноманітні препроцесори, бібліотеки та фреймворки, які надають можливості мінімізації. До таких технологій, які було використано в цій роботі можна віднести препроцесор scss, пакет gulp, Node.js, усі плагіни для gulp, шрифт font awesome 5 та бібліотека bootstrap 5.

Стек технологій, які було використано при розробці:

- VSCode
- Google Chrome
- HTML
- SCSS
- CSS
- Browser-Sync
- Gulp
- Gulp-autoprefixer
- Gulp-clean-css
- Gulp-htmlmin
- Gulp-imagemin
- Gulp-sass
- Gulp-rename
- Node.js
- GitHub
- Git
- JavaScript
- jQuery
- XMLHttpRequest
- Font Awesome 5
- Bootstrap 5

2.1VSCode

Рис. 2.1 логотип VSCode

Visual Studio Code – ide яку було обрано для написання сервісу. Вона була обрана мною через ряд переваг у написанні програмного коду для розробки веб додатків.

Перший плюс цієї IDE – це простота налаштування. Вона має дуже гнучкі налаштування в плані стилістики коду, в ній є вбудований функціонал для того, щоб задати певні правила по написанню коду і вони будуть відслідковуватись автоматично. Наприклад переноси на нову строку великих текстів чи ієрархія html елементів має бути виділена через 4 пробіли. Після налаштувань все працюватиме автоматично та завжди правильно.

До наступної переваги можна віднести кількість плагінів та розширень доступних у вільному доступі. Переходячи на сторінку із розширеннями там можна знайти неймовірну кількість допоміжних розширень та плагінів для будь якої мови програмування чи розширення файлу. Для прикладу наведу розширення, які були використані мною:

- Auto Rename Tag – для автоматичного перейменування частини HTML тега.
- Auto Close Tag – для автоматичного закриття HTML тега.
- Live Server – для швидкого запуску локального сервера із проектом. Як він працює розглянемо детальніше в окремій темі.

Та ще багато інших. А найголовніше, що для їх встановлення не потрібно нічого робити окрім, як натиснути одну кнопку «Встановити» і за рідким виключенням перезавантажити саму IDE.

Третім плюсом може слугувати його невибагливість до платформи. Він доступний на Linux, MacOS та Windows. Тобто його можна використовувати на будь-якому комп'ютері, якщо на ньому встановлена одна із перелічених операційних систем.

Наступним плюсом є вбудована інтеграція Git. Visual Studio Code робить ще один крок уперед, надаючи повну інтеграцію Git, що дозволяє програмістам миттєво бачити зміни, не виходячи із редактора. Значок Git зліва від бічної панелі, де можна викликати його та виконати кілька команд Git, таких як pull, push, publish та інші. Крім того, VSCode також працює з кількома репозиторіями Git, чи то локальними, чи віддаленими.

Неабиякою перевагою є палітра команд. Натискання Ctrl / Command + Shipt + P викликає панель команд, яка робить код VS доступним з клавіатури. Він дозволяє отримати доступ до всіх функцій VS Code, включаючи всі ярлики ключових слів. Крім того, ця палітра також дозволяє отримати доступ до багатьох команд.

Наступний плюс – це функції керування кодом. Visual Studio Code також надає функції управління кодом, такі як Go to Definition, Peek Definition, Find all References і rename Symbol. Клацнувши правою кнопкою миші у файлі коду, можна легко знайти ці функції в VSC.

Гнучкість налаштувань. Як і будь-який інший популярний редактор Visual Studio Code також забезпечує налаштування. Насправді, він забезпечує екстремальне налаштування завдяки своєму гнучкому налаштуванню переваг і безлічі розширень. VSC дає вам можливість змінити тему, змінити сполучення клавіш, налаштувати налаштування, створити фрагменти коду та багато іншого.

2.2 Google Chrome

Рис.2.2 Google Chrome

Велику роль в розробці веб-додатку відіграє браузер в якому проводиться огляд результату написаного коду. Вони різняться функціоналом та підтримкою технологій. Найбільш поширеним браузером серед розробників є Google Chrome. Він є найбільш функціональним та має найкращу підтримку технологій. Після виходу якоїсь нової технології через декілька тижнів, а інколи і днів Google оновлює браузер і вводить підтримку цієї технології. Прикладом може бути технологія flex для CSS. На зображенні номер 3 видно які браузери підтримують її та у якому обсязі.

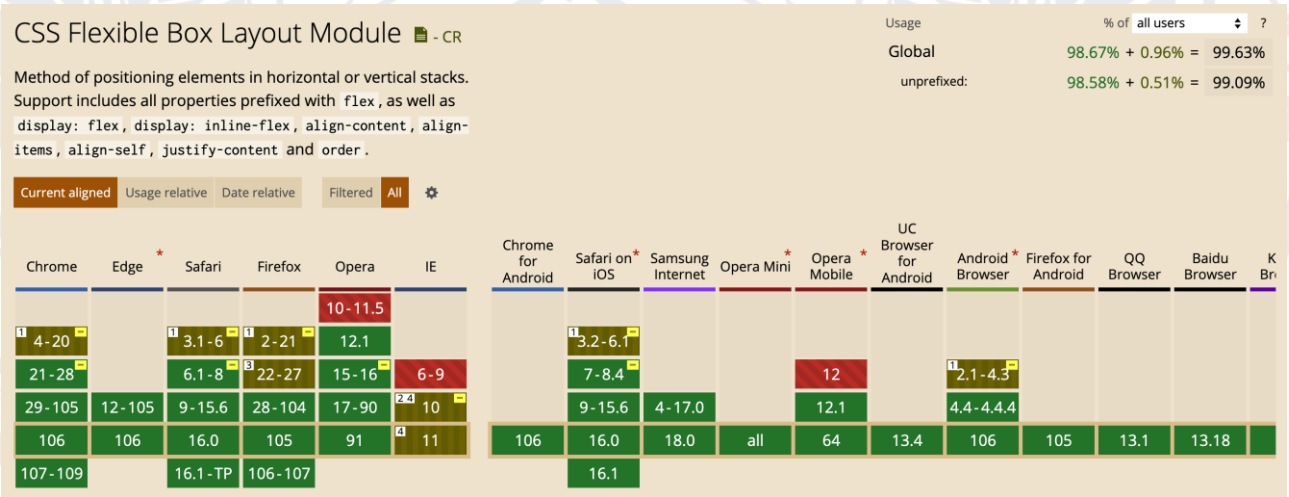

Рис. 2.3 Підтримка технологій

Також у браузерів є свій вбудований функціонал, який дозволяє відстежувати потрібні показники та значення. Такими функціями в Google Chrome: Network, Console, Elements, Sources та інші. Розглянем усі по черзі.

Elements

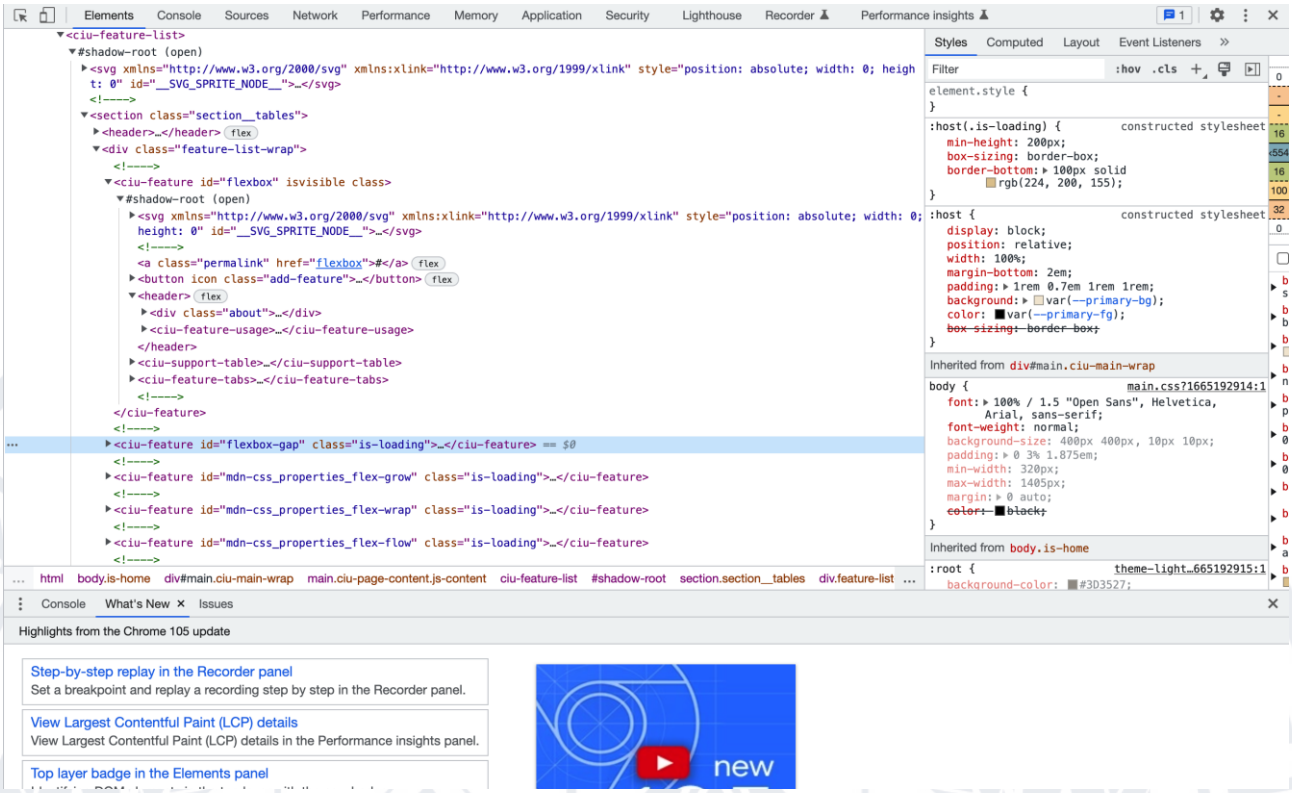

Рис. 2.4 сторінка Elements

З її допомогою можна відстежувати елементи та їх властивості на сторінці, можна редагувати стилі та перевіряти верстку на переповнення. Відстежувати елементи можна декількома способами. Першим способом є пошук в самому інспекторі коду по структурі. Другий спосіб – це пошук візуальний, для цього потрібно натиснути на іконку комп'ютерної миші та навести на елемент на самій сторінці сайту.

Також в цій вкладці можливе редагування коду прямо в браузері. Редагувати можна, як стилі так і структуру сторінки. Після того, як змінити, які-небудь параметри чи структуру, зміни будуть одразу відображені на сторінці.

Console

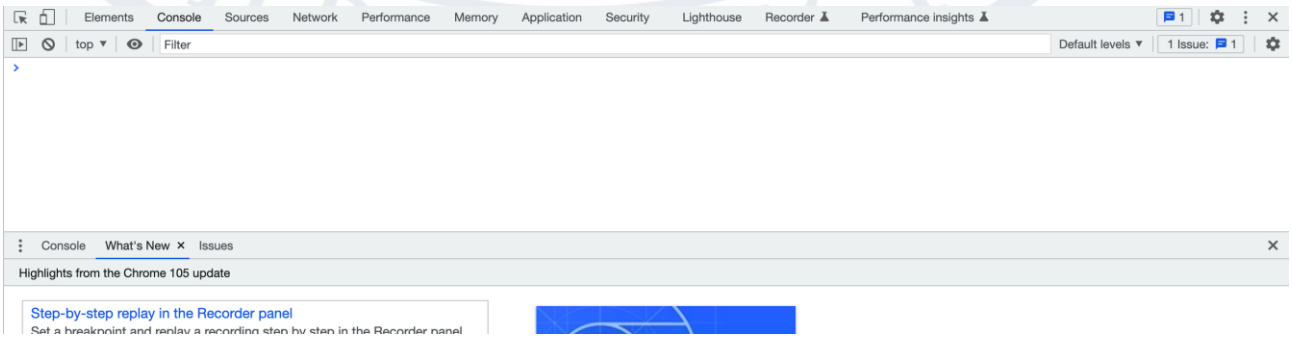

Рис. 2.5 сторінка Console

Дана вкладка в інструментах розробника Google Chrome дозволяє прописувати код та звертатись то javascript частини проекту. Також можна писати новий код та використовувати його на сторінці.

Network

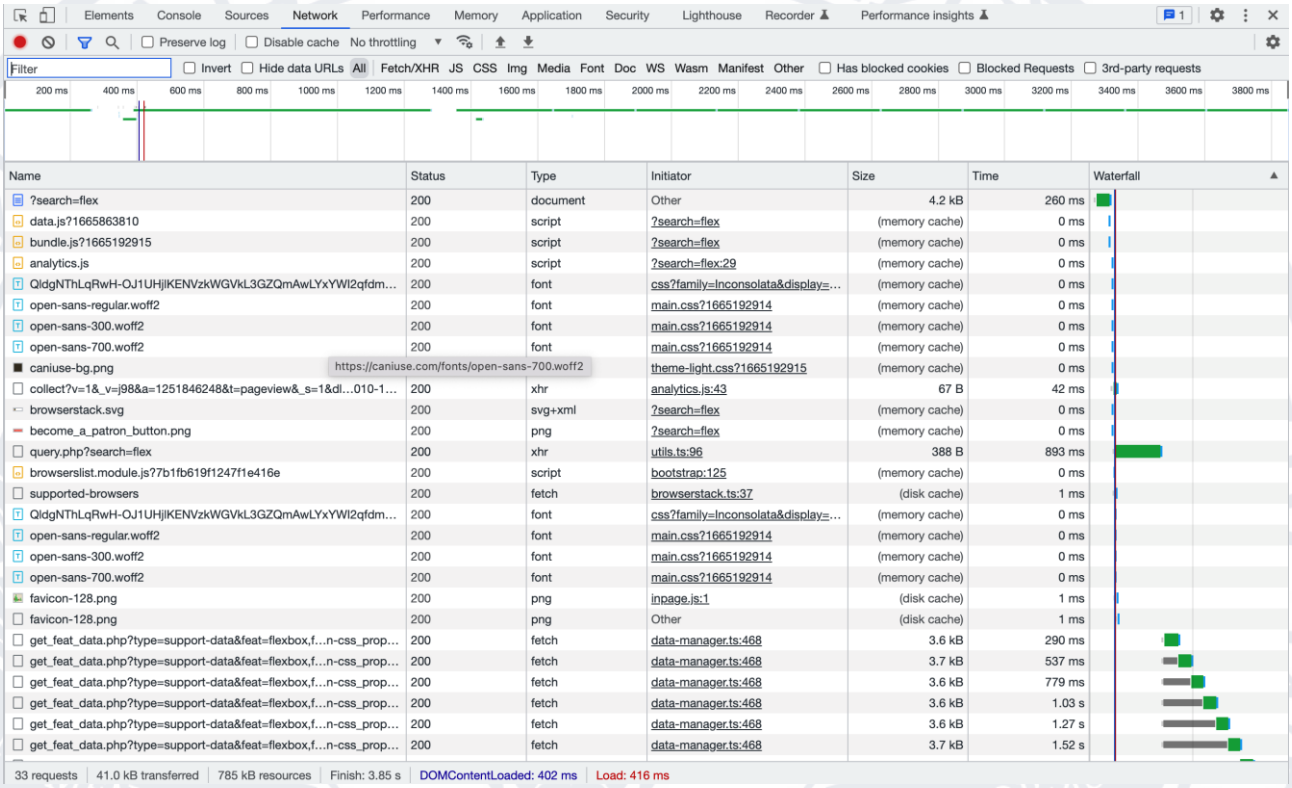

Рис. 2.6 сторінка Network

Кожен рядок у журналі Network – це ресурс. Усі вони перераховані у хронологічному порядку (за умовчанням). Верхній зазвичай є основним HTML-документом. Нижній — те, що запросили останнім.

Кожен стовпець – інформація про ресурс. Основні дані за замовчуванням:

Status - код відповіді;

Type – тип ресурсу;

Initiator – що викликало запит ресурсу. Клацнувши посилання в стовпці Initiator, виконається перехід до вихідного коду, що викликав запит;

Size – розмір ресурсу;

Time - як довго тривав запит;

Waterfall – графічне представлення різних етапів запиту.

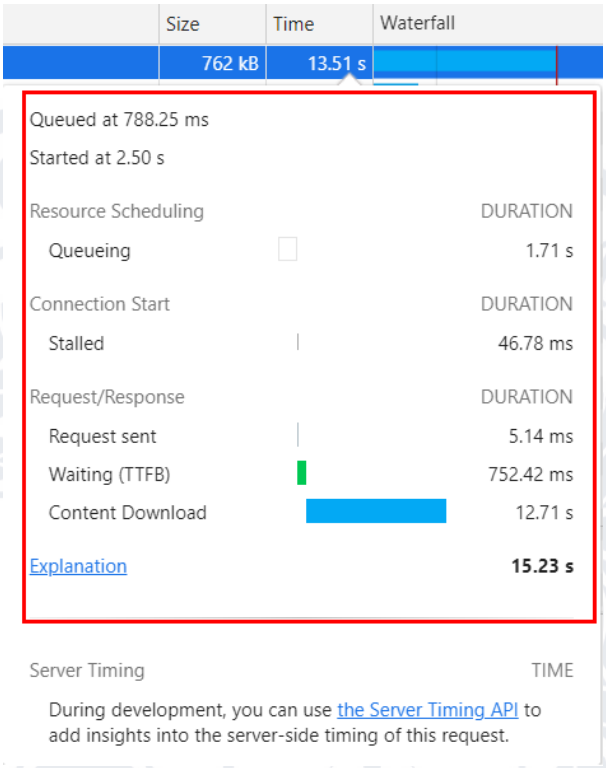

Рис. 2.7 Етапи завантаження

Функціонал вкладки Network:

1. Стовпці журналу можна настроювати. Є багато додаткових стовпців з корисною інформацією, які можна приховати.

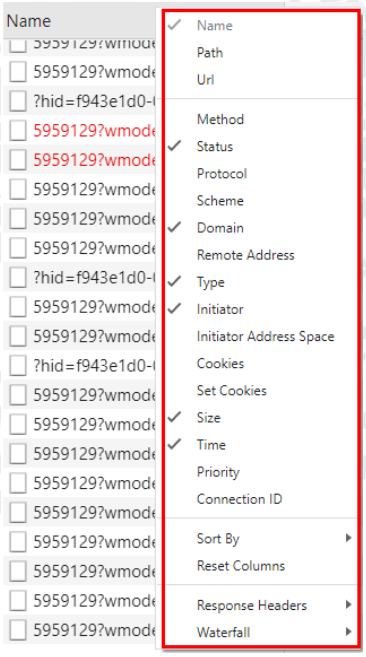

Рис. 2.8 Стовпці Network

2. Є можливість фільтрувати дані в журналі за допомогою панелі інструментів Filter. Інструмент Filter підтримує багато різних типів фільтрації. Можна використовувати не лише окремі слова, а й регулярні вирази та властивості.

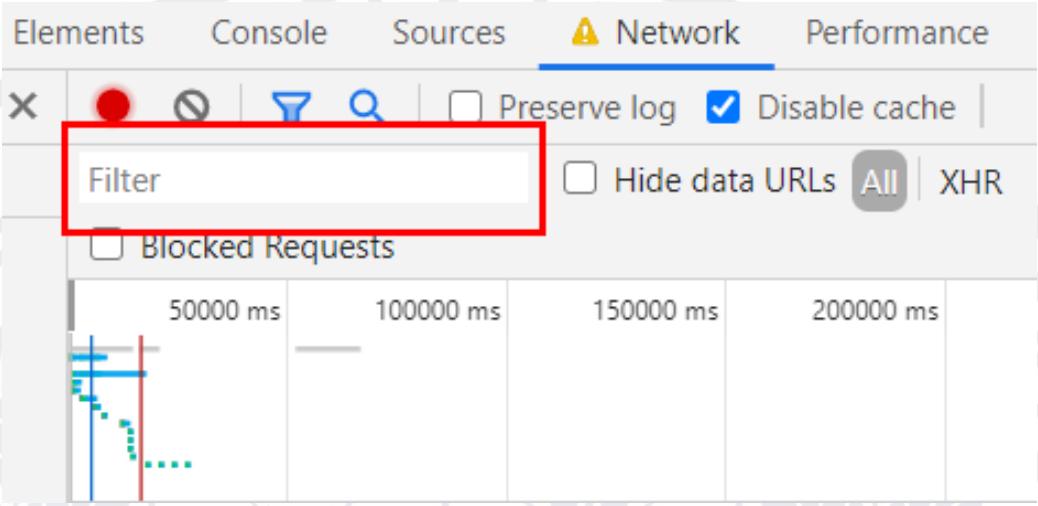

Рис. 2.9 фільтр Network

3. Крім того, є можливість фільтрувати дані типу ресурсу, використовуючи на панелі потрібний функціонал відбору.

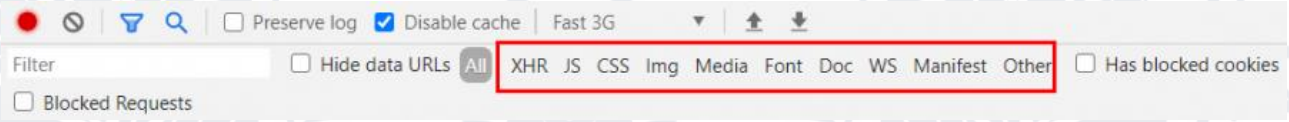

Рис. 2.10 Дані типу ресурсу

4. Поки відкрито Developers Tools, він буде записувати мережну активність до журналу.

5. Зазвичай мережне підключення комп'ютера швидше, ніж у мобільних пристроїв користувачів. Меню Throttling дозволяє регулювати швидкість підключення, щоб зрозуміти скільки часу потрібно для завантаження сторінки на мобільному пристрої.

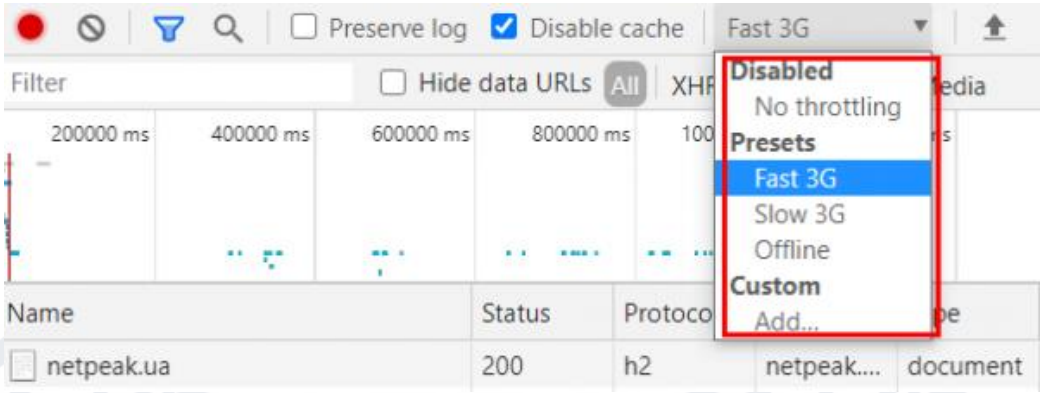

Рис. 2.11 список Throttling

6. При повторних відвідинах браузер часто використовує деякі файли зі свого кеша, що прискорює завантаження. Якщо хочете побачити, як відвідувач сприймає завантаження сторінки вперше, увімкніть Disable Cache.

7. Для збереження даних у журналі необхідно увімкнути Preserv log.

8. Якщо натиснути на шестерню, на панелі з'являться додаткові функції:

Group by frame – розбиває ресурси на чіткі групи залежно від домену чи типу;

Capture screenshots – дозволяє робити скріншоти сторінки під час її завантаження;

Use large request rows - додає додаткову інформацію про файли до таблиці;

Show overview – дозволяє приховувати та показувати графічну інформацію про завантаження сторінки.

Усі ці допоміжні функції неймовірно прискорюють процес розробки веб-додатків та сайтів та полегшують його.

2.3 HTML

HTML – це мова гіпертекстової розмітки. За допомогою цієї мови будується каркас сторінки. Для повного розуміння, що це і як воно виглядає зверніть увагу на рис.11.

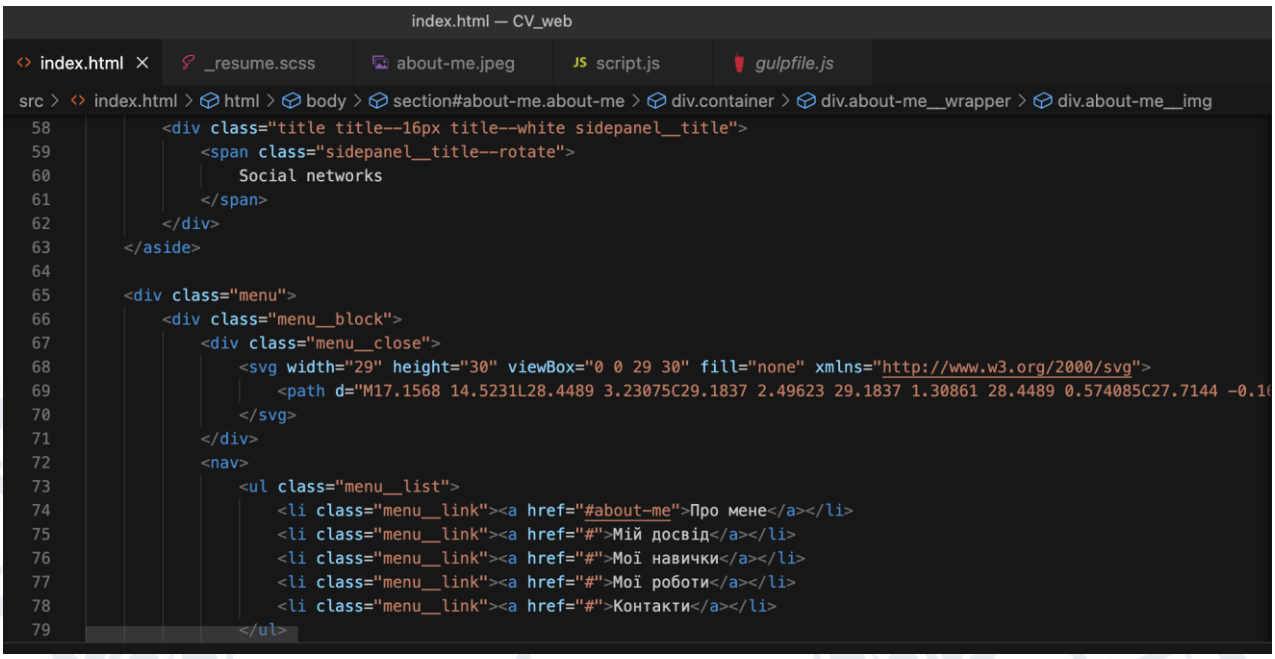

Рис. 2.12 HTML код

Скелет чи каркас сторінки будується за допомогою HTML тегів. HTML теги використовуються для позначення будь-якого елементу відповідним чином на сторінці веб-додатку.

В HTML будування розмітки базується на тегах. Кожен тег відповідає за якісь відповідний тип елементу. Наприклад тегом <img> позначаються зображення, тегом позначається який-небудь блок з контентом. Найбільш зустрічаючий тег – це . Його використовують багатьма ідеологіями, такими, як обгортка для певного блоку на сторінці, для виділення чи розбиття контенту на різні блоки та інше.

Так же потрібно пам’ятати, що у кожного тегу є можливі атрибути. Вони діляться на два типи: обов’язкові і необов’язкові. До обов’язкових можна віднести href та src атрибути. Вони відповідають за адресу посилання та шлях до файлу відповідно. Наприклад кожна картинка, яка завантажується через тег <img> повинна мати атрибут src із шляхом в якій саме папці лежить ця картинка та кожне посилання має обов’язковий атрибут href із повним посиланням.

HTML надає також можливість створювати форми та налаштовувати їх і проводити перевірку без взаємодії із серверною частиною додатку. Для того,

щоб створити власну форму існує спеціальний тег <form> який позначає початок форми. Усі форми мають спеціальні атрибути:

- accept - список типів вмісту, розділених комою, які приймає сервер.
- accept-charset - розділені пробілами символьні кодування, які приймає сервер. Браузер використовує їх у тому порядку, в якому вони перераховані. Значення за промовчанням означає те ж кодування, що й у сторінки. (У попередній версії HTML, різні кодування могли бути розділені комами.)
- action - URI-адреса програми, яка обробляє інформацію, передану через форму. Це значення може бути переписано за допомогою атрибута formaction на <button> або <input> елементі.
- autocomplete - Вказує, чи елементи керування можуть бути автоматично дописані у формі браузером. Ця установка може бути переписана за допомогою атрибута autocomplete на елементі форми. Можливі значення:
    - off: Користувач повинен явно ввести значення у кожне поле або документ, що надає свій власний метод автодоповнення; браузер автоматично не доповнює запис.
    - on: Браузер може автоматично доповнити значення, що базуються на значеннях, які користувач вже вводив, протягом попереднього використання форми.
    - Якщо встановити значення off для autocomplete атрибуту форми, тому що документ надає своє власне автодоповнення, то вам слід також встановити значення off для autocomplete кожного <input> елемента форми, який документ може автоматично доповнити.
- enctype - Коли значення атрибута method дорівнює post, атрибут - MIME тип вмісту, який використовується для передачі форми на сервер. Можливі значення:
    - application/x-www-form-urlencoded: Значення за промовчанням, якщо атрибут не встановлено.

    - multipart/form-data: Використовуйте це значення, якщо користуєтеся елементом <input> атрибутом type встановленим у "file".
    - text/plain (HTML5)Це значення може бути переписане атрибутом formtype на елементі <button> або <input>.
- method - HTTP (en-US) метод, який використовує браузер, для відправки форми.

  Можливі значення:
    - post: Відповідає HTTP POST методу; дані форми включаються в тіло форми і посилаються на сервер.
    - get: Відповідає GET методу; дані форми додаються до URI атрибута action, їх розділяє '?', і отриманий URI посилається на сервер. Використовуйте цей метод, коли форма містить лише ASCII символи і не має побічного ефекту. Це значення може бути переписано атрибутом formmethod на <button> або <input> елементі.
- name - Назва форми. У HTML 4 його використання заборонено (id слід використовувати натомість). Воно має бути унікальним і не порожнім серед усіх форм у документі HTML 5.
- novalidate - Це Boolean атрибут показує, що форма не перевіряється на валідність, коли надсилається серверу. Якщо атрибут пропущений (і тому форма перевіряється), ця установка за умовчанням, може бути переписана атрибутом formnovalidate на <button> або <input> елементі, що належить формі.
- target - Ім'я або ключове слово, що показує де відображати відповідь, яку буде отримано після відправки форми. У HTML 4 це ім'я або ключове слово для кадру. У HTML5 це ім'я або ключове слово, контексту перегляду (наприклад, вкладка, вікно, або лінійний кадр). Наступні ключові слова мають особливе значення:
    - _self: Завантажує відповідь у тому самому кадрі HTML 4 (або HTML5 контексті перегляду) як поточний. Це значення за промовчанням, якщо атрибут не вказано.

- _blank: Завантажує відповідь у новому безіменному вікні HTML 4 або HTML5 контексті перегляду.
- _parent: Завантажує відповідь HTML 4 у батьківському наборі кадру для поточного кадру або HTML5 батьківський контекст перегляду для поточного перегляду. Якщо немає батьків, ця опція діє так само як as _self.
- _top: HTML 4: Завантажує відповідь у повне, оригінальне вікно, закриваючи всі інші кадри. HTML5: Завантажує відповідь у верхній рівень контексту відтворення (тобто, контекст відтворення це предок поточного і не має інших предків). Якщо немає батьків, ця опція діє так само, як as _self.HTML5: Це значення може бути перезаписано formtarget атрибутом на <button> або <input> елементі.

Приклад форми:

```
<!-- Проста форма, яка надішле GET запит -->
<form action="">
  <label for="GET-name">Name:</label>
  <input id="GET-name" type="text" name="name">
  <input type="submit" value="Save">
</form>

<!-- Проста форма, яка надішле POST запит -->
<form action="" method="post">
  <label for="POST-name">Name:</label>
  <input id="POST-name" type="text" name="name">
  <input type="submit" value="Save">
</form>

<!-- Форма з fieldset, legend, та label -->
<form action="" method="post">
  <fieldset>
    <legend>Title</legend>
    <input type="radio" name="radio" id="radio"> <label for="radio">Click me</label>
  </fieldset>
</form>
```

2.4 CSS

CSS – дослівно перекладається з англійської (cascading style sheets), як каскадні таблиці стилів. Насправді це так і є. Ця технологія дозволяє оформляти та стилізувати веб-сторінки як завгодно.

Методи підключення CSS до документа.

Правила CSS можуть розташовуватися як у самому веб-документі, зовнішній вигляд якого вони описують, так і зовнішніх файлах, що мають розширення .css. Формат CSS - це текстовий файл, в якому міститься перелік правил CSS та коментарів до них.

Стилі CSS можуть бути підключені або впроваджені в описуваний ними веб-документ чотирма способами:

1. коли опис стилів знаходиться в окремому файлі, він може бути підключений до документа за допомогою елемента <link>, включеного до елемента <head>

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    .....
  </body>
</html>
```

2. коли файл стилів розміщується окремо від батьківського документа, він може бути підключений до документа інструкцією @import в елементі <style>

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <style media="all">
      @import url(style.css);
    </style>
  </head>
</html>
```

3. коли стилі описані всередині документа, вони можуть бути включені в елемент <style>, який включається в елемент <head>

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <style>
      body {
```

```
        color: red;
      }
    </style>
  </head>
  <body>
    .....
  </body>
</html>
```

4. коли стилі описані в тілі документа, вони можуть розташовуватися в атрибутах окремого елемента

```
<!DOCTYPE>
<html>
  <head>
    .....
  </head>
  <body>
    <p style="font-size: 20px; color: green; font-family: arial, helvetica, sans-serif">
      .....
    </p>
  </body>
</html>
```

Правила побудови CSS

У перших трьох випадках підключення стилів CSS до документа, кожне правило CSS з файлу має дві основні частини - селектор і блок оголошень. Селектор, розташований у лівій частині правила до значка «{», визначає, які частини документа (можливо, спеціально позначені) поширюється правило. Блок оголошень розміщується у правій частині правила. Він міститься у фігурні дужки, і, своєю чергою, складається з однієї чи більше оголошень, розділених знаком «;». Кожне оголошення є поєднанням властивості CSS і значення, розділених знаком «:». Селектори можуть групуватися в одному рядку через кому. У разі властивість застосовується до кожного їх.

```
Селектор {
   Властивість: значення;
   Властивість: значення;
   Властивість: значення;
   Властивість: значення;
}
```

Універсальний селектор. Зірочка (*) – універсальний селектор для CSS. Відповідає будь-якому тегу. Забираючи зірочки із простих селекторів має той самий ефект. Наприклад, * .warning та .warning вважаються рівними.

У CSS 3, зірочка (*) може використовуватися в комбінації з простором імен:

- ns|* - входження всіх елементів у просторі імен ns
- *|* - знаходить всі елементи
- |* - шукає всі елементи без оголошеного простору імен

Приклад:

```
*[lang^=en]{color:green;}
*.warning {color:red;}
*#maincontent {border: 1px solid blue;}
```

Селектор тегів. Як селектор може виступати будь-який HTML тег, для якого визначаються правила форматування, такі як: колір, фон, розмір та інші. Правила задаються в наступному вигляді.

```
p {
  font-family: arial, helvetica, sans-serif;
}
```

Спочатку вказується ім'я тегу, оформлення якого буде перевизначено, великими чи малими символами немає значення. Усередині фігурних дужок пишеться стильова властивість, а після двокрапки - його значення. Набір властивостей розділяється між собою крапкою з комою і може розташовуватися як в один рядок, так і кілька.

Селектор класів. У HTML документі, селектори CSS класу знаходять елементи з потрібним класом. Атрибут класу визначається як розділений пробілами список елементів, і один із цих пунктів повинен точно відповідати імені класу, наведеному у селекторі.

```
.class-name {
  font-family: arial, helvetica, sans-serif;
```

```
}
```

Селектор ідентифікаторів. Селектор ідентифікатора аналогічний до селектора класу, але перед ім'ям ставиться не точка, а знак “решітки” (#).

```
#id-name {
  font-family: arial, helvetica, sans-serif;
}
```

Селектор атрибутів. Селектори атрибутів відбирають елементи наявності атрибута або його значення.

[attr] - Позначає елемент з атрибутом на ім'я attr.

[attr=value] - Позначає елемент з ім'ям атрибута attr і значення точно збігається з value.

[attr~=value] - Позначає елемент з ім'ям атрибута attr значенням якого є набір слів розділених пробілами, одне з яких точно дорівнює value.

[attr | = value] - Позначає елемент з іменем attr атрибута. Його значення при цьому може бути або точно одно "value" або може починатися з "value" з відразу ж наступним "-" (U+002D). Це може бути використане, коли мова описується з підходом.

[attr^=value] - Позначає елемент з ім'ям атрибута attr, значення якого починається з "value".

[attr$=value] - Позначає елемент з ім'ям атрибута attr, чиє значення закінчується на "value".

[attr*=value] - Позначає елемент з ім'ям атрибута attr, чиє значення містить принаймні одне входження рядка "value" як під рядка.

Селектор псевдо класів.

```
a:active {
  color: blue;
}
```

Селектор псевдо елементів.

```
p::first-letter {
  font-size: 32px;
}
```

Вище було розібрано майже всі можливі записи оголошення стилів та всі способи підключення CSS до сторінки. Властивостей в CSS незлічена множина і розглядати їх на прикладах дуже довго та не потрібно адже все дуже інтуїтивно зрозуміло, але є ще тема яку слід було б розглянути і це CSS Variables.

CSS Variables.

Це змінні, які іноді дуже полегшують життя для розробника. Цей інструмент надає можливість оголошувати деякі змінні в CSS та потім їх використовувати необмежену кількість разів. Слід пам’ятати про те, що є декілька правил яким треба слідувати для того, щоб все правильно працювало. Найважливішим із цих правил є те, що змінні які оголошені на рівня якогось блока, можуть використовуватись тільки на рівні цього блока або ж в середині нього.

Для прикладу розглянемо наступний код:

HTML

```
<div>
   <div class="one"></div>
   <div class="two">Text <span class="five">- more text</span></div>
   <input class="three">
   <textarea class="four">Lorem Ipsum</textarea>
</div>
```

CSS

```
:root {
  --main-bg-color: brown;
}

.one {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 50px;
  height: 50px;
  display: inline-block;
}

.two {
```

```
  color: white;
  background-color: black;
  margin: 10px;
  width: 150px;
  height: 70px;
  display: inline-block;
 }
.three {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 75px;
 }
.four {
  color: white;
  background-color: var(--main-bg-color);
  margin: 10px;
  width: 100px;
 }

.five {
  background-color: var(--main-bg-color);
 }
```

Наслідування змінних

Змінні мають властивість унаслідуватись. Це означає, що якщо не встановлено ніяких значень для змінної в блоці то значення змінної буде унаслідуватись від батьківського елемента.

Приклад:

HTML

```
<div class="one">
  <div class="two">
    <div class="three"></div>
    <div class="four"></div>
  </div>
</div>
```

CSS

```
.two {
  --test: 10px;
}

.three {
  --test: 2em;
 }
```

Результат:

- two – 10px
- three – 2em
- four – 10px (унаслідувалось від батьківського)

На сьогоднішній день, цей інструмент є предметом першої необхідності для розробника.

2.5 Препроцесори

Препроцесор – це програма, яка отримує якісь дані на вході, обробляє їх та видає оброблені дані виході. Тобто простими словами препроцесори Saas/SCSS та LESS мають свій унікальний синтаксис для написання стилів та після обробки написаних стилів на мові препроцесора вони віддають стилі написані на синтаксисі CSS.

Написання таких стилів через препроцесор робиться в файлі із відповідним розширенням. Наприклад .scss , .saas , .less та інші. Далі потрібно конвертувати цей файл та перенести стилі у файл .css розширення. Для цього існує дуже багато різних програм та плагінів. Найпопулярніший плагін - це SASS для VSCode.

Переваги препроцесорів:

- Першим та найвагомішим плюсом препроцесорів є вкладеність. Можливість писати стилі вкладаючи селектори друг в друга дуже сильно скорочує написання коду для стилів.
- Легкість структурування коду в порівняні із CSS.
- Маленькі зусилля для внесення великих змін.

Особливості препроцесорів:

- Змінні – Вони дуже відрізняються від змінних CSS. Це змінні, які можна оголосити на початку файлу та використовувати будь-де в цьому файлі для будь-якого селектора та рівня вкладеності. Це дуже зручно тим, що якщо вам потрібно ввести якісь зміни в стилі проекту, можна замінити значення в одному місці та зміни будуть запровадженні у всіх місцях де ця змінна була використана.
- Вкладеність – На звичайному CSS не має можливості вкладати селектор в селектор, що приводить до збільшення написання коду, поганого освоєння та розуміння коду та поганій структуризації коду.

- Mixins – Це такі стильові функції, які дозволяють згрупувати декілька стильових властивостей або ж навіть дозволяють написання шаблонів для великих елементів та потім використовувати цей код у багатьох місцях.
- Імпорт – Якщо коротко, то це розбиття на файли та імпорт стилів з одного файлу в інші. Дуже корисна функція, яка спрощує підтримку стилів так, як набагато легше підтримувати добре структурований проект розбитий на окремі файли ніж розбиратись із одним файлом на десятки тисяч строк коду. Так само хороша практика створювати окремі файли для змінних, міксинів, шрифтових стилів тощо.

Препроцесор неймовірно полегшує життя для розробників та економить час на написанні стилів для проекту.

2.6 Node.js

Node.js – це програмна платформа, яка перетворює JavaScript з вузькоспеціалізованої мови програмування в мову загального призначення. Вона надає можливість JavaScript звертатись до пристроїв на вхід та вихід за допомогою API, встановлювати різноманітні сторонні пакети від інших розробників.

2.7 JavaScript

JavaScript – динамічна об’єктно-орієнтована мова програмування, яка дозволяє взаємодіяти з клієнтом на боці самого ж клієнта без додаткових звертань до серверної частини проекту. JavaScript відносять до мов програмування із динамічною типізацією.

Синтаксис мови дуже схожий із синтаксисом мови програмування C, але є відмінності:

- об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів
- функції як об'єкти першого класу

- обробка винятків
- автоматичне приведення типів
- автоматичне збирання сміття
- анонімні та стрілочні функції

Для використання цієї мови в парі із якоюсь сторінкою HTML не обов'язковий навіть окремий файл. Це можливо зробити написавши Javascript код прямо в HTML файлі у спеціальному тегові <script>. Також є більш правильний спосіб для використання js у проекті. Це підключення через тег <link>. Для цього потрібно у тег <link> передати атрибут src із вказаним місцезнаходження потрібного js файлу.

У мові JS існує поняття асинхронності та асинхронних скриптів. Для того, щоб вказати що скрипт буде асинхронний є два атрибути async та defer. Будь-який «скріпт», який позначений одним із цих двох атрибутів буде відтворюватись асинхронно, але є певні відмінності в цих атрибутів.

Перша з них говорить про те, що відносний порядок скриптів із атрибутом defer буде збережено і кожен «скріпт» буде виконуватись по черзі, а із атрибутом async скрипти будуть відтворюватись по мірі їх отримання браузером.

Друга відмінність полягає в тому, що скрипти із атрибутом defer почнуть виконуватись лише тоді коли html сторінка буде повністю завантажена.

Далі розглянемо типізацію цієї мови програмування. Оскільки мова програмування JS має динамічну типізацію, тому потрібно бути дуже уважним та перевіряти результати роботи коду у всіх можливих варіантах. Динамічна типізацію означає те, що в ході виконання, якогось коду число може перетворитися в стрічку чи щось в цьому ключі.

Налагодження скриптів це також дуже важлива стадія розробки і її ніяк не можна обійти. Для цього існує перелік додатків та програм, таких як: ESLint, Babel, Prettiler та інші. Ці програми перевіряють правильність написання коду та у випадку якихось проблем виведуть повідомлення про те де і яка проблема сталась. Також вони надають можливість задавати свої правила з написання коду і коли на проекті працює дуже багато людей із своїм так званим почерком, то це

вирішує проблему і зводить все до того, що всі програмісти пишуть код майже однаково. Це надає більше зрозумілості та структурованості коду.

2.8 Gulp

Gulp – це застосунок для мови програмування javascript, який виконує функцію менеджера задач для задач мінімізації, об’єднання, розбиття та тестування. Всі задачі описуються на мові програмування javascript та використовують тільки доступні плагіни, які можна встановити через менеджер пакетів npm.

Плагіни для Gulp можуть розроблятися, як великими компаніями так і простими людьми та викладатися на GitHub. Для того, щоб в подальшому використовувати плагін треба спочатку його встановити через менеджер пакетів npm та команду install.

Приклад встановлення плагіну:

```
npm install sass
```

Далі потрібно підключити цей плагін на початку gulp.js файлу ось таким чином.

```
//Adding dependencies
var gulp = require ( 'gulp');
var sass = require ( 'sass');
```

Завдання для gulp.js методу .task зі зверненням до об’єкту gulp. Першим аргументом цього методу є назва плагіна який буде використовуватись, другим параметром виступає анонімна функція в якій описана логіка задачі. Є можливість написати багатофункціональну задачу. Для цього потрібно другим аргументом передавати масив функцій.

Плагіни для Gulp, які було використано для виконання роботи:

- Browser-Sync – цей інструмент дозволяє відстежувати зміни у вихідних файлах та скриптах без перезавантаження веб-сторінки.

- Gulp-autoprefixer – плагін, який використовується для керування браузерними префіксами по проекту.
- Gulp-clean-css – плагін, який використовується для мінімізації css стилів.
- Gulp-htmlmin – плагін, який використовується для мінімізації html коду сторінки.
- Gulp-imagemin - плагін, який використовується для мінімізації та пере тискання зображень.
- Gulp-sass - плагін, який використовується для перетворення коду написаному на препроцесорах sass та scss у стилі написані на звичайному css.
- Gulp-rename - плагін, який використовується для перейменування файлів.

2.9 JQuery

Це бібліотека, яка надає набір функцій для мови програмування javascript для “спілкування” із HTML. Під “спілкуванням” мається на увазі полегшення отримання елементів DOM структури до javascript, звернення до атрибутів та вмісту DOM елементів та маніпулювати ними. Так же JQuery надає можливість через API взаємодіяти із AJAX.

Можливості:

- Двигун крос-браузерних CSS-селекторів Sizzle
- Перехід по дереву DOM, включаючи підтримку XPath як плагін
- Події
- Візуальні ефекти
- AJAX-доповнення
- JavaScript-плагіни

Роботу із цією бібліотекою можна поділити на 2 етапи. Перший етап – це отримання JQuery об’єкта. Другий етап – це виклики глобальних методів у JQuery об’єкта.

Приклади:

```
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
```

```
$.each([1,2,3], function() {
  document.write(this + 1);
});
```

2.10XmlHttpRequest

Технологія XmlHttpRequests – це вбудований в кожен браузер об’єкт, який надає можливість надсилати http запити до серверу без перезавантаження сторінки.

У назві присутня приставка Xml та це може ввести в оману, що цей об’єкт працює лише з даними типу Xml, але це не так. Дана технологія може працювати із будь-яким типом даних. Можна завантажувати та вивантажувати файли, відстежувати прогрес та багато іншого.

На сьогоднішній день не обов'язково використовувати XMLHttpRequest, оскільки існує інший, сучасніший метод fetch.

У сучасній веб-розробці XMLHttpRequest використовується з трьох причин:

- Підтримка вже існуючого коду
- Необхідність підтримувати старі браузери
- Необхідність у функціоналі, який поки, що недоступний для технології fetch. Наприклад, стеження за прогресом відправки на сервер.

Основи роботи із технологією. Існує 2 режими роботи із цією технологією: синхронний та асинхронний. Спершу розглянемо асинхронний режим так, як він найбільш популярний.

Для того, щоб надіслати запит потрібно виконати наступні кроки:

- Створити XmlHttpRequest

  `let xhr = new XMLHttpRequest();`
- Ініціалізувати його

  `xhr.open(method, URL, [async, user, password])`

- Надіслати запит

  `xhr.send([body])`

  Цей метод встановлює з'єднання та надсилає запит до сервера.

  Необов'язковий параметр body містить тіло запиту. Деякі типи запитів, як-от GET, не мають тіла. А деякі, як, наприклад, POST, використовують body, щоб надсилати дані на сервер.
- Прослуховувати події на xhr, для отримання відповіді

  Три найбільш використовувані події:
  - load – відбувається, коли отримано будь-яку відповідь, включаючи відповіді з помилкою HTTP, наприклад 404.
  - error – коли запит не може бути виконаний, наприклад, немає з'єднання або URL.
  - progress – відбувається періодично під час завантаження відповіді, повідомляє прогрес.

Після відповіді сервера можна отримати результат запиту у наступних властивостях xhr:

- status – Код стану. Наприклад 200, 403, 404…
- statusText – повідомлення про стан запиту.
- response – тіло відповіді

Тепер розглянем, що таке синхронні запити. Синхронні запити відрізняються від асинхронних тим, що відпрацювання javascript коду призупиняється на моменті send() та продовжується тільки після отримання відповіді на запит. Для того, щоб зробити запит синхронним, потрібно такий параметр, як async який за замовчуванням має значення true встановити в значення false.

2.11 Система контролю версій Git

Git – розподілена система контролю версій файлів та спільної роботи. Цікавим фактом є те, що ця система була спроектована та розроблена для контролю версій при створенні системного ядра Linux та підтримується до

сьогоднішнього дня. Git система контролю версій є найвідомішою та найбезпечніших в світі, що базується на злитті та розгалужені гілок. Для забезпечення цілісності історії збереження змін та стійкості до змін заднім числом використовуються криптографічні методи. Також можлива прив'язка цифрових підписів розробників до тегів та коммітів.

Цю систему використовують компанії різного калібру, від найменших до найбільших світових корпорацій. Ця система контролю версій надає найбільш стабільну роботу та неймовірну зручність та пристосованість для того, щоб користувачі мали усі необхідні можливості. Прикладами проектів, що використовують Git, є ядро Linux, АБІС Koha, LibreOffice, Eclipse, Android, Cairo, GNU Core Utilities, Mesa 3D, Wine, багато проектів з X.org, XMMS2, Qt, Ruby on Rails, GStreamer, Debian DragonFly BSD, Perl, PostgreSQL, VideoLAN, GNOME, KDE, PHP, One Laptop Per Child (OLPC), GNU LilyPond та ELinks і деякі дистрибутиви GNU/Linux.

Репозиторій (repository)

Репозиторій – це колекція файлів і папок, які використовуються для відстеження git. Це той великий склад, на якому зберігається код, який додали усі програмісти разом. Сховище складається з усієї історії змін вашої команди в проекті.

Github

Найпопулярніше хмарне сховище для git-репозиторіїв. Особливості: підписуватися на повідомлення сховища, він дозволяє вам відстежувати і відправляти помилки, встановлювати права доступу до проектів, приймати запити на поліпшення, використовувати графічний інтерфейс, а не командний рядок. Репозиторій за замовчуванням відкриті, але користувачі можуть мати й приватні репозиторії.

Commit

Думайте про це як про збереження вашої роботи. Коли відбувається фіксація репозиторію, це схоже на збір файлів в тому вигляді, в якому вони існують в даний момент, і поміщаєте їх в капсулу часу. Фіксація буде існувати

тільки на вашому локальному комп'ютері, поки вона не буде відправлена на віддалений репозиторій.

Push

Фіксація поміщає ваші файли в капсулу часу, а відправка – це те, що запускає капсулу в космос. Відправлення – це, по суті, синхронізація ваших збережень (фіксацій, коммітів) з хмарою (знову ж таки, ймовірно, Github). Також є можливість використовувати кілька коммітів одночасно. Можна працювати в автономному режимі, зробити багато роботи, а потім передати все це на Github.

Branch

Уявіть свій git-репо у вигляді дереві. Стовбур дерева, програмне забезпечення, яке запускається, називається майстер-гілкою (Master Branch). Це те, що є онлайн. Гілки цього дерева називаються, як не дивно, гілками. Це окремі екземпляри коду, який відрізняється від основної бази коду. Існує можливість відгалузити одну функцію або експериментальний патч. Розгалужуючись, є можливість зберегти цілісність основного програмного забезпечення і мати можливість відкотитися, якщо зробите щось зовсім божевільне. Це також дозволяє вам працювати над своїм завданням, не впливаючи на вашу команду (або вона на вас).

Merge

Коли гілка виправлена, не містить помилок (наскільки принаймні можна судити) і готова стати частиною первинної бази коду, вона буде об'єднана з головною гілкою. Об'єднання – це те, на що це схоже: злиття двох гілок. Будь-який новий або оновлений код стане офіційною частиною кодової бази. Той, хто відгалужується від точки злиття, також буде мати цей код в своїй гілці.

Clone

Клонування репозиторію – це майже те ж саме, як і звучить. Воно бере весь онлайн-репозиторій і робить точну копію на вашому локальному комп'ютері. Вам потрібно буде зробити це по ряду причин, і не в останню чергу для збереження сумісності.

Fork

«Форкінг» багато в чому схожий на клонування, тільки замість того, щоб робити копію існуючого репозиторію на локальному комп’ютері, буде отримано абсолютно новий репозиторій цього коду під своїм власним ім’ям. Ця функція в основному використовується для взяття існуючої кодової бази і перехід з нею в абсолютно новому напрямку, що часто трапляється в програмному забезпеченні з відкритим вихідним кодом; розробники бачать базову ідею, яка працює, але хочуть піти іншим шляхом. «Форкінг» дозволяє цьому статися. Також є можливість взяти участь в репозиторії іншого розробника, як у своїй особистій пісочниці. І якщо зробити щось, що, на вашу думку, може йому сподобатися, можна зробити попередній запит на об’єднання.

Pull Request

Запит на підтвердження – це коли йде відправка запиту з внесеними змінами (або в гілці, або в відгалуженні), які повинні бути перенесені (або об’єднані) в основну гілку (Master Branch) сховища. Це великий час, і тут відбувається диво. Якщо запит на підтвердження буде схвалений, тоді офіційно буде внесений внесок в програмне забезпечення, і Github завжди буде показувати, хто саме, що зробив. Однак, якщо запит відхиляється з якої-небудь причини, ревізор зможе дати відгук про те, чому запит був відхилений і що можна зробити, щоб він був прийнятий.

В даній роботі систему контролю версій використано для контролювання процесу розробки мобільного додатку та зберігання резервної копії програмного коду у хмарі так як це найпотужніший сервіс контролю версій на сьогоднішній день.

2.12 Font Awesome 5

Шрифт – це шрифт, який підключається до проекту, як звичайний шрифт, але його стилі написані так, що певний клас, відповідає за певну іконку. Чим це зручно? По-перше економія трафіку. Завантаження шрифту браузером займає набагато менше часу ніж завантажувати кожен раз одні й ті самі svg чи png картинки. Для прикладу візьмем ситуацію коли одна іконка зустрічається на

одній сторінці 10 разів. Для того, щоб її відобразити через шрифти потрібно завантажити 1 раз файл шрифтів і все, але для того щоб відобразити її через png, вона має бути 10 разів описана в структурі HTML та має бути завантажена 10 разів. По-друге це легкість налаштування. Мається на увазі те, що на ці іконки дуже просто впливати через css.

Шрифти це невід'ємна частина стилістика веб-додатку. Цей сервіс надає вже готові рішення із наборами іконок, як платними так і з безкоштовними. Він має декілька варіантів підключення його до проекту.

Перший спосіб підключення через CDN сервер. Для цього потрібно всього лиш скопіювати три теги link в документації із посиланнями на сервери CDN та вставити в head свого проекту. Він підкупає своєю простотою, але є декілька нюансів. Для початку потрібно розуміти, що якщо щось станеться із сервером чи компанія закриється або ж припинить підтримку даної версії сервісу будуть втрачені всі іконки зі свого сайту чи додатку. Далі слід відмітити те, що при підключенні через CDN не можна обирати, які шрифти вам потрібні, а які ні з чого слідує погіршення швидкості загрузки сторінки HTML так як завжди буде завантажуватись повний font awesome 5.

Другий спосіб локально зберігати всі файли цього сервісу та підключати їх так само локально. Цей спосіб найбільш безпечний для вашого проекту, так як коли файли у вас лежать локально, то навіть якщо компанія перестане підтримувати цей сервіс та закриватиме його, він буде доступний для вашого проекту та буде працювати. Так само, можна редагувати файли та залишати лише ті іконки які вам треба для мінімізації та мінімізації і швидшого завантаження сторінки.

Для використання сервісу використовують тег <i> із унікальним класом class="fas fa-info-circle" наприклад.

Рис. 2.13 іконка Font Awesome 5

2.13 Bootstrap 5

На сьогоднішній день бібліотека bootstrap оновилась до 5ої версії та принесла розробникам дуже багато цікавого та зовсім нового, але так як дуже багато проектів, які використовують минулу версію bootstrap 4 та ще не переїхали на п’яту версію, компанія залишила підтримку минулої версії на певний період.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, який розробили для створення та стилізації веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для форм, кнопок, навігації та інших компонентів інтерфейсу, а також додаткові розширення JavaScript.

Структура

Цей фреймворк створено з допомогою препроцесора стилів Less. Що собою він представляє? Це по суті конструктор, з якого можна скласти повноцінний веб-додаток чи сайт. Простими словами цей фреймворк має прописані стилі для усіх компонентів які він підтримує, наприклад навігаційного меню чи форми, та в документації наведено шаблони які доступні та структура яка має бути використана для такого шаблону. Далі через зарезервовані класи до яких написано стилі вже оформлюється сам компонент і так само по цим класам, але не тільки по ним, а і по атрибутам працює скриптова частина цього фреймворку.

Основні компоненти Bootstrap:

- **Сітки** (grid) — Написані на технології flex готові сітки, або ж колонки, які дозволяють без зайвих зусиль розбити блок на певні колонки та рядки. Батьківський клас має бути обов’язково .row. Дочірні класи

розбивають контент на колонки, яких в сумі має бути максимум 12. Приклад: два блоки із класами . col-6 або 3 блоки із класами .col-4.

- **Шаблони** (template) — Фіксовані чи адаптивні шаблони сторінок. Іншими словами вже написані за вас приклади сторінок, які вільно можна брати та налаштовувати під себе та свій проект.
- **Типографіка** (typography) — Стилістичні класи для різноманітних шрифтів, які використовуються для різних цілей. Наприклад: шрифти для коду, для цитат, для наповнення контентом сторінки тощо.
- **Мультимедіа** (media) — Засоби управління зображеннями та відео.
- **Таблиці** (table) — Спеціальні стилістичні та скриптові засоби, які полегшують керування та надають оформлення для html таблиць.
- **Форми** (form) — Написані окремі шаблони форм, деякі їх окремі компоненти та стилі для засобів їх керування. Тобто це можна використовувати, як конструктор форм так і використовувати вже готові рішення та налаштовувати їх.
- **Навігація** (nav, navbar) — Написані окремі шаблони навігаційних меню під різні формати веб сторінок та вбудована адаптація їх під мобільні девайси, яка вже реалізована та інтегрована під капотом фреймворку.
- **Сповіщення** (alert) — Стилістичні можливості для оформлення сповіщень, нагадувань та спливаючих вікон.
- **Іконочний шрифт** (icon font) — Дещо схожа технологія на Font Awesome, але не така об’ємна в плані доступних іконок.

Bootstrap.js

Окрім стилістичних функцій фреймворк вміє взаємодіяти зі своїми компонентами через вбудований javascript, який написаний із використанням такої технології JQuery, що дозволяє спілкування із компонентами зробити легким та швидким. Розглянем модулі, які доступні на сьогодні:

- **Transitions** — Цей модуль своєю назвою підказує для чого він створений. Властивість transition зазвичай використовують для того,

щоб зробити інтерактивність на сторінці плавною та згладженою. Так само й тут, він використовується для того щоб згладити та зробити анімації плавними.

- **Modal** — Модуль, який відповідає за функціональність усіх модальних вікон, які інтегруються у сервіс чи додаток чи сторінку.
- **Dropdown** — Цей модуль є одним із найпоширеніших у використанні. Він надає можливість безболісно створювати випадаючи списки без використання тегів select.
- **Scrollspy** — Плагін, що автоматично змінює активний пункт у меню залежно від позиції прокручування сторінки.
- **Tab** — Плагін відповідальний за вкладки або ж «таби». Найпопулярнішим його застосуванням є розробка навігаційного меню веб-додатку.
- **Tooltip** — Модуль, який відповідає за зрозумілість сайту чи додатку. Він надає функціонал підказок та наголошень. По суті стилізує та програмує спливаючі підказки.
- **Popover** — Аналог спливаючих підказок, але з більшими можливостями. У підказку можна дописувати який небудь заголовок, до того ж блок з'являється після кліку на об'єкт, а не після наведення.
- **Alert** — Інформаційні повідомлення, які створюються класом .alert , але з можливістю закриття.
- **Button** — Плагін для керуваннями кнопками. Завдяки методам плагіну можна змінювати стан і тип кнопки, а також створювати елементи, які поводяться як checkbox або radio button , але при цьому є звичайними блочними елементами.
- **Collapse** — Інтерактивний елемент, який дозволяє згортати та розгортати контент на сторінці.
- **Carousel** — Мультимедійна галерея зображень.
- **Affix** — Плагін, що «приліплює» меню до одного з країв екрану під час прокручування сторінки.

Висновки до розділу 2

В даному розділі було описано детально весь інструментарій, який використано для розробки даного додатку та переваги конкретних інструментів у конкретних ситуаціях. Також було наведено усі потрібні приклади для кращого розуміння наведеної інформації про технологію.

Процес визначення із інструментами для розробки цього сервісу мав дуже пильну увагу та глибокий підхід, для того, щоб обрати найефективніше, найпотужніше та найбільш підходяще.

РОЗДІЛ 3. ПРОЦЕС РОЗРОБКИ ТА ВИРІШЕННЯ УСІХ ПОСТАВЛЕНИХ ЗАДАЧ

В першому розділі даної роботи було поставлено задачу, яку треба вирішити. В другому розділі було описано весь стек технологій, які використовувались. У цьому ж розділі буде описано весь процес розробки у деталях та згадається не одноразово про всі нюанси та підводні каміння з якими довелось боротись під час розробки.

Першочергово перед написанням будь-якого коду потрібно спроектувати це та розбити процес на етапи. Етапи розробки:

- Проектування
- Визначення потрібного функціоналу
- Визначення потрібних інструментів
- Визначення із загальним дизайном
- Написання коду із паралельним налагодженням згідно обраної методології написання коду
- Тестування додатку
- Реліз додатку

Ось таким чином було розбито роботу на певні етапи весь процес розробки. Взагалі від організації процесу залежить дуже багато. Від швидкості отримання результату до якості отриманого результату.

3.1 Проектування

На етапі проектування здебільшого були прийняті усі рішення із структуризації проекту та розбиття коду по файлам. Для початку розберем для чого взагалі потрібно розбивати проект на файли та папки. Це потрібно для того, щоб легше розробляти та підтримувати проект у майбутньому. Такі розбиття називаються деревами. Приклад. Такого дерева наведено на зображенні рис. 2.14.

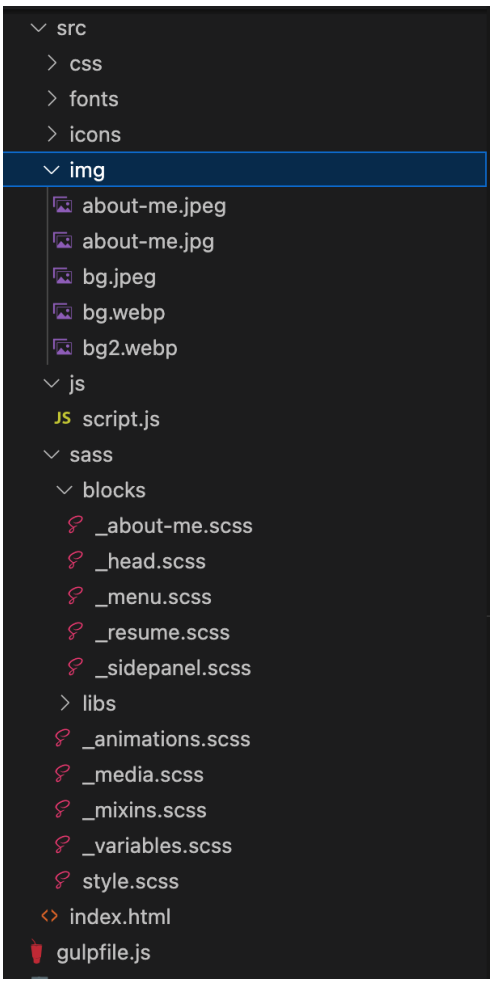

Рис. 2.14 Структурне дерево

Полегшення розробки та підтримки у майбутньому досягається через те, що коли всі стилі розбиті по різним файлам, пошук слабкого місця або ж помилок стає набагато швидшим, підтримка вже існуючого коду стає також зручнішою за рахунок того, що для пошуку місця де треба ввести якісь правки стає набагато швидшим та комфортнішим.

3.2 Визначення потрібного функціоналу

Під визначенням потрібного функціоналу розуміється, що потрібно було визначитись, які функції потрібно реалізувати для того, щоб сервіс міг вирішити поставлену задачу. Прикладом такого функціоналу можу слугувати можливість спілкуватись з іншими сервісами через Rest API.

Вміння надсилати запити це основний функціонал, який є необхідним для вирішення поставленої задачі. Це потрібно для того, щоб сервіс міг на основі

обраних користувачем значень надіслати запит до інших сервісів та отримати відповідь.

Вміння стежити за статусом запиту теж не аби, яка по важливості функція адже при негативній відповіді потрібно все одно надавати хоч якусь відповідь.

Обробка відповідей – це також основоположна функція даного сервісу адже від зрозумілості відповіді для користувача залежить подальша взаємодія користувача із сервісом. Цей функціонал має забезпечити найбільш оптимізований «парсинг» текстової відповіді від сервера та вставка відповіді до DOM структури HTML коду для відображення для користувача.

3.3 Визначення потрібних інструментів

На основі визначеного функціоналу можливо зробити вибірку інструментів необхідних для розробки. Цей вибір зазвичай робиться через порівняння інструментів із однієї галузі та вибір найбільш підходящого під вирішення поставлених задач.

Браузер

Мною було обрано браузер Google Chrome так, як його можливості для розробників просто нещадно обганяють усі інші браузери. Другим пунктом на корить Google Chrome була база користувачів. Він являється найпопулярнішим браузером у світі тому найбільшої уваги потребує саме підтримка цього браузера.

IDE

Середа розробки це той інструмент від зручності якого залежить напряму результат роботи, тому особливу увагу приділено вибору середи розробки сервісу. Мій вибір пав на VSCode через ряд переваг в сторону VSCode. Усі ці переваги було описано в першому розділі.

3.4 Технології програмування

Усі технології програмування було обрано в порівнянні із своїми конкурентами та змінниками, але є технології замінників яких просто не існує. Прикладом таких технологій можуть слугувати CSS та HTML. У будь-якому веб-додатку та веб-сторінці для будування каркасу використовується HTML та для стилізації використовується CSS. Інша справа коли для написання каркасу та стилізації використовуються препроцесори та фреймворки, але все одно на виході ви отримуєте голий HTML та CSS.

Для написання логіки та функціоналу було обрано стандартну мову програмування JavaScript. Ця мова програмування має усі необхідні можливості для реалізації всього того функціоналу, який було описано. Під необхідними можливостями розуміється не тільки нативний функціонал, але й інтеграцію та можливість працювати із сторонніми технологіями. Наприклад фреймворки, бібліотеки тощо.

Для полегшення розробки та написання коду було знайдено таку технологію, як gulp. Цей менеджер задач та його плагіни полегшують взаємодію із фреймворками та бібліотеками.

3.5 Визначення із загальним дизайном

Дизайнерська робота – це здебільшого творча робота, але потрібно розуміти деякі підводні каміння.

Першим може бути адаптивність. Кожен нормальний веб-додаток та сайт має бути адаптивним так як на сьогоднішній день люди дивляться інтернет з телефона набагато більше ніж з комп’ютера.

Другий камінь це читаність сторінки. Для того, щоб на сторінку було приємно дивитись та читати потрібно обирати палітру кольорів по сумісності. Наприклад чорний та білий дуже добре виглядають в поєднанні між собою, а ось синій та зелений дуже погано. Також не слід проходити повз шрифти. Шрифт дуже важлива складова веб-дизайну та його правильний чи не правильний підбір сильно впливає на сприйняття користувачем інформації.

Третім питанням було оформлення кнопок та полів вводу. Найбільш важливим було дотриматись усіх стандартів та правил «гарного тону» у сфері веб-розробки, які актуальні на сьогоднішній день.

3.6 Написання Front End частини

Написання коду можна поділити на 2 гілки. Написання Front End коду та написання Back End коду.

Front End частина коду, яка відповідає за: структури сторінок, стилістику додатку та розміщення контенту. Дана частина програми розроблена із застосуванням таких інструментів, як HTML5, CSS, SCSS, Gulp, Font Awesome, Bootstrap. Розглянемо все по порядку.

Для написання каркасу сайту було використано HTML5 та декілька фреймворків і бібліотек. На початку, було створено базову структуру усіх сторінок та розбито все на блоки обгортки, які надалі будуть слугувати контейнерами для контенту. Детальну схему будови сторінки наведено нижче.

Рис. 3.1 Схема будови основної сторінки.

Розглянемо кожен блок окремо та розпочнемо з Header. Header – це шапка сторінки. Зазвичай такі блоки містять в собі навігаційне меню та логотип

із назвою компанії, додатку тощо. На наступному рисунку 3.2 буде показано яку шапку було створено для цього додатку.

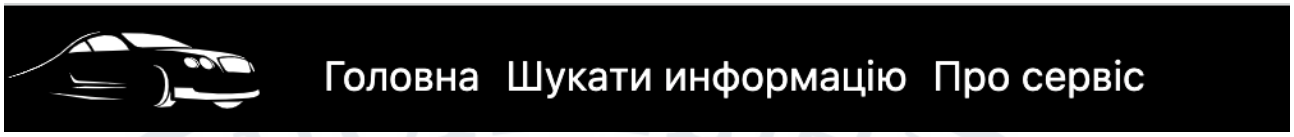

Рис. 3.2 Шапка додатку

Далі розглянемо один із двох найбільш важливих блоків додатку. Блок де розміщено поля вводу інформації. В даному додатку використовуються лише поля типу select так, як найбільш зручний спосіб для користувача визначити яка саме марка, модель та кузов автомобіля цікавить. Тобто простіше кажучи в нас є декілька полів типу select та кожен з них відповідає за конкретну характеристику авто. Варіанти відповіді в кожному наступному полі починаючи з другого варіюються залежно від того які відповіді обрано раніше. Наприклад в першому полі задано марку BMW. В такому випадку користувачеві не доцільно надавати можливість обрати модель яка належить Mercedes чи Audi. Тому для цього написано логіку, яка відслідковує це та замінює варіанти в наступному полі. Так саме і з кузовом. Прикладом може слугувати така ситуація. Користувач обирає марку BMW та модель 5 series. В такому випадку йому не доцільно показувати такі кузова авто як e90 чи f13. Тому, що ці кузова притаманні моделям 3 series та 6 series відповідно. Як саме це зроблено докладніше описано у пункті де описана логіка проекту (Back End частина).

Наступним не менш важливим блоком є блок із інформацією, яку сервіс отримує в результаті надсилання запиту. Цей блок являється простою обгорткою в яку серверна частина проекту підкидає потрібну інформацію, але детальніше про цей блок описано у пункті де описана логіка проекту (Back End частина).

Footer так само, як і Header являється тривіальним блоком в якому вказана інформація про те коли створено додаток, контактна інформація та інше.

Слід сказати про те, що це не одно сторінковий додаток та основна сторінка це сторінка пошуку інформації про автомобіль, про яку описано вище. Інші дві сторінки «Про сервіс» та «Головна».

На сторінці «Про сервіс» надано інформацію про засновника та розробника даного додатку, яка мета створення цього сервісу та таке інше.

Сторінка «Головна» містить в собі інформацію про марки авто які наразі доступні для пошуку, який функціонал присутній у сервісі та, як ним користуватись, для кого потрібен цей додаток та для чого він загалом. Інформація про марки авто з якими сервіс вміє працювати наведено у вигляді «каруселі» написаної за допомогою бібліотеки Slick slider.

Рис. 3.3 Карусель із марками авто

Бібліотека slick slider це одна із найпопулярніших бібліотек для створення подібних і не тільки «слайдерів» для веб-додатків. Написаний він на JQuery та має дуже великий функціонал та гнучкі налаштування. Також він дуже простий у налаштуванні та це надає можливість дати волю своїй фантазії та створювати неймовірні речі. Із основних налаштувань можна виділити:

- Autoplay – характеристика, яка відповідає за автоматичне перемикання слайдів або ж її відсутність.
- autoplaySpeed – відповідає за швидкість автоматичного перемикання кадрів. Швидкість визначається у мілі секундах.
- Accessibility - вмикає навігацію за допомогою вкладок і стрілок. Якщо autoplay не встановлено: true, після зміни слайда встановлює фокус браузера на поточний слайд (або перший із поточного набору слайдів,

якщо декілька слайдів показується за один раз). Для повної відповідності увімкніть focusOnChange на додаток до цього.

- adaptiveHeight – характеристика, яка відповідає за адаптацію висоти «слайдера».
- Arrows – підключає стрілки навігації.
- appendArrows - змінює місце приєднання стрілок навігації (Selector, htmlString, Array, Element, об’єкт jQuery).
- Dots - підключає точки навігації.
- appendDots - змінює місце приєднання точок навігації (Selector, htmlString, Array, Element, об’єкт jQuery).
- lazyLoad - приймає «ondemand» або «progressive» для техніки відкладеного завантаження. 'ondemand' завантажить зображення, щойно ви переміститеся до нього, 'progressive' завантажує одне зображення за іншим під час завантаження сторінки.

Приклад налаштувань, які використовувались для розробки даного додатку показано на рис. 3.4.

```
$(document).ready(function() {
    $('.intro__slider').slick({
        speed: 1200,
        adaptiveHeight: true,
        prevArrow: '<button type="button" class="slick-prev"><img src="icons/chevron_left.svg"></button>',
        nextArrow: '<button type="button" class="slick-next"><img type="button" src="icons/chevron_right.svg"></button>',
        slidesToScroll: 1,
        autoplay: true,
        autoplaySpeed: 2000,
        responsive: [
            {
                breakpoint: 992,
                settings: {
                    arrows: false,
                    dots: true,
                    adaptiveHeight: true
                }
            }
        ]
    });
```

Рис. 3.4 Налаштування Slick Slider

Уся інша інформація подана на сторінці «Головна» у текстовому вигляді із використанням непронумерованих списків.

Після того, як каркас готовий, його потрібно оформити та за стилізувати. Ось на цьому етапі починається найцікавіше із роботи по Front End частині. Для стилізації веб-додатків використовують CSS та різноманітні бібліотеки для нього. У даному проекті використовувалась бібліотека SCSS для написання стилів на мові препроцесора та плагін gulp-sass для компіляції коду з SCSS до звичайного CSS. Для написання стилів використовувались усі потрібні інструменти доступні для використання.

Першим із таких інструментів можна відмітити змінні SCSS. Їх було використано для того, щоб мінімізувати та спростити написані коди стилів. Спрощення відбулось за рахунок того, що змінними було замінено усі потрібні кольори, контури, розміри шрифтів і т д. Таким чином маємо, що замість того, щоб розбиратись який колів вказано у форматі HEX його було замінено на змінну по типу $color-white тощо. Таким чином було створено оригінальну палітру кольорів для розробки веб-додатку. Так само із розміром шрифту та іншим. Також не слід забувати про те, що це підвищує також швидкість написання коду, тому що в результаті не потрібно гаяти лишній час на те, щоб знайти якийсь код кольору тощо.

Наступним інструментом є міксини. Міксини – це також дуже зручна штука для економії часу та зменшення написаного коду. Вони працюють наступним чином. Для прикладу, якщо у проекті є якийсь цілий елемент, який повторюється декілька разів у різних блоках то для його оформлення варто використати міксин. Чому саме міксин варто використати? Тому, що для кожного блоку варто використовувати окремий файл із окремими стилями, як каже методологія BEM. З чого випливає, що його стилізацію потрібно буде копіювати та переносити стільки разів у скількох блоках він повторюється, а якщо створити окремий файл для міксинів та описати його там, тоді за допомогою такого інструменту, як @include можна переносити цілі шматки коду. Приклад використання на рисунку 3.5.

SCSS Sass

```scss
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}
.box { @include transform(rotate(30deg)); }
```

CSS

```css
.box {
  -webkit-transform: rotate(30deg);
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}
```

Рис. 3.5 Приклад міксина

У даному проекті міксини використовувались лише для того, щоб оформляти однаково текст на різних сторінках.

Далі слід розглянути розбиття на блоки та, як це працює. У кожному більш-менш новому проекті кожен блок чи сторінка має окремий файл зі стилями. Це робиться для того, щоб код був більш читабельним, а структура проекту більш зрозуміла. Але усі ці стилі потрібно десь об’єднати тому дуже часто роблять наступним чином, як на рисунку 3.6.

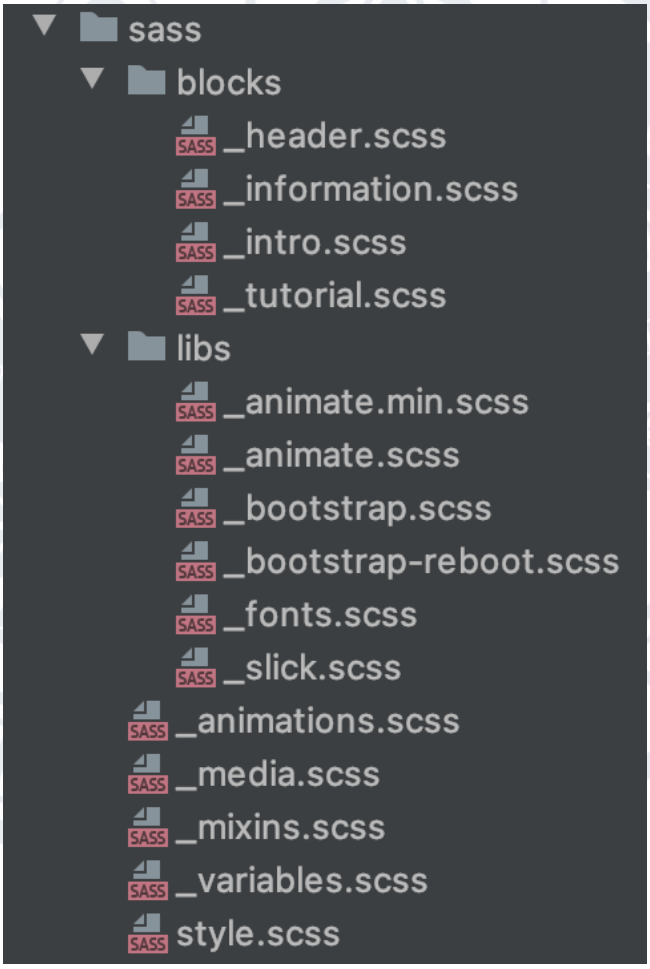

Рис. 3.6 Файли стилів

Як видно з малюнка вище, у даному проекті кожен файл відповідає за окрему частину коду та лежить у відповідній папці, тому розберем все по порядку:

- папка sass – папка в якій лежать абсолютно усі стилі проекту.
- Папка blocks – папка в якій лежать файли зі стилями лише окремих блоків сторінок.
- _header.scss – файл де прописані усі стилі для оформлення шапки додатку.
- _information.scss – файл із стилями для блока, який відповідає за виведене інформацію про авто.
- _intro.scss – файл де записано усі стилі, які відповідають за блок де надано ввідну інформацію про сервіс.
- _tutorial.scss – файл де записано стилі, які відповідають за блок із інформацією про те як користуватись сервісом.
- Папка libs – папка в якій зберігаються усі бібліотеки підключені до проекту. Усі бібліотеки було викачано та підключено локально за для того, щоб проект не втратив частину функціоналу, якщо раптом хоч якась із цих бібліотек перестане підтримуватись.
- _animate.min.scss та _animate.scss – файли бібліотеки animate, яка надає можливість на льоту підключати різноманітні анімації для елементів сторінок.
- _bootstrap.scss та _bootstrap-reboot.scss – файли відповідальні за бібліотеку bootstrap.
- _slick.scss – файл бібліотеки Slick Slider.
- _fonts.scss – файл відповідальний за шрифти по всьому проектові.
- _animations.scss – файл у якому написані власні анімації для елементів.
- _media.scss – файл із медіа запитами. Іншими словами саме цей файл відповідає за те, як користувачі бачать веб-додаток на девайсах менших від ноутбука.
- _mixins.scss – файл із міксинами.
- _variables.scss – файл зі змінними.

- style.scss – найважливіший файл де поєднано усі ці файли.

Найбільш важливим файлом у цьому проекті являється файл style.scss так як саме він відповідає абсолютно за всі стилі веб-додатку, але в той самий час він є і найменшим файлом якщо рахувати по стрічкам коду. Вміст цього файлу видно на рисунку 3.7.

```
@import 'libs/bootstrap-reboot';
@import 'libs/bootstrap';
@import 'libs/fonts';
@import 'libs/slick';
@import 'libs/animate';

@import 'variables';
@import 'mixins';
@import 'animations';

@import 'blocks/header';
@import 'blocks/intro';
@import 'blocks/tutorial';
@import 'blocks/information';

@import 'media';

* {
    box-sizing:border-box;
    margin: 0;
    padding: 0;
}

html {
    font-family: 'Museo Sans Cyrl';
    font-weight: 300;
}

.container {
    max-width: 1140px;
    margin: 0 auto;
    padding: 0 15px;
}
```

Рис. 3.7 файл style.scss

Якщо домислити, що всі ці файли збираються в один до купи за допомогою якогось @import це не важко, то зрозуміти чому саме такий порядок підключення можуть не всі, але є пояснення. Так як кожен імпорт по суті просто копіює код із файлу який імпортується та вставляє замість @import цей код у файл у який імпортується код, тому потрібно слідкувати за тим який код вище, а який нижче, адже якщо за імпортувати наприклад змінні в останню чергу , то їх не вийде використати ні в одному місці проекту і т д. Саме тому, з самого початку йде імпорт усіх бібліотек, які будуть використовуватись, далі усі файли із міксинами, шрифтами, змінними і таким іншим. Слід за ними йдуть усі блоки по черзі, а в кінці медіа запити для адаптації усього вище написаного коду.

Після написання SCSS коду його потрібно скомпілювати у звичайний CSS тому, що браузер не розуміє код написаний не на CSS. Для цього застосовують спеціальні компілятори, наприклад плагін для Gulp gulp-sass. Цей плагін потрібно підключити через gulp.js та налаштувати його там же. Приклад налаштувань цього плагіна можна побачити на рис. 3.8.

```
gulp.task('watch', function() {
    gulp.watch( glob: "src/sass/**/*.+(scss|sass|css)", gulp.parallel('styles'));
    gulp.watch( glob: "src/*.html").on('change', gulp.parallel('html'));
    gulp.watch( glob: "src/fonts/**/*").on('change', gulp.parallel('fonts'));
    gulp.watch( glob: "src/js/*.js").on('change', gulp.parallel('scripts'));
    gulp.watch( glob: "src/img/**/*").on('change', gulp.parallel('images'));
    gulp.watch( glob: "src/icons/**/*").on('change', gulp.parallel('icons'));
});
```

рис. 3.8 Налаштування gulp-sass

На рисунку 3.8 відображено комплексну задачу gulp.js яка водночас компілює SCSS код в CSS, стискає html, компілює файли шрифтів, стискає js файли, стискає картинки та іконки і в кінці кінців переносе всі ці нові файли в потрібне місце в проекті.

Після написання структури та стилів йде реалізація логіки та функціональності проекту.

3.7 Написання Back End частини

Back End частина відповідає за функціональність веб-додатку та логіку його поведінки. Для реалізації логіки даного додатку використовувались наступні технології: JQuery, XmlHttpRequests, JavaScript, API сторонніх сервісів та інше.

З самого початку стояла задача з реалізації налаштування запиту, який потрібно надіслати. Під налаштуванням мається на увазі надання деяким змінним, які відповідають за конкретні характеристики запиту, якихось коректних значень. Під цими змінними розуміється марка, модель, кузов автомобіля. Для повного розуміння, як це працює слід розібратись із API сервісу до якого надсилається запит.

Для того, щоб мати можливість працювати із API сторонніх додатків здебільшого потрібно реєструвати аккаунт розробника та отримати всі потрібні дозволи зі сторони стороннього сервісу. Після реєстрації аккаунту стануть доступними ключі та шляхи для авторизації для надсилання запитів та отримання відповідей від сервера. Здебільшого сервіси в яких API працює з не секретними чи конфіденційними даними використовують авторизацію через API key який генерується вручну в профілі аккаунту розробника.

Наступним етапом розробки йде налаштування полів вводу інформації. Логіка досить проста та тривіальна, в залежності від обраних попередніх варіантів у випадаючих списках варіанти відповіді мають змінюватись у наступних списках та вони мають розблоковуватись. Наприклад при обраній марці BMW у користувача не має бути можливості обрати модель іншої марки автомобіля.

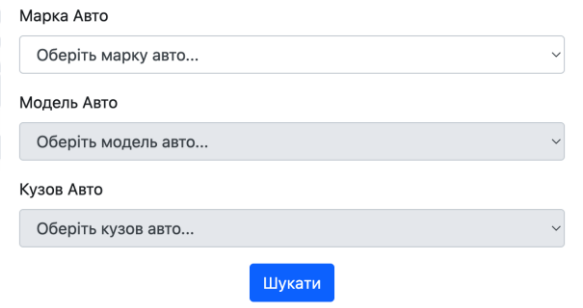

Рис. 3.9 Поля вводу інформації

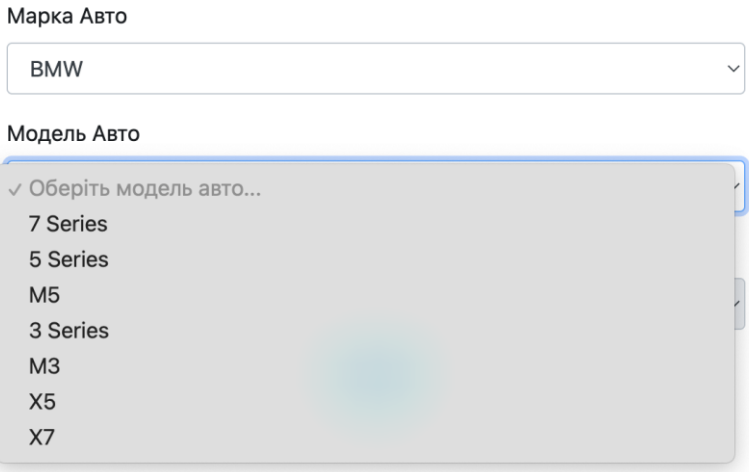

Рис. 3.10 моделі BMW

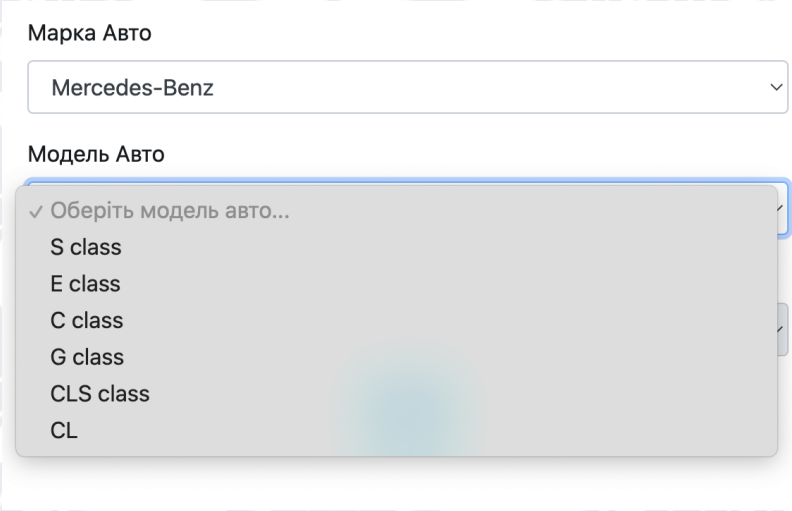

Рис. 3.11 моделі Mercedes

Для реалізації цього функціоналу було використано звичайні JQuery запити орієнтовані на залежність від варіанту відповіді у першому списку. Будова запиту наступна: 1) отримати значення з першого списку 2) в залежності від отриманого значення відібрати значення, які потрібно підставити у варіанти відповіді в другий по номеру список 3) підставити значення.

```
$('#AutoModel').append("" +
    "<option brand='bmw' value='7ser'>7 Series</option>" +
    "<option brand='bmw' value='5ser'>5 Series</option>" +
    "<option brand='bmw' value='m5'>M5</option>" +
    "<option brand='bmw' value='3ser'>3 Series</option>" +
    "<option brand='bmw' value='m3'>M3</option>" +
    "<option brand='bmw' value='x5'>X5</option>" +
    "<option brand='bmw' value='x7'>X7</option>" +
    "<option brand='mb-benz' value='s-class'>S class</option>" +
    "<option brand='mb-benz' value='e-class'>E class</option>" +
    "<option brand='mb-benz' value='c-class'>C class</option>" +
    "<option brand='mb-benz' value='g-class'>G class</option>" +
    "<option brand='mb-benz' value='cls-class'>CLS class</option>" +
    "<option brand='mb-benz' value='cl-class'>CL</option>" +
    "<option brand='porsche' value='panamera'>Panamera</option>" +
    "<option brand='porsche' value='macan'>Macan</option>" +
    "<option brand='porsche' value='cayenne'>Cayenne</option>" +
    "<option brand='audi' value='a8'>A8</option>" +
    "<option brand='audi' value='a7'>A7</option>" +
    "<option brand='audi' value='a6'>A6</option>" +
    "<option brand='audi' value='a4'>A4</option>" +
    "<option brand='chevy' value='impala'>Impala</option>" +
    "<option brand='chevy' value='camaro'>Camaro</option>" +
    "");
```

Рис. 3.12 Усі моделі авто

```
$('select#AutoBrand').change( function() {
    let auto_brand = $('select#AutoBrand option:checked').val();

    $('#AutoModel').removeAttr('disabled');

    if(auto_brand === 'bmw') {

        $("option[brand = 'bmw']").removeAttr('style');
        $("option[brand = 'mb-benz']").css({'display':'none'});
        $("option[brand = 'porsche']").css({'display':'none'});
        $("option[brand = 'audi']").css({'display':'none'});
        $("option[brand = 'chevy']").css({'display':'none'});

    } else if (auto_brand === 'mb-benz') {

        $("option[brand = 'mb-benz']").removeAttr('style');
        $("option[brand = 'bmw']").css({'display':'none'});
        $("option[brand = 'porsche']").css({'display':'none'});
        $("option[brand = 'audi']").css({'display':'none'});
        $("option[brand = 'chevy']").css({'display':'none'});

    } else if (auto_brand === 'porsche') {

        $("option[brand = 'porsche']").removeAttr('style');
        $("option[brand = 'bmw']").css({'display':'none'});
        $("option[brand = 'mb-benz']").css({'display':'none'});
        $("option[brand = 'audi']").css({'display':'none'});
        $("option[brand = 'chevy']").css({'display':'none'});

    } else if (auto_brand === 'audi') {

        $("option[brand = 'audi']").removeAttr('style');
        $("option[brand = 'bmw']").css({'display':'none'});
        $("option[brand = 'mb-benz']").css({'display':'none'});
        $("option[brand = 'porsche']").css({'display':'none'});
        $("option[brand = 'chevy']").css({'display':'none'});
```

Рис. 3.13 перевірки на марку автомобіля

Таким чином відбувається налаштування усіх полів вводу інформації для подальшого налаштування запитів до сторонніх сервісів.

Наступний крок розробки серверної частини додатку йде розробка зв'язку між полями вводу інформації із локальними змінними мови JavaScript. Найбільш простий спосіб для реалізації цього це той самий JQuery. Після того, як користувач обрав усі необхідні відповіді у полях вводу, за цими відповідями закріпляються певні значення. Для того, щоб отримати ці значення і використовувати їх в подальшому для побудови запитів, потрібно звернутись через JQuery запит до цих полів вводу по їх ID та отримати їх Value (значення). Цей запит, як і наступні дії мають бути зав’язаниі на подію click по кнопці пошуку інформації.

```
let brand = $("#AutoBrand").val();
let model = $("#AutoModel").val();
let body = $("#AutoBody").val();
```

Рис. 3.14 Отримання значень з полів вводу

Після того, як у серверна частина отримала усі необхідні змінні для формування запиту до API іншого сервісу розпочинається формування запиту. Для цього потрібно значення цих змінних конвертувати під кожне API окремо, так як наприклад фільтрація по марці BMW в одному сервісі позначається, як brand='bmw’ а в іншому automobile_brand=’12’. Для того, щоб знайти інформацію про це треба вивчити документацію по API. Приклад url повністю налаштованого запиту:
https://auto.ria.com/reviews/api/search_api?categoryId=0&markaId=6&modelId=1943&baseId=0&yearFrom=0&yearTo=0&bodyId=0&fuelId=0&page=1&sort=0&langId=4&size=20&firstOwner=false

Результатом цього запиту являється дуже великий масив даних із усіма відгуками та характеристиками про автомобілі Audi Q7.

Наступним завдання стояло розібрати результат та оформити його у людському вигляді. Для вирішення цієї задачі було написано так званий «парсер», який отримує на вході масив даних, перетворює його у потрібний формат та видає результат користувачеві (рис. 3.15 та рис. 3.16).

Переваги BMW E39

+ Вже базова комплектація автомобіля передбачає кондиціонер, подушки безпеки (6 штук), електропакет (вікна та дзеркала), антиблокувальну та антипробуксовувальну систему, гідропідсилювач та електронну систему стабілізації. При цьому у продажу часто можна знайти і комплектацію багатшою, що включає двозонний клімат – контроль.
+ Стильний та комфортний салон . При цьому він досить просторий та місткий, що особливо важливо, якщо є необхідність встановлення дитячого крісла. Основні види оббивки сидінь – шкіра та тканина. Випускалися автомобілі і з комбінованим салоном – тканинними сидіннями та шкіряними підголівниками.
+ Двоконтурна гідросистема гальм . Перший впливає на передні гальма, другий – на задні. Таким чином забезпечується гальмування автомобіля, якщо один із контурів вийшов з ладу (наприклад, через розгерметизацію системи). Така продумана гальмівна система цілком логічна для такого потужного та швидкісного автомобіля.
+ Ідеально підійде для водіїв-початківців, завдяки хорошій керованості . Машина дуже чутлива, що не може не надати впевненості.
+ Надійний та потужний двигун на будь-якій моделі, незалежно від року випуску. Якщо все ж таки потрібен дрібний ремонт, то великих витрат вдасться уникнути, використовуючи якісні неоригінальні деталі. А ось капітальний ремонт може вимагати великих витрат, тому при придбанні автомобіля слід провести ретельну перевірку двигуна.
+ Надійна коробка передач . Причому це стосується і механічної, і автоматичної коробки. Є моделі з автоматичною трансмісією, що передбачає ручне перемикання. При великих пробігах масло може почати просочуватися через сальники, тому потрібно стежити, щоб воно не виходило з коробки. Заміна сальників великих витрат не вимагатиме.
+ Хороший ресурс зчеплення на автомобілях з механічною коробкою передач (150 000 – 200 000 км).

Рис. 3.15 Результат частина перша

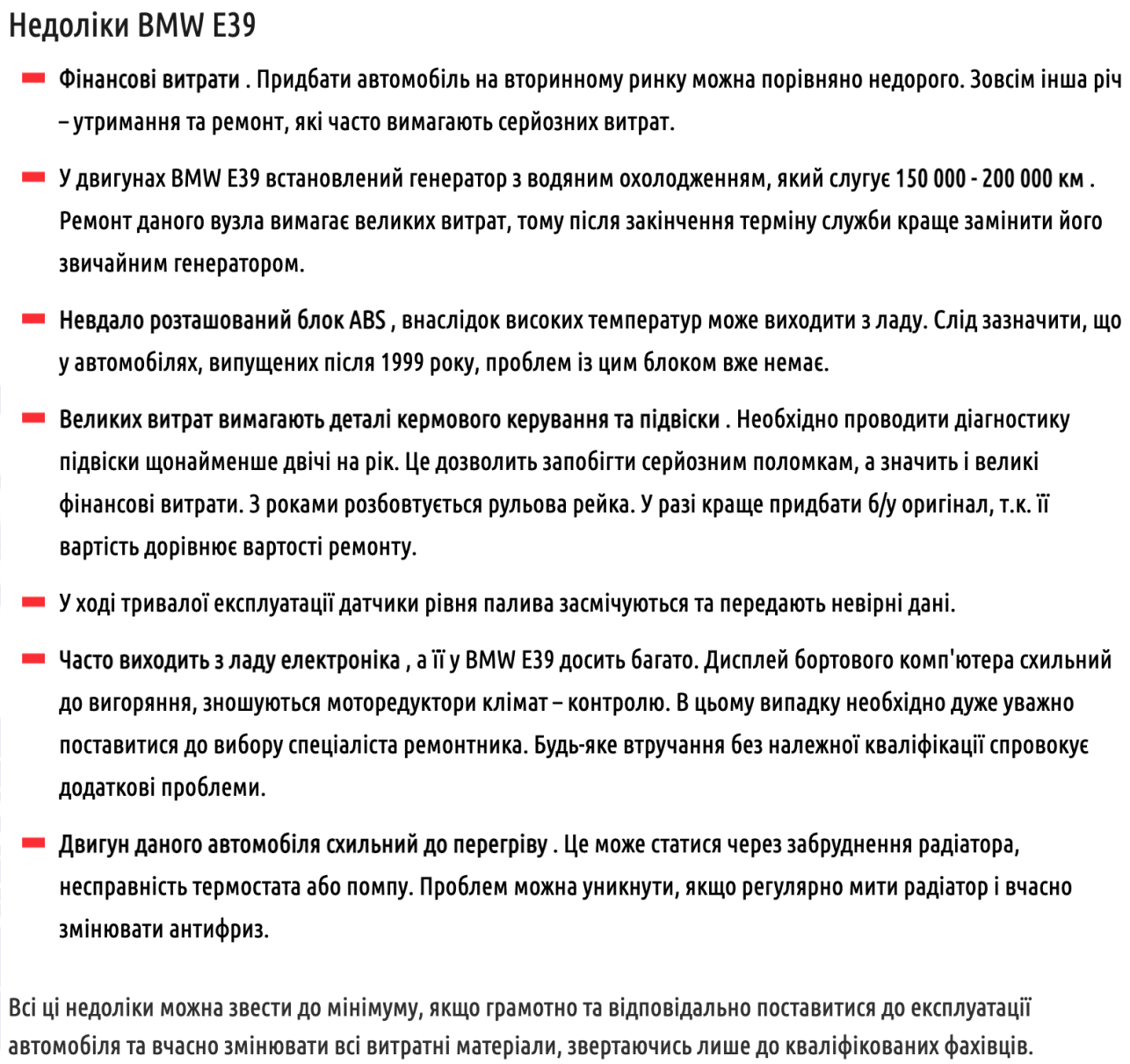

Рис. 3.16 Результат частина друга

Висновок до 3 розділу

В даному розділі було описано поетапно процес розробки додатку, описано усі тонкості та нюанси з якими прийшлось зіткнутись під час розробки. Детально було описано, як працює весь функціонал сервісу та як його було реалізовано.

Дизайн є мінімалістичним та зрозумілим, що наддасть переваги поміж інших додатків із схожою тематикою, а функціонал розроблено так, щоб користувач отримав максимально коректний результат роботи програми.

ВИСНОВКИ

Підсумувавши усе вище сказане можна виділити декілька основних питань, які було розглянуто.

Першим питанням було те, що для розробки веб-додатків існує дуже велика кількість різноманітних фреймворків та бібліотек, які полегшують розробку, але все одно 2 речі, які неможливо замінити це HTML та CSS.

Друге питання було стосовно роботи з API. Потрібно розуміти, як працює API стороннього сервісу для того, щоб наладити коректну роботу із ним.

Третім можна відмітити питання стосовно дизайну та макету додатку. Він повинен мати мінімалістичний дизайн із мінімальною кількістю лише вкрай необхідної інформації для зручності користувачів. Це приваблює користувачів, так як вони одразу розуміють, що це, для чого цей додаток та як ним користуватись.

Четверте питання стосовно логіки та функціональності. Для реалізації логіки існує дуже багато різноманітних технологій, тому слід відповідально підходити до вибору технології для вирішення того чи іншого питання.

Наступним та найбільш вагомим було питання доцільності та потрібності реалізації такої системи. Для того, щоб така система мала хоча б якийсь сенс вона має бути унікальною та мати унікальний функціонал, який не мають її аналоги та схожі системи. Для цього потрібно підійти творче до цього питання, придумати унікальні властивості та функції, які будуть вирізняти цей додаток серед багатьох інших та впливати на те, щоб користувач розумів, що процес пошуку інформації через саме цей додаток буде швидшим, зручнішим та змістовнішим.

СПИСОК ДЖЕРЕЛ

1. HTML Wiki. URL: https://ru.wikipedia.org/wiki/HTML (дата звернення: 11.06.2022)
2. W3School HTML. URL: https://www.w3schools.com/html/ (дата звернення: 11.06.2022)
3. html.com. URL: https://html.com/ (дата звернення: 12.06.2022)
4. W3School CSS. URL: https://www.w3schools.com/css/ (дата звернення: 11.06.2022)
5. Developer Mozilla. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 12.07.2022)
6. Дизайн та макет. URL: https://dan-it.com.ua/uk/rozrobka-mobilnih-dodatkiv-vid-a-do-ja-povnij-gajd/ (дата звернення: 13.06.2022)
7. Git Wiki. URL:]https://uk.wikipedia.org/wiki/Git (дата звернення: 20.08.2022)
8. Особливості Git. URL: https://sebweo.com/scho-take-git-ta-github-kerivnitstvo-dlya-pochatkivtsiv/ (дата звернення: 21.06.2022)
9. Vue.js.org. URL: https://vuejs.org/ (дата звернення: 21.06.2022)
10. Vue Wiki. URL: https://en.wikipedia.org/wiki/Vue.js (дата звернення: 22.05.2022)
11. RIA API Docs. URL: https://developers.ria.com/ (дата звернення: 25.06.2022)
12. GitHub Rest API AutoRia. URL: https://github.com/ria-com/auto-ria-rest-api (дата звернення: 27.07.2022)
13. Developer Mozilla JSON parse. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse (дата звернення: 27.06.2022)
14. JSON. URL: https://www.json.org/json-en.html (дата звернення: 27.06.2022)
15. Beginning JSON 1st edition. Ben Smith. From: Apress, 2015, 210 с (дата звернення: 27.07.2022)
16. JavaScript: сильні сторони. Douglas Crockford. From: O'Reilly Media, 2008, 176 с (дата звернення: 20.08.2022)

17. GitBook. URL: https://git-scm.com/doc (дата звернення: 07.07.2022)
18. JSON examples. URL: https://json.org/example.html (дата звернення: 06.07.2022)
19. Веб-додатки. URL: https://habr.com/ru/post/450282/ (дата звернення: 10.07.2022)
20. Web apps. URL: https://webcase.com.ua/blog/cho-takoe-web-prilozhenie-vse-vidy/ (дата звернення: 11.07.2022)
21. Fonts. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS/Styling_text/Fundamentals (дата звернення: 12.07.2022)
22. Font Awesome. URL: https://fontawesome.com/ (дата звернення: 12.07.2022)
23. BootStrap. URL: https://getbootstrap.com/ (дата звернення: 12.07.2022)
24. Bootstrap lessons. URL: https://itproger.com/course/bootstrap (дата звернення: 13.07.2022)
25. Vue lessons. URL: https://habr.com/ru/company/ruvds/blog/458324/ (дата звернення: 13.07.2022)
26. Requests and servers. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data (дата звернення: 15.07.2022)
27. Веб API. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs (дата звернення: 16.07.2022)
28. Сторонні API. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Third_party_APIs (дата звернення: 17.07.2022)
29. Асинхронний JS. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous (дата звернення: 10.08.2022)
30. Promise. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Promises (дата звернення: 11.08.2022)
31. Workers. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Introducing_workers (дата звернення: 11.08.2022)

32. Promise-based API. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Implementing_a_promise-based_API (дата звернення: 12.08.2022)
33. JS Objects. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects (дата звернення: 13.08.2022)
34. Prototypes. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object_prototypes (дата звернення: 16.08.2022)
35. Classes in JavaScript. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Classes_in_JavaScript (дата звернення: 18.08.2022)
36. Работа с JSON даними. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON (дата звернення: 20.09.2022)
37. Vue components. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_rendering_lists (дата звернення: 21.09.2022)
38. Vue and CSS. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_styling (дата звернення: 21.09.2022)
39. Vue props. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_computed_properties (дата звернення: 22.09.2022)
40. Vue resources. URL: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Vue_resources (дата звернення: 23.09.2022)