ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: **«Методологія застосування програмних засобів при розробці комерційних вебдодатків»**

Виконав: здобувач вищої освіти за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр групи 126ІСТ_мз_2023[1](л.н.)
Шкода Вадим Анатолійович
Керівник: Калініченко Антоніна Володимирівна
Рецензент: Муравльов Володимир В'ячеславович

Полтава – 2024 року

ВСТУП

Актуальність теми кваліфікаційної роботи пов'язана з потребами кваліфікованої експертизи широкого спектру вебтехнологій та неоднозначністю вибору програмних засобів для реалізації вебдодатків, зокрема, комерційного або інформаційного призначення. Невпинне удосконалення та диверсифікація функцій в галузі веброзробки призводить до появи величезної кількості інструментів різного призначення. Розуміння та впровадження в роботу новітніх технологій стає критично важливим для досягнення конкурентних переваг. Як відомо, вебдодатки використовуються в різних сферах, що накладає певні умови на функціонал та дизайн, а відтак вимагає дослідження для вдосконалення результативності та оптимальності вибору програмних засобів проєктування.

Отже, враховуючи широкий спектр можливостей, які надають програмні засоби веброзробки та дизайну, важливо провести ретельне дослідження їх характеристик, особливостей та результатів використання, взаємозв'язку.

Зв'язок роботи з науковими програмами, темами. Робота виконана у відповідності до науково-дослідної ініціативної теми «Організаційно-методологічні аспекти впровадження інформаційно-комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки» ДРН 0123U105060, яка реалізується на кафедрі інформаційних систем та технологій Полтавського державного аграрного університету (ПДАУ).

Метою кваліфікаційної роботи є здійснення всебічного аналізу та обґрунтованого вибору технологій розробки, якісного дизайну та SEO-оптимізації сучасних вебдодатків комерційного спрямування.

Завданнями для досягнення мети визначені:

– дослідити тенденції розвитку сучасних вебтехнологій;

– провести аналіз та обґрунтований вибір програмних засобів веброзробки (мов програмування, фреймворків, бібліотек і т. ін.);

– дослідити сучасні тенденції дизайну та засобів досягнення функціональності й властивостей комерційних вебсайтів;

– здійснити вибір програмного середовища розробки для створення вебдодатку;

– розробити дизайн та його технічну реалізацію на прикладі комерційного вебдодатку з використанням обраних технологій та заходів з оптимізації для покращення продуктивності та швидкості завантаження.

Об'єктом дослідження кваліфікаційної роботи є вибір та комбінування спеціалізованих вебтехнологій та інтегрованих програмних середовищ при проєктуванні вебдодатків та сервісів.

Предметом дослідження є властивості й технічні аспекти застосування сучасних вебтехнологій та інтегрованих програмних середовищ.

Методи наукових досліджень: абстрактно-логічний, інформаційно-пошуковий, аналітичний, розрахунково-конструктивний, технічного аудиту вебсайтів, графічний.

Інформаційну базу кваліфікаційної роботи складають: наукові статті, тематична література, статистичні дані аналітичних компаній, які розміщені у вільному доступі, власні та фахові наукові дослідження з поставленої проблеми.

Елементи наукової новизни полягають у здійсненні аналізу та оцінці впливу сучасних вебтехнологій на якісні параметри розробки вебдодатків.

Дістало подальшого розвитку:

– напрями розвитку сучасних вебтехнологій та інструментарію розробки;

– методологія вибору та використання сучасного інструментарію розробника, під час створення вебдодатку комерційного призначення.

Практична значущість одержаних результатів: результати роботи базуються на наукових дослідженнях з можливістю практичного втілення, підкріплені оригінальними розробками. Застосування результатів кваліфікаційної роботи дає змогу розширити розуміння можливостей технологій веброзробки та досягати поставлених задач у цій сфері.

Апробація результатів дослідження відбувалася шляхом оприлюднення доповідей на наукових міжнародній та студентській конференціях.

Публікації. За результатами проведеного дослідження опубліковано тези доповідей: «Системи управління проєктами: порівняльна характеристика функціональних можливостей та зручність використання для бізнесу», представлені на IV міжнародній науково-практичній конференції «Сучасне підприємництво: проблеми теорії та практики» (м. Дніпро, 19 травня 2023 року); «Використання мови TypeScript при розробці вебсайтів» на I міжнародній науково-практичній конференції «Стратегічний менеджмент агропродовольчої сфери в умовах глобалізації економіки: безпека, інновації, лідерство» (м. Полтава, 28 вересня 2023 року).

Обсяг і структура кваліфікаційної роботи. Пояснювальна записка кваліфікаційної роботи складається зі вступу, 3 розділів, висновків, списку використаних джерел (50 найменувань), 5 додатків. Кваліфікаційна робота містить 4 таблиці, 47 рисунків , викладена на 78 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНИЙ ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ РОЗРОБКИ ВЕБДОДАТКІВ

1.1 Аналіз сучасних вебтехнологій та прикладні аспекти їх застосування

З появою інтернету та служби WWW, які стали відправним пунктом розробки вебсайтів та спеціального програмного забезпечення для їх створення і обслуговування, з'явився і новий напрямок у діяльності програмістів – вебпрограмування. [1]. Розробка сайтів це ніша в програмуванні, яка весь час оновлюється, адаптується та покращується в цілому, і, наразі є актуальною, такою буде і надалі. Відтак, щоб бути сучасним розробником, потрібно постійно слідкувати за змінами і нововведеннями в технологіях.

Для сучасного бізнесу вебдодаток відіграє одну з ключових ролей. Не є перебільшенням переконання, що компанія без вебсайту є неповноцінною. Завдяки вебсайту компанія може: допомогти клієнтам в здійснені оплати, залучати клієнтів до взаємовигідної співпраці, підвищувати свій імідж, аналізувати онлайн-конверсію, визначати цільову аудиторію, формувати статистику та аналітичні звіти, рекламувати свою діяльність, товар або послугу. Розглядаючи важливість вебсайтів у бізнесі, важливо враховувати, що не існує універсального набору інструментів для їх створення. Різні категорії вебсайтів вимагають різних підходів до створення.

Через постійний розвиток та ускладнення архітектури вебсайтів, вимоги до швидкості, функціоналу та дизайну зростають з кожним роком. Автори публікації [2] зазначають, що майже 70 % споживачів, приймаючи рішення про здійснення покупки, враховують, наприклад, швидкість роботи сайту.

Очікується, що вебдодатки будуть доступні цілодобово з будь-якої точки світу та будуть адаптовані для використання з будь-якого пристрою з будь-яким розміром екрану. Окрім цього, важливими складовими сучасного вебдодатку є

гнучка і масштабована система безпеки та багатий досвід користувача, побудований на клієнті, вважають автори [3].

На сучасному етапі інтернет-розвитку існує велике розмаїття вебсайтів і вебдодатків, спрямованих на виконання різних завдань та задоволення потреб користувачів. Чимало аналітичних компаній обробляють величезні обсяги даних про поведінку користувачів інтернету, популярність та цільове використання вебсайтів. Мільйони людей у більш ніж 230 країнах світу щороку читають звіти та аналітику, включаючи співробітників багатьох провідних світових компаній, визнаних журналістів, ключових осіб, які приймають рішення в суверенних урядах, і провідних науковців. Наприклад, графік популярності вебсайтів різного призначення станом на жовтень 2024 р. зображено на рис. 1.1.

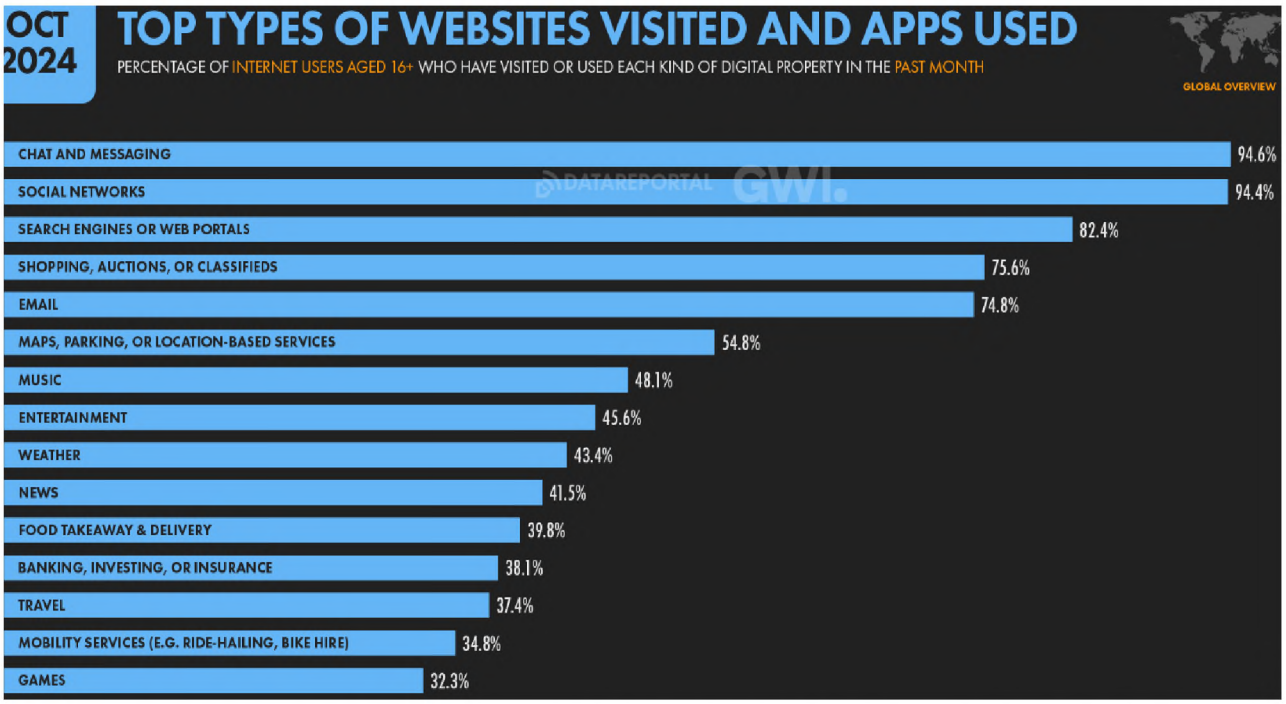


Рисунок 1.1 – Статистика відвідуваності різних типів вебсайтів (за матеріалами [4])

Як бачимо, найпопулярнішими типами сайтів є чати, соціальні мережі, пошукові двигуни та всеможливий e-Commerce: інтернет-магазини, аукціони, банкінг. Сайти можна класифікувати по-різному. Для дослідження саме в

розрізі використовуваного інструментарію, їх можна умовно поділити на категорії за такими показниками як: вид діяльності, характеристика, перелік типових програмних рішень. Узагальнення дослідження наведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика категорій вебсайтів

Категорія вебсайтів	Опис	Характеристика	Типовий інструментарій
Сайти-лендінги	Односторінкові вебсайти створені з метою реклами і маркетингу. Користувачі потрапляють на такі сайти через посилання в електронному повідомленні або рекламу в соц. мережах.	Простий дизайн, чітка ідея, максимально концентрований контент.	HTML та CSS, або можна використати генератор сайтів: Wix, Jekyll, Hugo.
Інформаційні сайти	Спрямовані на надання користувачам інформації різного характеру: новинні портали, сайти компаній, освітні ресурси ті інші.	Розгалуджена структура, акцент на текстовому і мультимедійному контенті, можливість взаємодії з користувачем через зворотній зв'язок.	CMS (Content Management Systems), або ж фреймворки: Django, Ruby on Rails, Express.js.
Соціальні мережі	Платформи для взаємодії та обміну інформацією Блоги фокусуються на індивідуальному контенті, соціальні мережі - на взаємодії, форуми – на обговореннях.	Складні за будовою сайти, наявна можливість публікації та обміну контентом, профілі користувачів, механізми коментування та «лайків».	Для створення платформ використовують технології, Facebook та Instagram фреймворк React.
e-Commerce	Інтернет-магазини, майданчики оголошень (по типу OLX), сайти зі знижками і промокодами, інтернет-аукціони, системи онлайн-платежів.	Каталог товарів з можливістю фільтрації та пошуку, корзина покупок та система онлайн-оплати, особисті кабінети.	Для сайтів такого типу зазвичай використовуються спеціалізовані CMS: Joomla, OpenCart.
Інтерфейси масштабних проєктів	Можуть бути корпоративні портали, системи управліннями великими даними, системи управління проєктами, банківські сайти, пошукові системи.	Складна архітектура з багатьма взаємодіючими модулями, забезпечення безпеки та управління доступом, формування аналітичних звітів.	Для розробки подібних вебсайтів використовуються потужні фреймворки: Angular, Nest, Laravel, Spring.
Інтерфейси хмарних обчислювальних систем	Вебінтерфейси для взаємодії з хмарними системами, наприклад: консолі управління хмарними платформами, такими як AWS Management Console чи Google Cloud Console.	Управління ресурсами хмарної інфраструктури, моніторинг та налаштування хмарних послуг, взаємодія з різними сервісами хмарної платформи.	Для таких систем, аналогічно з інтерфейсами масштабних проєктів найкраще використовувати потужні фреймворки.

Отже, як видно з табл. 1.1, кожна категорія вимагає відповідного підходу та підбору інструментарію для успішної реалізації.

Потрібно враховувати, що технології дуже швидко змінюються: те що було популярним вчора, сьогодні вже може бути не актуальним. Втрачають актуальність багатосторінкові сайти, адже швидкість та продуктивність таких сайтів є вкрай низькою. Це пов'язано з тим, що на кожну дію користувача запитується нова сторінка з серверу, та відбувається повне перезавантаження на клієнтській частині. На зміну їм прийшли односторінкові «сайти» - Single Page Applications (SPA), які отримують сторінку з сервера лише раз, а всі інші операції виконуються завдяки асинхронним AJAX та JavaScript скриптам [5].

Подібні зміни вимагали нових ідей та підходів до розробки. Наприклад, поява концепції «компонентів» у веброзробці дала змогу створювати елементи, які можна повторно використовувати в різних частинах коду. Окрім цього, стали з'являтися різні бібліотеки для управління станом додатку (Redux, MobX). Обов'язковою стала можливість інтеграції вебзастосунків з різними платформами і сервісами (соціальні мережі, платіжні системи, API сторонніх сервісів). Для створення real-time застосунків, таких як чати, було створено технологію websockets, яка дозволяє встановлювати постійне двостороннє з'єднання між користувачами, без необхідності постійно створювати нові «коннекти», як при використані традиційних методів [6].

Саме через необхідність постійного створення нового інноваційного інструментарію для реалізації подібних ідей, сформувалась та невпинно розвивається величезна екосистема вебтехнологій: спеціалізовані мови програмування, фреймворки, бібліотеки, бази даних та системи керування, інтегровані середовища розробки тощо.

Окремо важливим моментом є адаптивність. На сьогодні популярним є підхід «Mobile First». Цей принцип означає, що застосунки повинні розроблятися таким чином, щоб в першу чергу гарно виглядати на мобільних пристроях, а вже після цього на широких екранах. Чимало наукових публікацій, як в роботі [7], закликають завжди дотримуватися цього принципу розробки.

Сайт, створений за принципом «Mobile First» має наступні переваги: високе ранжування, швидке завантаження, сприяння здійсненню швидкої

покупки. До появи таких інструментів як CSS та JavaScript такий підхід, та й взагалі адаптивність вебсторінки була неможливою. Це важливо, оскільки якщо подивитись на статистику інтернет-трафіку за 2023-2024 рр., то можна побачити, що загалом більше, ніж 60 % трафіку припадає на користувачів мобільних пристроїв (рис. 1.2).

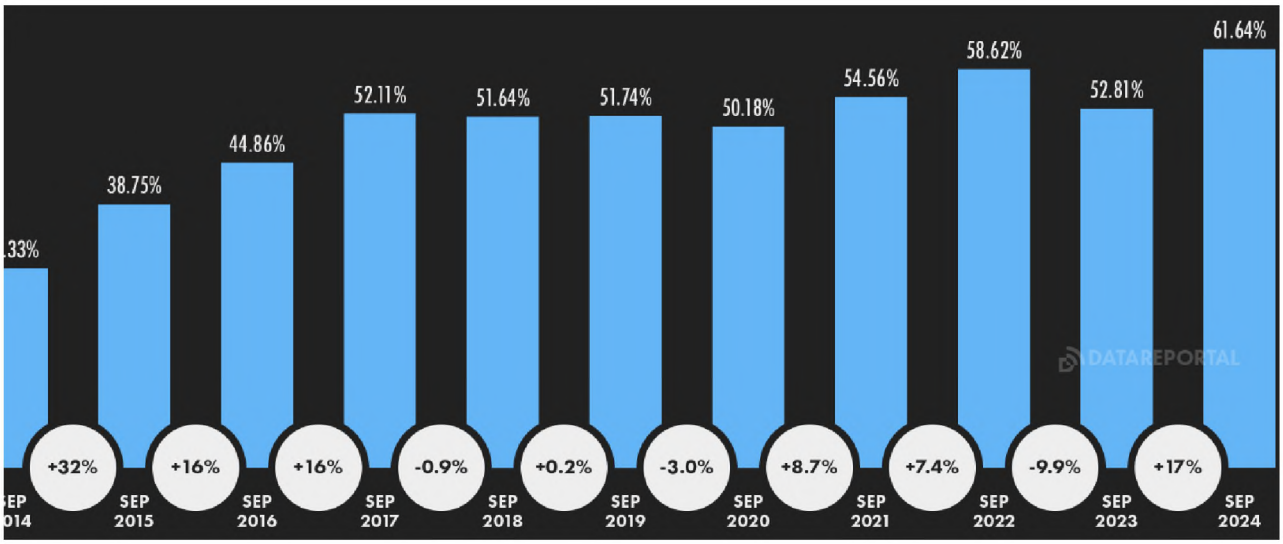


Рисунок 1.2 – Графік глобального використання інтернет-трафіку користувачами мобільних пристроїв (за матеріалами [8])

Крім того, розвиток мобільних технологій також призводить до необхідності впровадження в розробку специфічних фреймворків та інструментів, таких як React Native, Flutter або Xamarin. За допомогою цих інструментів розробники можуть створювати кросплатформені додатки, які працюють як в браузерах, так і на iOS та Android. Такий підхід дозволяє писати один код, який буде виконуватись на різних видах девайсів, що суттєво економить фінансові ресурси замовника та час розробника [9].

Враховуючи стрімкий розвиток галузі та постійну появу нових інноваційних інструментів, а також велику конкуренцію між вебстудіями, одним із головних завдань сучасного веброзробника є аналіз та підбір найкращого рішення саме для потреб його проєкту.

1.2 Характеристики базового інструментарію для розробки вебдодатків

Від появи та розвитку вебсайтів мережі інтернет, веброзробка базується на таких засобах, як мова гіпертекстової розмітки HTML, каскадних таблиць стилів CSS та мови програмування JavaScript [10]. При цьому JavaScript є інтегратором функцій сайту, керує його інтерактивністю та робить більш реактивним до дій користувача, дозволяє оновлювати зміст сайту залежно від запитів відвідувача. Однак, серед програмістів часто зустрічаються нарікання на незручність JavaScript, зясовується, яка ж мова програмування є кращою для веброзробки.

Б'ярн Страуструп, який розробив і впровадив C++, узагальнює думки про JavaScript у своєму вислові: «Є лише два види мов: ті, на які люди скаржаться, і ті, якими ніхто не користується». JavaScript, як і кілька інших мов (особливо PHP), протягом кількох років називали «поганою». Потім сталися кардинальні зміни. Завдяки поширенню Ajax, випуску кількох бібліотек, таких як Prototype, Moo Tools і jQuery, а також нових високо інтерактивних вебдодатків, розробники почали розуміти потенціал мови JavaScript (JS). Сьогодні JavaScript також є однією з найбільш поширених мов завдяки Node.js – платформі, яка дозволяє використовувати його як серверну мову, і PhoneGap, фреймворку для створення гібридних мобільних додатків.

На рис. 1.3 представлено результати дослідження відомої аналітичної компанії щодо популярності мов програмування за 2023-2024 рр. [11]. Згідно з опитуванням Stackoverflow, JS залишається найпопулярнішою мовою програмування одинадцятий рік поспіль. Для більшості розробників програмування – це вебпрограмування. Python помінявся місцями з SQL і став третьою за популярністю мовою. В опитуванні взяли участь 67053 респондентів, серед яких основну частину склали професійні девелопери. Такою популярністю пояснюється і така кількість фреймворків та бібліотек для

JavaScript. Саме вони значно впливають на швидкість розробки і додають величезну кількість нових можливостей.

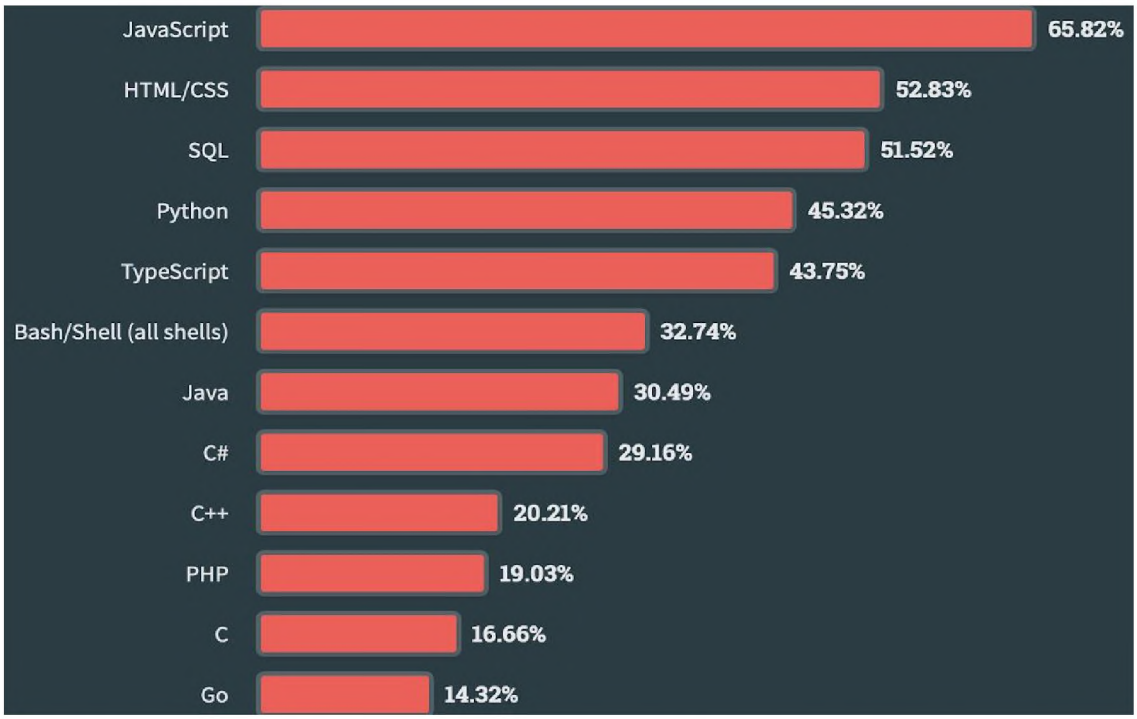


Рисунок 1.3 – Рейтинг мов програмування за популярністю у 2023 р. [11]

За спостереженнями аналітиків кілька технологій за минулий (2023) рік піднялися на одну позицію (Bash/Shell, C, Ruby, Perl і Erlang), а дві піднялися на дві позиції (Elixir і Lisp). Великим рушієм, який отримав сім позицій з 2022 р., стала Lua, вбудована мова сценаріїв.

Три найкращі технології для професійних розробників залишаються такими ж, як і минулого року: JavaScript, HTML/CSS і SQL. Але це інша картина для тих, хто вчиться кодувати. HTML/CSS і JavaScript майже однакові як найпопулярніші мови для людей, які навчаються програмувати. Студенти-розробники використовують Python частіше, ніж SQL (59% проти 37%), а професійні розробники повідомляють, що використовують SQL більше, ніж Python (52% проти 45%). Порівняно з професійними розробниками ті, хто вчиться програмувати, частіше повідомляють про використання Java (37% проти 31%), C++ (32% проти 20%) та C (32% проти 17%).

Професіонали, які повсякчас стикаються з величезною кількістю рутинних або повторюваних операцій, в першу чергу оцінили інноваційні підходи до побудови кодів, які надали бібліотеки та фреймворки.

Фреймворк включає в себе бібліотеки, але також він задає «каркас» майбутнього застосунку. Зазвичай фреймворк реалізує один з паттернів проєктування, таких як: MVC (Model-View- Controller), MVP (Model-View-Persistor), MVVW (Model-View-ViewModel). Тобто, саме фреймворк керує розробником, вказує йому як саме писати код. Деякі фреймворки дають більше свободи, деякі менше. Вони прискорюють та спрощують процес розробки, а також забезпечують більшу безпеку, ніж самописні рішення. В сучасних фреймворках враховані всі класичні інструменти злому [12].

З огляду на всесвітні тенденції серед професіональних розробників вже згаданий аналітик подає такий розподіл популярності за 2023 р., як на рис. 1.4.

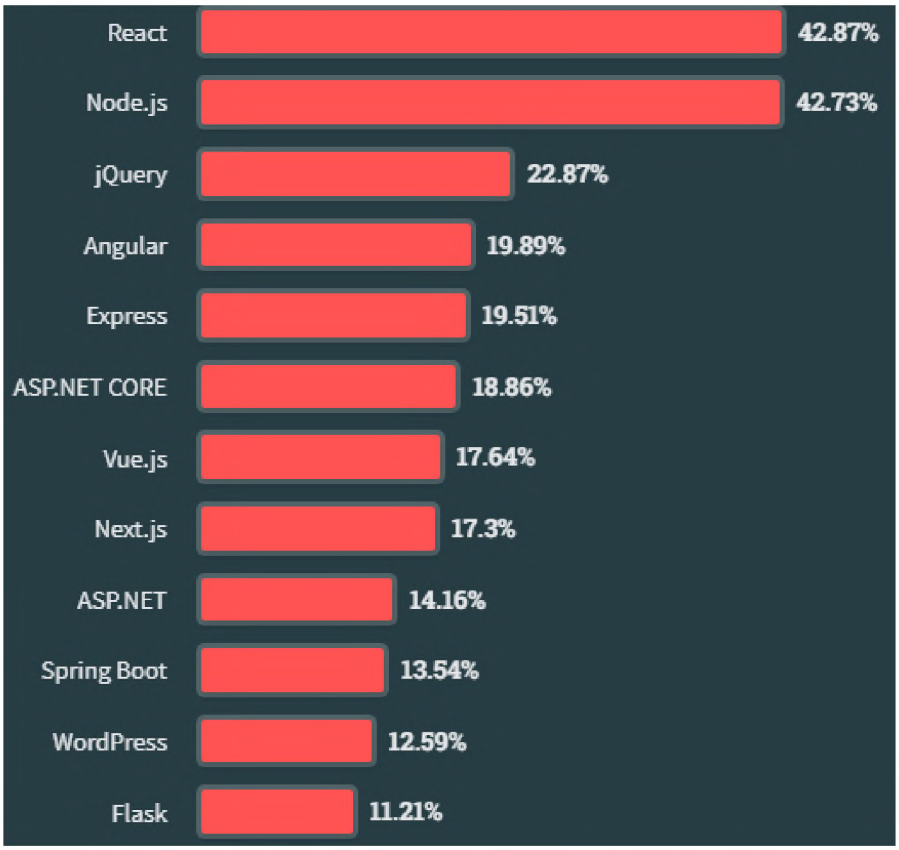


Рисунок 1.4 – Рейтинговий розподіл популярних фреймворків у 2023 р. [13]

Як слідує з висновків аналітичного дослідження StackOverflow [13], Node.js і React.js – дві найпоширеніші вебтехнології, якими користуються всі

респонденти. Професійні розробники використовують обидва досить однаково (див. рис. 1.4), а ті, хто вчиться кодувати, використовують Node.js більше, ніж React (52% проти 48%). jQuery та Express є наступними двома популярними вебтехнологіями для всіх респондентів, і jQuery частіше використовують професійні розробники, ніж ті, хто навчається коду (24% проти 18%), тоді як Express більше використовують ті, хто навчається, ніж професіонали (25% проти. 20%). Next.js перемістився з 11-го місця у 2022 р. на 6-е 2023 р., ймовірно, серед тих, хто вчиться кодувати.

Бібліотека – комплекс модулів, функцій та компонентів та іншого «готового» коду. По суті, це збірник коду, який можна повторно використовувати в своїх проєктах. Завдяки ним веброзробнику не потрібно писати з нуля такі речі як наприклад модальні вікна, каруселі зображень, слайдери і багато інших. Це значно спрощує процес розробки [12].

Варто звернути увагу, що окремі програмні засоби по великому рахунку є програмними бібліотеками, а не фреймворками. Між цими поняттями існує концептуальна відмінність. Як фреймворки, так і бібліотеки містять код, який написав інший розробник. Цей код використовують, коли вирішують однотипні завдання чи поширені проблеми. Принцип раціональної розробки програмного забезпечення (ПЗ) передбачає не дублювати код, який багато разів використовується. Користувачі (особливо комерційних додатків) зацікавлені у продуктивності, швидкості, популярності та співвідношенні ціни та якості, в той час як розробники прагнуть покращити свої рішення, щоб успішно конкурувати.

Загалом, різницю між бібліотекою та фреймворком дуже добре описує поняття «Inversion of Control». Це означає, що використовуючи бібліотеку ви контролюєте де і як ви викликаєте її функції. А у випадку з фреймворком саме він контролює де та яким чином вам розміщувати свій код, а також, коли його потрібно викликати [14].

Фреймворк це платформа для створення вебсайтів та програм, яка полегшує розробку завдяки великій кількості реалізованих функцій.

Програмістові не треба писати код із нуля. Він бере готове рішення і створює надбудову для реалізації продукту [15].

Бібліотеки – готові окремі компоненти, які вирішують певні завдання [15]. Наприклад, є бібліотеки для обробки файлів та виведення картинки на екран. За принципом дії: фреймворк звертається до додатку, який містить основний код, та реалізує в ньому певні дії. До бібліотеки ж, навпаки, звертається додаток і включає її готові елементи, поєднуючи із основним планом, проєктом.

Основна відмінність фреймворку від бібліотеки пояснюється тим, що фреймворк ніби задає жорсткі рамки. Розробник інтегрує свій код у стороннє рішення, але не може вийти за межі стандартної логіки. Бібліотеки ж можна або використовувати будь-якої миті, або відключити зовсім, якщо є альтернативи [14]. Ці відмінності бажано розуміти, щоб правильно використовувати обидві технології. Коли замовник просить створити сайт на базі готової CMS, PHP-фреймворки вже не знадобляться, а ось CSS можуть стати в нагоді. Можна підключити їх до проекту та не писати код стандартних функцій.

Обов'язковим для реалізації фронтенд-частини сайту є лише 1 засіб – HTML. Все інше, включаючи CSS, JavaScript та різні бібліотеки, фреймворки й препроцесори не є обов'язковим з точки зору браузера, але в сучасному світі без подібних інструментів не створюється жоден (окрім самих простих, коли розробник тільки навчається) вебдодаток у світі.

HTML (Hypertext Markup Language) – основа будь якого сайту, його каркас. За допомогою нього описується сама структура сайту, елементи які на ньому будуть знаходитися [16]. Препроцесори дозволяють розширити функціонал HTML при роботі з верстанням. Написаний за допомогою препроцесора код після інтерпретації сервером перетворюється в звичайний HTML. Прикладом таких препроцесорів може бути Pug або Haml.

До переваг використання HTML-препроцесорів відноситься:

– короткий, інформативний та чистий синтаксис, не потрібно слідкувати за закриттям тегів;

– дотримання принципу DRY;

– згенерована верстка є валідною (перевірено W3C – Markup Validation Service) [17].

Оскільки HTML дозволяє лише розміщувати елементи на сторінці, постає необхідність в інструменті, який буде вказувати браузеру як ці об'єкти виглядають. Рішенням цієї проблеми є CSS (Cascading Style Sheets). За допомогою таблиці стилів можна задавати, розмір, колір, положення одних елементів відносно других, поведінку курсора при наведенні та багато інших. Актуальною версію CSS є CSS3.

Для CSS, так само як і для HTML, існують препроцесори, а також фреймворки. Найпопулярнішими препроцесорами для CSS є Stylus, SCSS, LESS. Загальна ідея CSS-фреймворків – допомогти побудувати гарно структурований вебсайт з можливістю подальшого розвитку, а також додати додаткові функції для покращення процесу розробки стилів. Фреймворки дозволяють створювати сучасні вебінтерфейси, але також вони накладають свою специфіку та обмеження [18].

Рейтинг популярності CSS-фреймворків наведено в додатку А. З графіку видно, що найпопулярнішими рішеннями період 2019-2023 рр. є: Bootstrap, Tailwind CSS, MaterializeCSS, Foundation, Bulma, PureCSS, Ant Design [19].

1.3 Переваги використання популярних фреймворків та бібліотек JavaScript

Використання стандартного JavaScript хоч і цілком можливе, але часто розробники обирають якусь готову бібліотеку, або фреймворк. JavaScript-фреймворки визначають дизайн вебдодатків, викликають JS-бібліотеку та використовують код у ній для завершення програми. Ці фреймворки пропонують повну дорожню карту для створення вебзастосунків замість одного рішення. Розробники додали ці фреймворки у свій набір інструментів для

розробки вебрішень для своїх клієнтів. Перевага використання фреймворків полягає в тому, що можна створювати безліч вебсторінок, використовуючи їхні багаторазово використовувані компоненти. На теренах інтернету ведуться дискусії щодо рейтингів та затребуваності різних бібліотек, аналітичні компанії складають регулярні звіти (рис. 1.3). Наприклад, серед топових бібліотек і фреймворків станом на початок 2024 р. в топі для розробки сайтів були названі наступні спеціалізовані засоби [20].

React – це фреймворк з відкритим кодом, що належить Facebook (Meta). React – це вільний і декларативний фреймворк, який широко використовується для розробки інтуїтивних інтерфейсів для вебдодатків. Його можна використовувати для більш швидкого створення додатків, що масштабуються. Крім того, React має невеликі пакети, тому його легко освоїти новачкам. React можна використовувати для розробки великомасштабних вебзастосунків, що використовують різні дані [21].

Svelte – це JavaScript-фреймворк з відкритим вихідним кодом, який допомагає веброзробникам перетворити необроблений код статичних вебдодатків на інтуїтивно зрозумілі та інтерактивні інтерфейси користувача. Цей фреймворк, створений Річем Харрісом, збагатив екосистему JavaScript. Як і React, Svelte – це більше, ніж фреймворк; він працює як компілятор, що перетворює код Svelte на ванільний JavaScript для отримання більш швидких вебрішень, ніж Vue.JS і React. Особливості Svelte розглянуто нижче [22].

1. Робота без віртуального DOM. На відміну від React, Svelte працює з кодом без Document Object Module (DOM) і переносить більшу частину коду на компіляцію, досягаючи більш швидких результатів, ніж інші фреймворки.

2. Забезпечує реактивність. Svelte перетворює модулі на функції DOM, які реагують на зміни даних та відображають модифікації у вигляді коду

3. Вимагає мінімального коду від веброзробника. Фреймворк svelte вимагає мінімальної кількості коду, ніж React та Vue, що допомагає веброзробникам витрачати більше часу на інші завдання, крім кодування.

4. Забезпечує модульну стилізацію вебсторінок. Svelte забезпечує послідовний дизайн усіх вебсторінок додатків за рахунок додавання унікальних стилів та створення унікальних імен для класів.

Фреймворк Vue був розроблений у 2014 р. Еваном Ю. та використав ліцензію MIT. Еван є колишнім співробітником Google і досі допомагає команді підтримувати свій продукт. Це фреймворк з відкритим вихідним кодом, який допомагає створювати інтуїтивно зрозумілі інтерфейси користувача, поєднує популярні функції Angular і React. Він використовує шаблони Angular та прив'язку даних, а також реквізити React.

Фреймворк Vue має відмінну особливість від інших фреймворків, оскільки його можна поступово переймати для створення програм [23]. До інкрементів входять рішення для маршрутизації, інструментарій, командне введення (CLI) та управління станом. Цей фреймворк пропонує модулі, що інкрементально впроваджуються, тому ви можете кодом додавати нові інкременти у свій вебпроект. Архітектура цього фреймворку проста у використанні, і не потрібно розумітися на всіх функціях. Варто відзначити, що Vue – це легкий фреймворк розміром 23 Кб. Vue вимагає великих знань JavaScript, HTML і CSS для створення масштабних вебдодатків та об'єднання їх у нові або вже існуючі програми за допомогою JSX. Основні особливості предствлені нижче.

1. Зв'язування DOM з об'єктними даними за допомогою HTML-шаблонів. Фреймворк Vue надає HTML-шаблони для зв'язування DOM із даними об'єктів. Більше того, цей фреймворк запускає компіляцію шаблонів у вигляді HTML, сумісного з усіма браузерами.

2. Забезпечення інтерактивності. Vue дозволяє додавати ефекти переходу, щоб пожвавити вебсторінки. Більше того, ви можете додати сторонні бібліотеки анімації для більшої інтерактивності вебзастосунків.

3. Дозволяє користувачам швидше переміщатися сайтом. Маршрутизатор Vue перемикається між сторінками без частого оновлення вебсторінок і прискорює навігацію для користувачів.

4. Надають директиви для інтеграції коду. Директиви – це набір інструкцій для інтеграції коду екземплярів Vue. Ці візуальні засоби забезпечують інтерактивність вашої програми для покращення користувальницького досвіду.

Angular був створений компанією Google в 2010 р., є фреймворком з відкритим вихідним кодом, написаним мовою Typescript. Це фреймворк, що базується на інкрементах, який пропонує безліч інтегрованих бібліотек та інструментів для створення, тестування та підтримки коду вебдодатків. Цей фреймворк є надійним варіантом для веброзробників, що дозволяє створювати інтерактивні програми за допомогою коду JavaScript [24].

В Angular framework інтегровані інтерактивні шаблони та інструменти наскрізної інтеграції для пом'якшення проблем у вебпроектах. Цей фреймворк можна використовувати для створення одноразових вебзастосунків. У зв'язку з великою популярністю існує багато версій фреймворку Angular. Він є найкращим варіантом для розробки програм корпоративного рівня шляхом інтеграції складних функцій. Іноді Angular плутають з AngularJS, але між ними є різниця. Angular сумісний з усіма мобільними браузерами, тоді як версія AngularJS підтримує лише настільні браузери.

У фреймворків є переваги і недоліки, тому їх не можна назвати ідеальним інструментом. При правильному використанні вони економлять час, але, якщо у розробника недостатньо досвіду роботи, то у взаємодії з новим фреймворком виникає багато проблем. Завдання, які вирішують фреймворки:

1. Покращують швидкість розробки. У програміста буде відкладене «ядро», яке можна використовувати як основу проекту.

2. Зменшують вартість задач. Якщо програмісту не хочеться створювати сайт з нуля, а можна використовувати фреймворк, то сайт може коштувати дешевше. Самописні CMS можуть розроблятися кілька років, а бюджет часто перевищує розумні значення.

3. Звільняють від рутинних завдань. Розробник може займатися реалізацією нестандартних функцій.

4. Допомагають залишитися конкурентоспроможним на ринку. Якщо програміст в досконалості освоїв кілька популярних фреймворків, він не залишиться без роботи.

Без сумніву, JavaScript-фреймворки популярні, але є завдання, в яких їх використання не обов'язково. Припустимо, розробка вебпрограми проста і не вимагає складних функцій. У таких випадках використання фреймворку не рекомендується. Фреймворки пропонують безліч функцій, але вони мають свою цінову політику. Ці фреймворки, безсумнівно, забезпечують більш декларативний код, але в результаті ви можете отримати додаток з вищими обчислювальними витратами, ніж звичайний JavaScript.

Графік популярності базований на кількості питань, що стосуються відповідного фреймворку на Stack Overflow наведено на рис. 1.5.

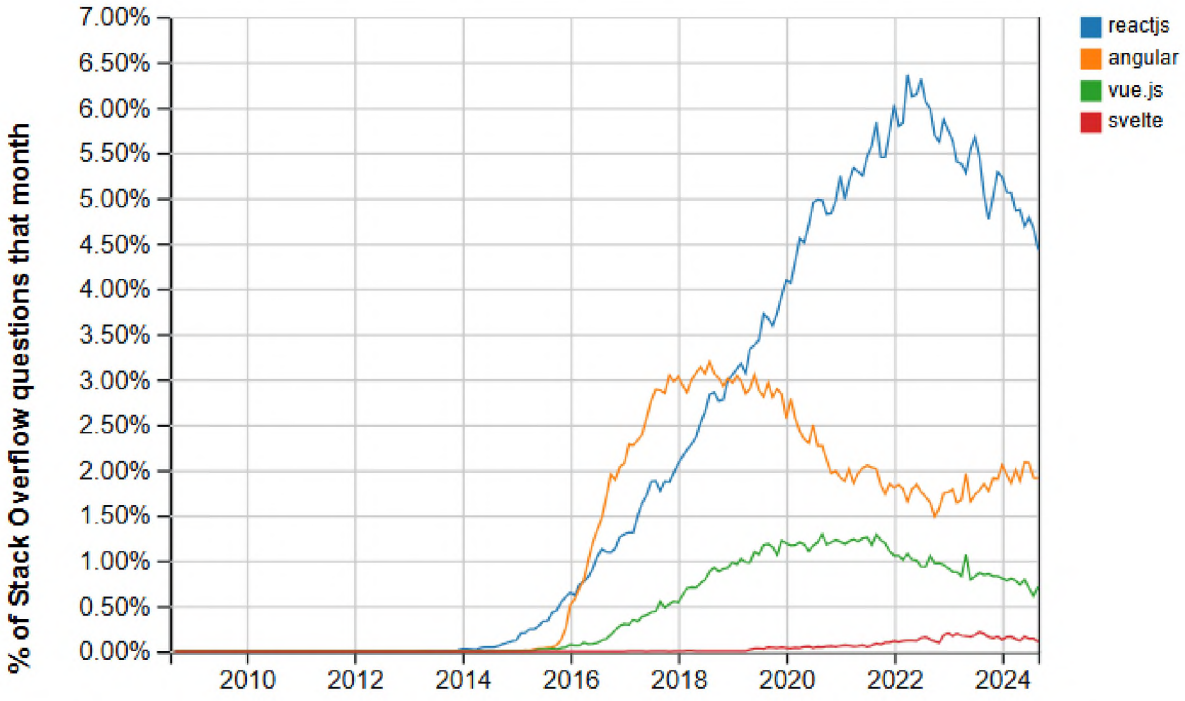


Рисунок 1.5 – Графік популярності фреймворків на Stack Overflow станом на 14.05.2024 (за матеріалами [25])

На графіку видно, що найпопулярнішим фреймворком з великим відривом від інших є React, хоча за останній рік він дещо погіршив свої показники. Angular, який був неймовірно популярним в 2016-2018р. з тих пір втратив значну частину користувачів, але залишається стабільна база

розробників, які надають перевагу саме цьому фреймворку. Vue зберігає свою відносно велику аудиторію. Новий Svelte ще тільки набирає популярності, але деякі розробники вже вважають його «вбивцею» інших фреймворків.

Вибір веброзробниками бібліотек є досить суб'єктивним та розрізняється підходами до реалізації при розробці, але всі інструменти для самостійної розробки сайту мають конкурентоспроможні ресурси.

Для самостійного відстеження популярності бібліотек та фреймворків можна використовувати порівняльні можливості Google Trends. Він показує динаміку попиту з урахуванням кількості пошукових запитів [26]. Можна ввести кілька ключів відразу, щоб відобразити дані на одному графіку та порівняти їх, наприклад, на рис. 1.6 представлені графіки популярності пошукових запитів в Україні по трьом популярним бібліотекам: React, jQuery та Svelte.

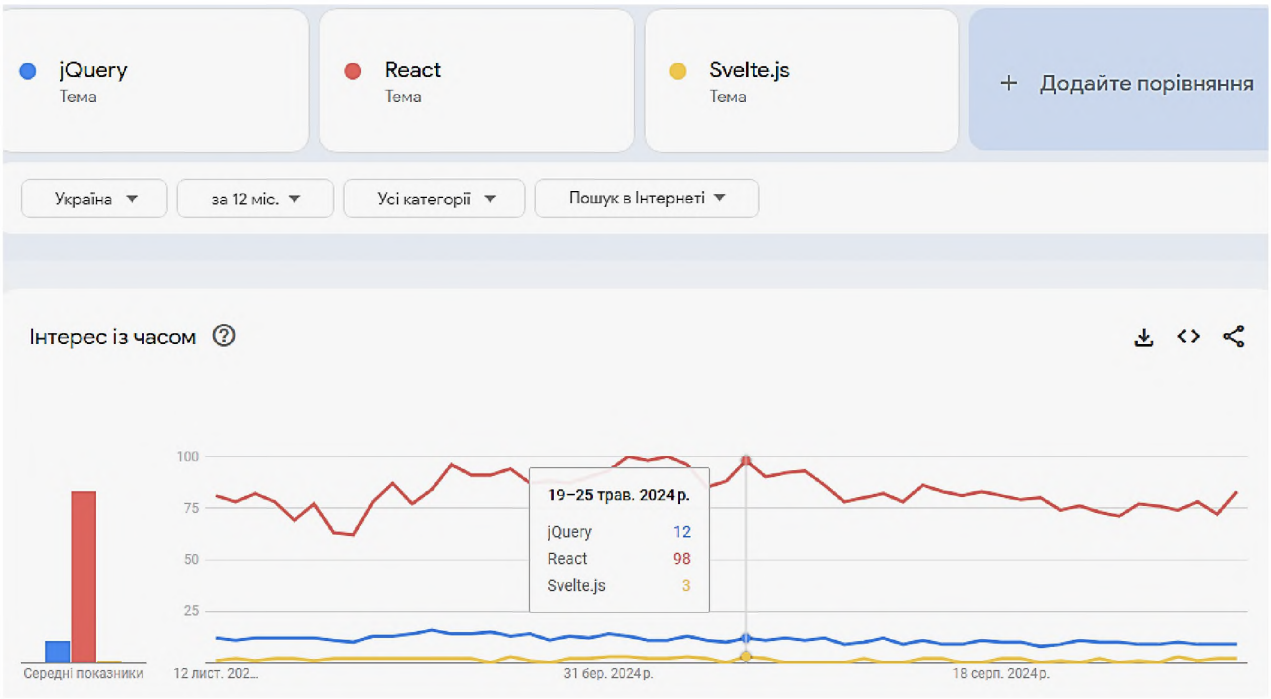


Рисунок 1.6 – Динаміка запитів Google про React, jQuery та Angular за 2024 р.

За станом графіків (див. рис. 1.6) видно, що інформацію про React шукають приблизно в 8 разів частіше, ніж про jQuery, а запити на інноваційну Svelte взагалі були на нижній межі протягом 11 місяців 2024 р. Розглянемо детальніше особливості вже добре апробованої бібліотеки jQuery.

jQuery – це відома безкоштовна, ліцензована відповідно до ліцензії Массачусетського технологічного інституту (MTI) популярна бібліотека JavaScript, створена Джоном Резігом у 2006 р., призначена для спрощення сценаріїв HTML на стороні клієнта [Г11]. Зазначається, що jQuery це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона значно спрощує такі речі, як обхід HTML-документів і маніпуляції з ними, обробка подій, анімація та Ajax завдяки простому у використанні API, який працює в багатьох браузерах [27]. Завдяки поєднанню універсальності та розширюваності jQuery змінив спосіб написання JavaScript мільйонами людей. Використання настільки поширене, що, згідно зі статистикою BuiltWith (станом на квітень 2015 р.), jQuery використовувалася на 63% мільйона найкращих вебсайтів [28]. Останні результати показані на рис. 1.7.

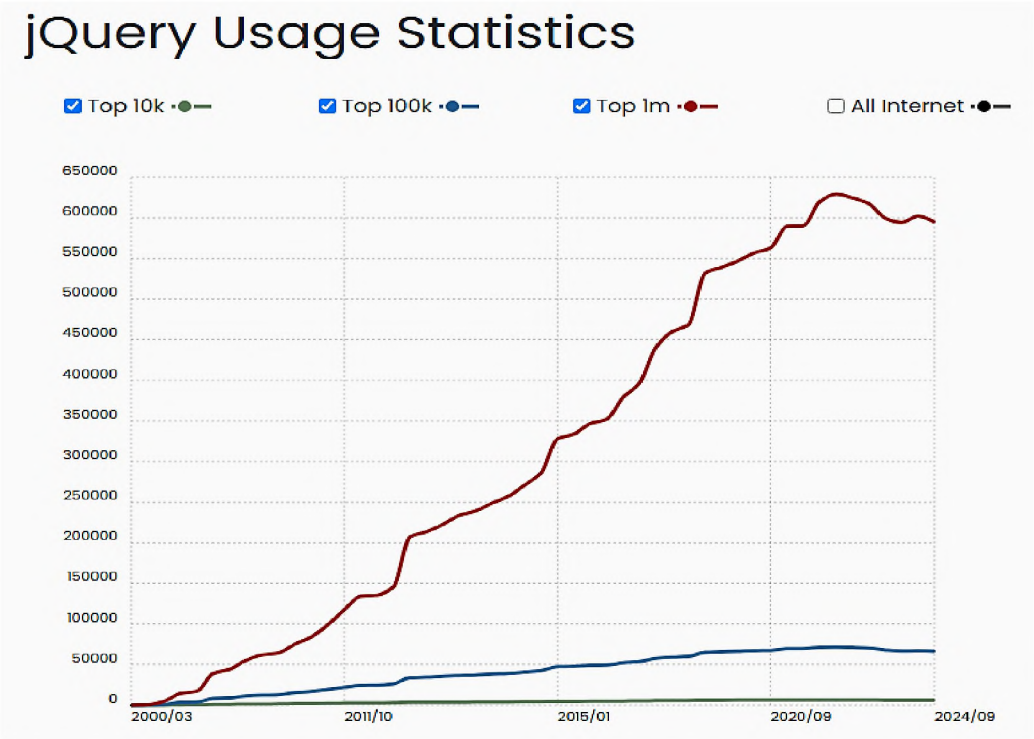


Рисунок 1.7 – Тенденції використання jQuery в світі на топ 10 тис., 100 тис, 1 млн вебсайтів

При цьому, серед українських розробок 227990 вебсайтів включали jQuery [28]. Загалом за 20 років популярність цієї бібліотеки постійно зростала.

1.4 Інструменти Backend-розробки

Клієнтська частина вебдодатку спілкується з серверною частиною за допомогою HTTP-запитів. В цих запитах міститься посилання на endpoint (URL-адреса на сервері, яка може приймати запити), тип запиту, та, за необхідності, додаткова корисна інформація – payload. Після обробки запиту сервер повертає відповідь клієнту, яка містить в собі інформацію, що передбачена розробником. Сервер (як backend частина вебдодатку) повинен виконувати наступні функції:

- обробка HTTP-запитів;
- обробка бізнес-логіки;
- взаємодія з базою даних;
- авторизація та аутентифікація;
- логування.

Вебсервер зберігає та надає доступ до контенту вебсайту – зображень, тексту, відео. Графічне пояснення класичної взаємодії клієнта (frontend) та сервера (backend) наведено на рис. 1.8.

Рисунок 1.8 – Взаємодія клієнта та сервера

Хоча вебсервери, зазвичай, розміщують вебсайти, доступні в інтернеті, вони також можуть використовуватися для зв'язку між вебклієнтами та серверами в локальних мережах, таких як Intranet-компанії. Вебсервери здатні обмежувати швидкість відповіді окремим клієнтам, щоб уникнути перевантаження ресурсів та обробити запити більшої кількості користувачів [29].

Після CERN httpd (першого вебсервера) Тіма Бернерса-Лі у перші пару років існування інтернету, Apache, який вперше випущений у 1995 р., швидко завоював ринок і став найпопулярнішим вебсервером у світі. Зараз він все ще займає цю ринкову позицію, але в основному через причини спадщини. На сьогодні конкурентом Apache є Nginx. «Nginx було створено спеціально для усунення обмежень продуктивності вебсерверів Apache» - за словами розробника. Однак, результати дослідження [30] показали, що в більшості тестувань Apache виявився швидшим за Nginx, з чого можна зробити висновок, що Apache краще підходить для вебсайтів з великою кількістю запитів в хвилину. На рис. 1.9 зображена статистика використання вебдодатками різних вебсерверів. Ці дані стосуються загальної кількості вебсерверів у всьому світі, але якщо ми візьмемо вибірку з мільйона найкращих вебсайтів, Nginx вже деякий час перебуває в лідерах. Google Search Trends, здається, також відображає цей факт (рис. 1.10).

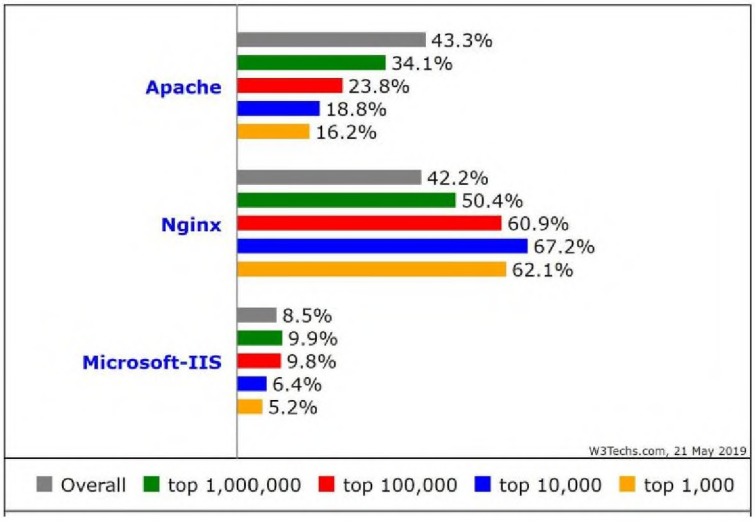


Рисунок 1.9 – Статистика використання вебсерверів (за матеріалами [31])

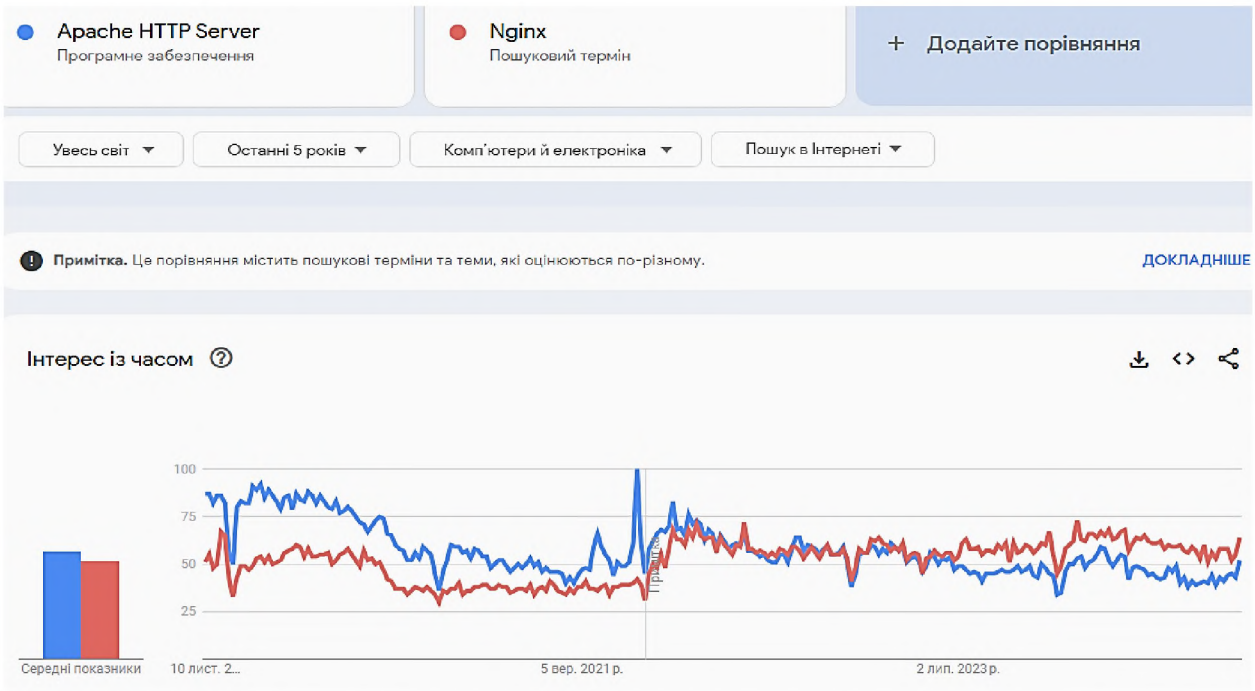


Рисунок 1.10 – Тенденції пошуку Google: Nginx проти Apache, 2024-2024 рр.

Backend-частину вебсайту зазвичай пишуть на таких мовах програмування як Java, Python, Ruby, Node.js, PHP. Проте, зараз майже ніхто не використовує мову програмування саму по собі. Переважна більшість backend-розробників в своїй роботі так само використовують фреймворки. Найпопулярніші backend-фреймворки станом на кінець 2023 р. наведено на рис. 1.11.

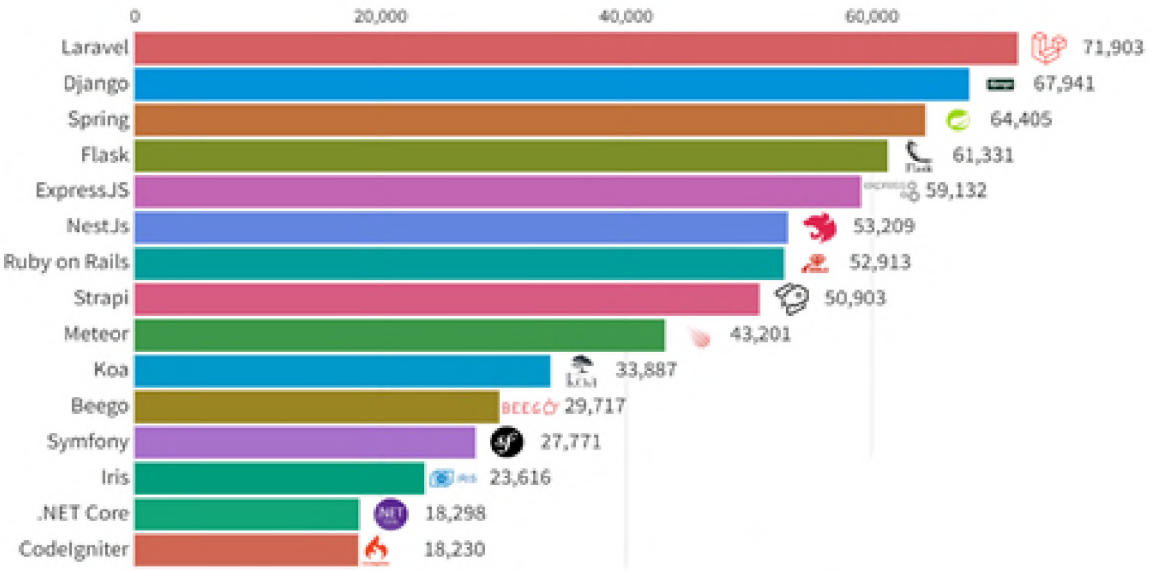


Рисунок 1.11 – Діаграма популярності backend фреймворків (за [31])

На діаграмі видно, що найбільш популярними фреймворками є: Laravel (PHP), Django (Python), Spring (Java) та ExpressJS (Node.js).

Окрім мов програмування, важливою складовою серверу є бази даних (БД). База даних – це організована структура, призначена для зберігання, зміни й обробки взаємопов'язаної інформації. БД бувають: централізовані, хмарні, реляційні, нереляційні, об'єктно-орієнтовані, операційні та розподілені. Найпопулярнішими є реляційні, нереляційні (NoSQL) та об'єктно-орієнтовані БД.

В реляційних БД інформація зберігається у вигляді відношень, пов'язаних між собою. Перевага реляційних баз – ACID (Atomicity, Consistency, Isolation, Durability). Цей принцип гарантує надійність та цілісність даних в процесі їх обробки. В нереляційних БД можливо сховище виду «ключ-значення», документоорінєтоване сховище, графові бази та bigtable-подібні бази. В таких базах немає чітких зв'язків між даними та немає чіткої структури. Перевагою NoSQL є швидкість та продуктивність [32].

В об'єктно-орієнтованих БД інформація представлена у вигляді об'єктів, як в об'єктно-орієнтованих мовах програмування. Перевагою таких баз є чітка типізація та відсутність неспівпадіння моделі даних в застосунку та БД.

Для звернення до реляційних та об'єктно-орієнтованих баз даних використовуються SQL-запити – спеціальні конструкції написані за допомогою мови SQL. При використанні NoSQL баз форма запитів відрізняється в залежності від обраної БД, наприклад в MongoDB використовується мова запитів, подібна до JavaScript – MongoDB Query Language.

Отже, реляційні БД ідеально підходять для роботи з даними, структура яких не вимагає частих змін, нереляційні – для зберігання великих об'ємів неструктурованої інформації, а об'єктно-орієнтовані БД – для проєктів, де потрібна потужна база з високою функціональністю. Рейтинг популярності баз даних наведено на рис. 1.12.

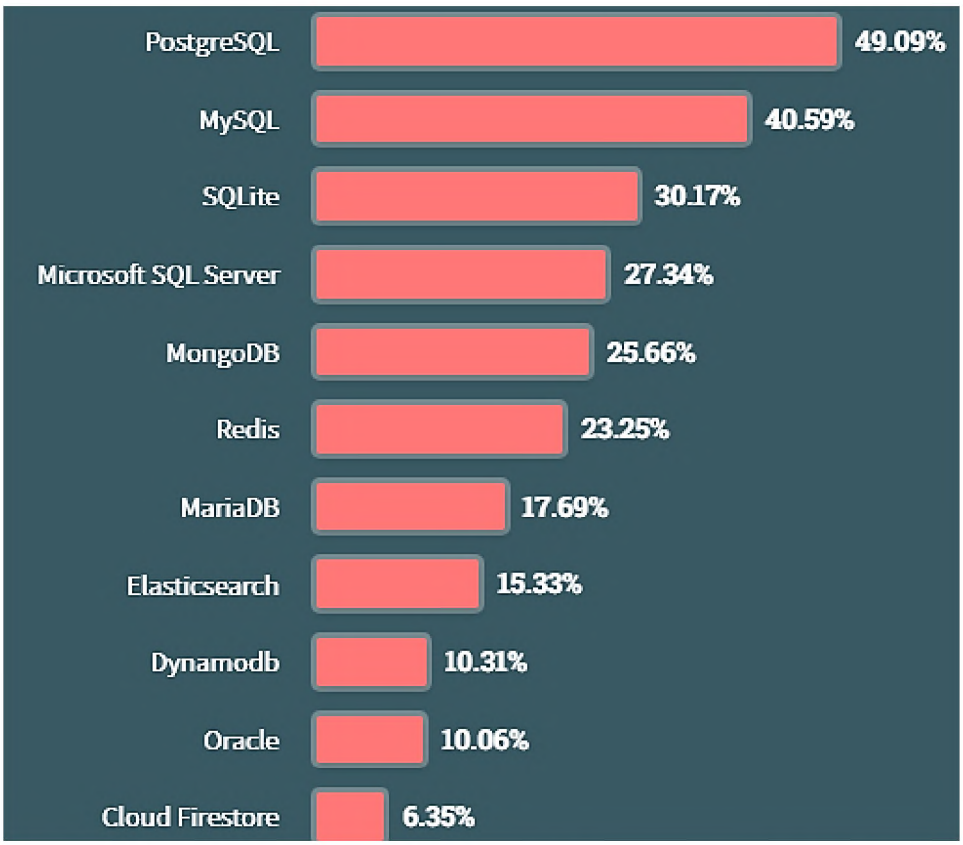


Рисунок 1.12 – Діаграма популярності баз даних (за матеріалами [33])

Отже, є мови програмування, є бази даних, системи управління базами даних, але це не все. Серверну частину потрібно тестувати. Для вирішення таких питань існує сучасне рішення – інструменти тестування та налагодження API, або REST-клієнти. Найпопулярнішими представниками даного класу інструментів є Postman та Insomnia.

До функціоналу подібних інструментів входить:

– Відправлення HTTP-запитів. REST-клієнти дозволяють легко створювати та відправляти будь-які типи HTTP-запитів. Розробник може налаштовувати параметри запитів, передавати заголовки та тіло запиту;

– Тестування API. Присутня можливість створення тестів для перевірки відповідей від серверу;

– Колекції. REST-клієнт дозволяє організовувати запити і тести в колекції, що спрощує управління великою кількістю запитів. Це дозволяє автоматизувати тести, а також перемикатись між різними конфігураціями середовища (тестове, розробка, продакшн) [34].

Окрім цього існують інструменти для автоматичної генерації документації API. Найвідомішим інструментом такого призначення є Swagger. Завдяки формату OpenAPI він може детально визначити структуру API, його ресурси, ендпоінти та параметри. Це створює чітку та зрозумілу основу для взаємодії з API, що полегшує розуміння його можливостей та використання.

Висновки до розділу 1

Нами було проаналізовано сучасний стан вебтехнологій та тенденції їх розвитку, на підставі чого ми можемо стверджувати, що наразі відбувається стрімкий розвиток вимог та інструментів для розробки вебдодатків, а також, що вебсайти є невід'ємною частиною існування сучасного бізнесу. Було проведено аналіз особливостей різних вебсайтів, та на основі цього створено таблицю, в якій вони були поділені за категоріями: за характеристиками, призначенням та оптимальним стеком технологій. Це потрібно тому що, не існує універсального інструменту для кожної задачі.

Було розглянуто особливості використання різних інструментів сучасного веброзробника на різних етапах та структурних частинах вебдодаків клієнт-серверної архітектури, зокрема, визнані мови програмування та їхній рейтинг; фреймворки мови JS та рейтингові позиції, бібліотеки JS. Особливий акцент зроблено на виявленні різниці між фреймворками та бібліотеками, розкрито причини популярності відомих фреймворків та необхідності їхнього застосування. Особливо корисними є відібрані та представлені в розділі різні графіки популярності та порівняльні таблиці відповідних технологій, на які можна опиратися під час вибору інструментарію.

РОЗДІЛ 2
ОСОБЛИВОСТІ ЗАСТОСУВАННЯ ПРОГРАМНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ДЛЯ РОЗРОБКИ КОМЕРЦІЙНИХ ВЕБСАЙТІВ

2.1 Аналіз та обґрунтування вимог вибору інтегрованого середовища для розробки коду вебдодатку

Прикладною частиною кваліфікаційної роботи є проєктування та розробка комерційного вебдодатку з використанням стеку сучасних програмних звсобів, ефективність яких доведена, зокрема, рейтингами популярності серед професіональних девелоперів. Використовуючи теоретичний огляд мов програмування, результативності та технічних характеристик фреймворків, бібліотек JavaScript, було вирішено обрати саме поєднання традиційних технологій HTML&CSS з фреймворком React, бібліотекою JQuery та інші.

Програмісти багато часу проводять за написанням та налагодженням коду. Робота з кодом потребує також спеціалізованого середовища розробки. В сучасному світі з'явилася величезна кількість різноманітних редакторів та довершених інтегрованих середовищ розробки кодів (IDE).

На сьогоднішній день найбільш популярними та потужними з них вважаються Visual Studio Code, Sublime Text, Atom, WebStorm [35]. Сучасні редактори коду вміють підсвічувати синтаксис на будь-якій мові програмування, виставляти автоматичні відступи, доповнювати код, скорочуючи процес розробки, розділятись на кілька робочих областей, встановлювати велику кількість розширень, налаштовувати зовнішній вид та інструменти, мають консоль, дебагер та систему контролю версій, як от Git. Розглянемо детальніше характеристики порівнюваних застосунків по пунктам.

1. Visual Studio Code – це легкий, але потужний редактор вихідного коду, який працює на робочому столі та доступний для Windows, macOS та Linux. Він поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js, а також

має багату екосистему розширень для інших мов (наприклад, C++, C#, Java,

Visual Studio Code призначений для проєктів різноманітної складності, розробників будь-якого рівня, при цьому дуже гнучко налаштовується під будь-які запити розробника, незалежно від напрямку його діяльності. Має зручний інтерфейс на будь-який смак з можливістю вибору тем, від світлої до самої темної, систему контролю версій, дебагер, автоматичний пошук помилок в коді. IntelliSense – автодоповнювач коду у зв'язці з Emmet значно пришвидшує процес розробки, іноді достатньо написати лише 1 символ і редактор вже запропонує вам потрібну підказку.

2. Sublime Text 3 – це кроссплатформений текстовий редактор, розроблений для верстальщиків та програмістів і дозволяє використовувати від Erlang до C-подібних мов програмування. Він досить мінімалістичний, в редакторі відсутні панель інстументів та діалогові вікна. Але при детальному перегляді у ньому можна знайти плагіни, розумне автодоповнення коду, виділення багатьох елементів відразу, що дозволяє швидко замінювати назви змінних чи файлів, функція полегшення доступу до файлів та багато інших корисних функцій. Sublime Text досить простий в засвоєнні і підійде програмісту з будь-якими задачами. Виділяють його «легкість» і швидкість, адже він не нагромаджений зайвими функціями, доки вони вам не знадобляться.

Даний редактор має відкритий вихідний код і надає як безкоштовнітак і платні версії. Безкоштовної версії достатньо для легкої та комфортної розробки.

Серед недоліків можна виділити хіба що баги, які іноді виникають з певними плагінами, тому варто уважно підходити до їх вибору [36].

3. WebStorm – це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях. Як і інші IDE JetBrains, WebStorm дозволяє автоматизувати рутинну роботу та легко справлятися зі складними завданнями, роблячи розробку більш цікавою [37]. Він, як і всі сучасні IDE, забезпечує автодоповнення коду, його аналіз прямо по ходу розробки, навігацію по коду, інтеграцію з системами контролю версій, такими як Git та рефакторинг коду.

Рефакторинг реалізований якісно і надає змогу змінювати код JavaScript і HTML в різних файлах та папках проекту, що значно полегшує налагодження роботи додатку, що розробляється.

У WebStorm інстальовані більше 100 різних доповнень, які забезпечують легку підтримку і зручну розробку завдяки використанню фреймворків, бібліотек і плагінів. Заслугою даного редактора коду є так званий Live Edit, що відразу інстальований в ньому. Він дозволяє при збереженні файлу відразу бачити внесені зміни, що досить вагомо полегшує розробку і зменшує кількість непотрібних дій розробника.

Звичайно, редактор підтримує усі сучасні мови програмування останніх версій, такі як Node.js, Maria.db та інші. Дана IDE є взірцем якісного продукту, але при цьому є досить навантаженою, тобто «важкою» і за доступ до більшості функцій потрібно буде купити платну підписку.

aver – візуальний HTML-редактор Adobe. Редактор не користувався особливою популярністю у розробників, але в оновленій версії з'явилося безліч додаткових можливостей, таких як сучасний користувальницький інтерфейс і гнучкий механізм для швидкого написання коду. Ці функції спрощують роботу вебдизайнерів і розробників інтерфейсу користувача, дозволяючи створювати проекти, писати код і керувати веб-сайтами, які чудово виглядають на будь-якому екрані [38]. Серед явних плюсів редактора можна виділити реалізацію власного «Live Edit», як в IDE WebStorm, що надає змогу розробляти в режимі реального часу і бачити зміни безпосередньо без оновлення сторінки браузера. Також присутні синтаксичні підказки, перевірка помилок, а також попередньою обробкою CSS-коду.

є платним, як і інші додатки компанії Adobe, входить в пакети, що купуються разом з іншими, іноді не потрібними вам функціями. Конкретно ця IDE входить до пакету Adobe Creative Cloud, що також включає в себе Photoshop, Adobe Stock та інші програмні рішення;

5. Vim – це спеціалізований текстовий редактор програмістів. Він призначений для роботи з великими обсягами коду без миші [37]. Саме

вилучення миші з процесу розробки є його головною перевагою, адже для того, щоб зробити якусь дію в проєкті не потрібно відривати руки від клавіатури, що значно пошвидшує швидкість розробки. Для того, щоб, наприклад, видалити рядок коду, в Vim можна це зробити без миші просто за домопогою певної комбінації клавіш. Звичайно, цей редактор коду більше підходить розробникам на Java або С-подібних мовах, але може бути корисний для верстальщика.

Після звикання робота в Vim можна порівняти з грою на гітарі: ви керуєте текстом, ніби він продовження вашої руки…[37]».

Серед недоліків звичайно можна виділити важкість в освоєнні, відсутня функція автодоповнення коду, що може призвести до синтаксичних помилок у новачків. Саме тому дане рішення підходить лише для досвідчених розробників, які досконально знають синтаксис і можуть передбачити помилки, що можуть виникнути в ході розробки продукту.

Узагальнюючи характеристики, серед основних переваг названих IDE можна виділити наступні:

1. безкоштовність; з малої літери всі пункти;
2. автодоповнення коду;
3. можливість редагування в режимі реального часу або, так званий,
4. зручність;
5. простота інтерфейсу;
6. гнучкість;
7. простота освоєння;
8. практичність.

Є доцільним виділити основні переваги цих застосунків та відібрати їх для порівняння за допомогою оцінки їх балами за результатами статистики відгуків та проведеного якісного аналізу. Далі заносимо оцінки виділених характеристик редакторів коду в табл. 2.1 і визначаємо бали за шкалою від 1 до 10, де 1 – абсолютно не задовільно, а 10 – повністю відповідає на основі досконалості даних переваг у редактора.

Таблиця 2.1 – Порівняльна характеристика популярних IDE

Критерії порівняння	Бальна оцінка за критеріями кожного із середовищ розробки				
	VS Code	Sublime Text	WebStorm	Dreamweaver	Vim
Безкоштовність	10	9	5	5	10
Автодоповнення коду	10	9	10	9	5
Live Edit	10	10	10	9	9
Зручність	10	8	10	8	10
Простота інтерфейсу	8	8	7	8	6
Гнучкість	10	8	10	7	8
Простота освоєння	9	9	9	9	4
Практичність	9	8	10	8	10
Середній бал якості	9.5	8.63	8.88	7.88	7.75

Отже, після розрахунку середнього балу якості найбільше переваг має редактор коду Visual Studio Code (результати – 9,5) через свою зручність, гнучкість та безкоштовність.

Використання редактора коду Visual Studio Code (VS Code) для роботи з React є не обов'язковим, але дуже популярним і практичним вибором через його функціональність та підтримку сучасних технологій веброзробки. Серед основних причин, чому VS Code часто використовується для роботи з React, який був обраний для роботи, можна окреслити наступні.

1. Зручність і швидкість роботи:

– VS Code легкий, швидко завантажується і працює навіть на середніх за потужністю комп'ютерах;

– має простий і інтуїтивний інтерфейс, що дозволяє легко налаштовувати середовище під конкретний проект.

2. Наявність плагінів і розширень, які полегшують роботу з React, а саме:

– швидкі вставки коду, що економлять час.

або ESLint – автоматичне форматування коду та перевірка на помилки.

– розширення для глибокого аналізу компонентів React.

3. Редактор має вбудований термінал, що дозволяє:

– запускати команди для створення проекту через create-react-app, vite, або інші інструменти;

– виконувати команди, пов’язані з npm або Yarn, без переключення між вікнами.

4. VS Code має вбудовану підтримку Git, яка дозволяє:

– легко відстежувати зміни в коді;

– управляти гілками проекту і виконувати коміти прямо з редактора.

5. Live Server і налагодження для швидкого перегляду змін у браузері. Інструменти дебаггінгу спрощують процес пошуку помилок у коді.

6. Підтримка TypeScript: якщо в проєкті React використовується TypeScript, VS Code забезпечує відмінну підтримку, включаючи автодоповнення, типізацію та перевірку помилок.

Отже, використання VS Code для проектів на React є практичним і ефективним вибором, хоча це не є вимогою. Ключові можливості та характеристики Visual Studio Code [40] розглянемо більш детально по пунктах.

Інтерфейс користувача. VS Code має чистий та інтуїтивно зрозумілий користувацький інтерфейс з бічною панеллю для навігації, центральною областю редагування та рядком стану для відображення інформації та інструментів. Він забезпечує вільний робочий простір, дозволяючи розробникам зосередитися на своєму коді (рис. 2.1– 2.2).

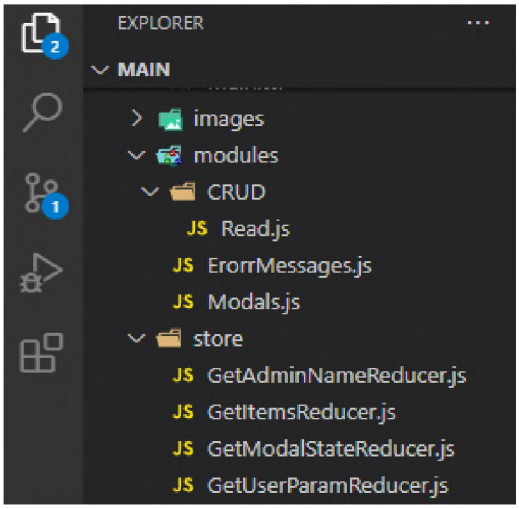


Рисунок 2.1 – Вигляд бічної панелі Visual Studio Code

Бічна панель VS Code забезпечує швидкий доступ до різних функцій та можливостей редактора коду. Бічна панель містить іконки навігації, такі як

Провідник, Розширення та інші, що дозволяють користувачам легко перемикатися між різними файлами та інструментами в редакторі.

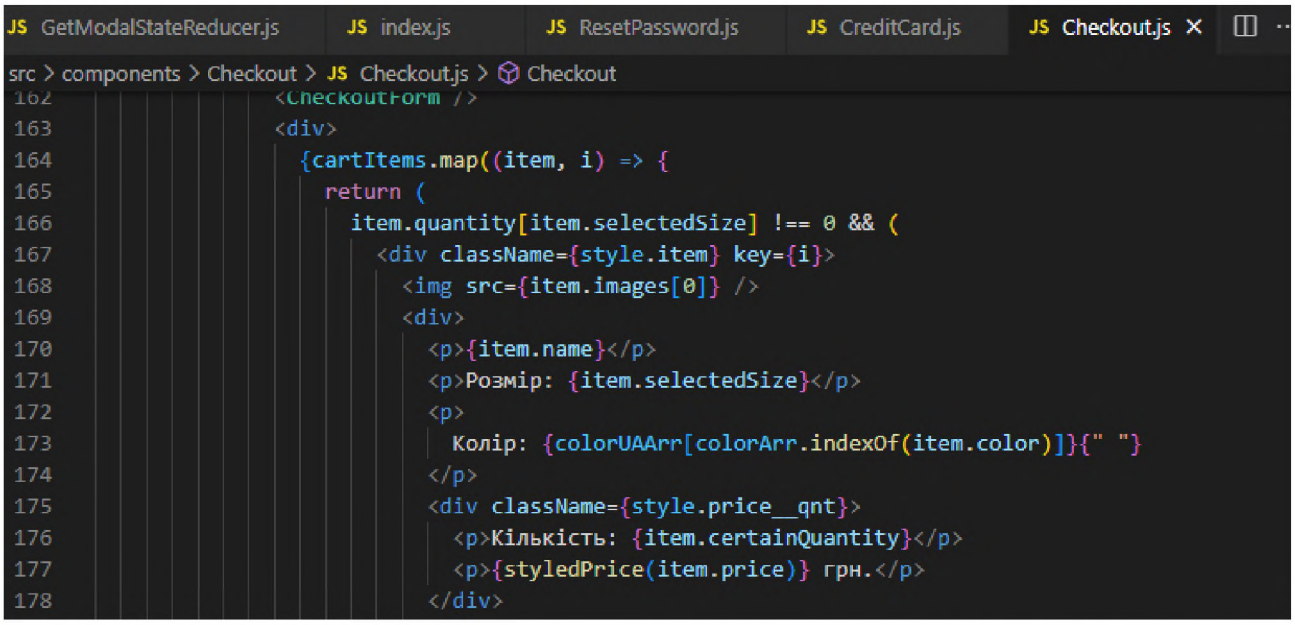


Рисунок 2.2 – Вигляд центральної області Visual Studio Code (фрагмент)

1. Крос-платформна сумісність. Visual Studio Code доступний для Windows, macOS та Linux, що забезпечує сумісність та однакову роботу з різними операційними системами.

2. Розширення та маркетплейс. Однією з визначних особливостей VS Code є його широка екосистема розширень. Розробники можуть налаштовувати та покращувати свій досвід кодування, встановлюючи широкий спектр розширень для різних мов програмування, фреймворків та інструментів. На Visual Studio Code Marketplace розміщено величезну колекцію розширень, створених спільнотою.

3. Мовна підтримка. Комплексна мовна підтримка для багатьох мов програмування, включаючи такі популярні, як JavaScript, Python, C++, Java, HTML, CSS та багато інших. Вона забезпечує підсвічування синтаксису, завершення коду, форматування, розбиття на рядки та можливості налагодження, специфічні для кожної мови.

4. Інтегрований термінал. Дозволяє розробникам запускати команди, компілювати код і виконувати різні завдання без перемикання на окремий

термінальний додаток. Це полегшує безперебійний робочий процес розробки в самому редакторі (рис. 2.3).

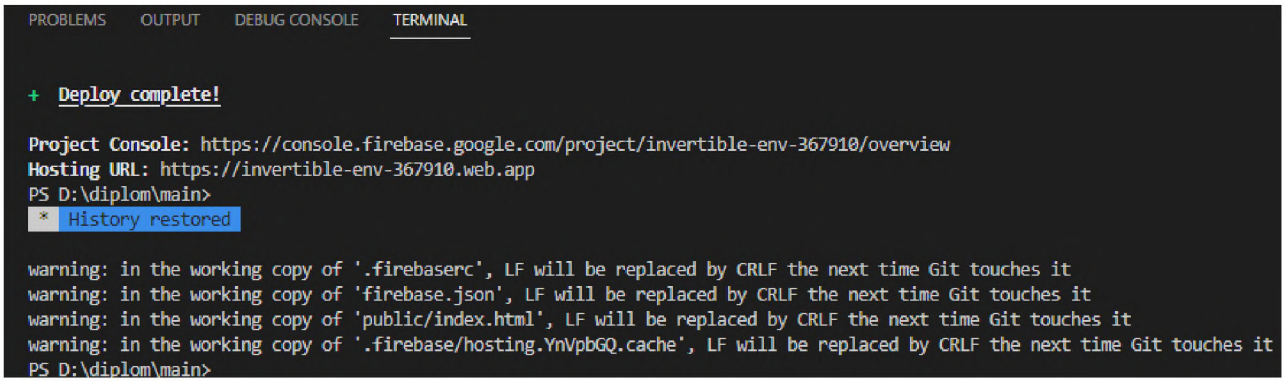


Рисунок 2.3 – Вигляд інтегрованого терміналу Visual Studio Code

5. Інтеграція з Git. VS Code має вбудовану інтеграцію з Git, що дозволяє контролювати версії та співпрацювати з колегами по команді (рис. 2.4). Розробники можуть легко переглядати та керувати репозиторіями Git'а, фіксувати зміни, переглядати різниці та виконувати операції з Git'ом, не виходячи з редактора [41].

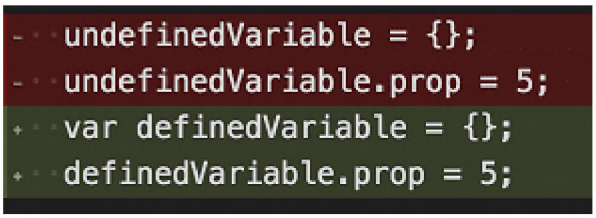


Рисунок 2.4 – Вигляд різних версій проєкту Visual Studio Code

6. Можливості налагодження. Visual Studio Code забезпечує надійну підтримку налагодження для різних мов і платформ. Вона пропонує покрокове налагодження, перевірку змінних та інші важливі функції налагодження, які допомагають розробникам виявляти та виправляти проблеми в коді.

7. IntelliSense. Інтелектуальна функція завершення коду у VS Code, яка надає контекстно-залежні пропозиції під час введення коду (рис. 2.5). Вона пропонує автозавершення, підказки щодо підписів функцій та документацію, щоб пришвидшити кодування та зменшити кількість помилок [42].

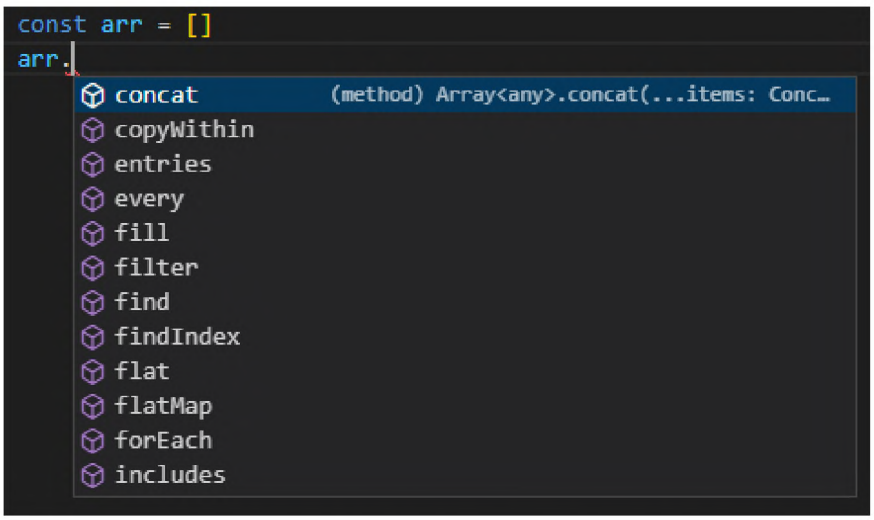


Рисунок 2.5 – Вигляд IntelliSense Visual Studio Code

8. Автоматизація завдань. VS Code підтримує автоматизацію завдань за допомогою інтегрованого бігуна завдань. Розробники можуть визначати власні завдання збірки, запускати скрипти і виконувати різні автоматизовані завдання в редакторі, підвищуючи продуктивність і ефективність робочого процесу.

9. Кастомізація. Visual Studio Code дуже добре налаштовується, що дозволяє розробникам персоналізувати своє середовище кодування. Користувачі можуть змінювати теми, встановлювати розширення, налаштовувати параметри та створювати власні комбінації клавіш, щоб пристосувати редактор до своїх уподобань [43].

Загалом, Visual Studio Code надає потужне та гнучке середовище кодування з багатим набором функцій, розширень та можливостей налаштування, що робить його популярним вибором для розробників усіх рівнів. Його легкість, широка мовна підтримка та динамічна екосистема сприяють його широкому розповсюдженню у спільноті розробників.

2.2 Приклади застосування обраного інструментарію на етапі проєктування та роботи з кодом вебдодатку

Розглянемо приклади використання обраних технологій для раціоналізації написання коду вебдодатків. Продемонструємо, зокрема, принципи роботи з бібліотекою jQuery та фреймворками.

jQuery використовують деякі з найважливіших компаній і вебсайтів у світі, як-от Microsoft, Amazon, Dell, Etsy, Netflix, Best Buy, Instagram, Fox News, GoDaddy та багато інших. Якщо у вас виникли сумніви щодо jQuery, то ці дані можуть переконати кожного, що це стабільна та надійна бібліотека, яку можна використовувати у своїх проектах. Для знайомства з особливостями бібліотеки передбачається, що розробник володіє мінімальними знаннями JavaScript.

Рекомендації для встановлення програми додаються на вебсайті, тому розпочати роботу в системі можна одразу і безпечно [28].

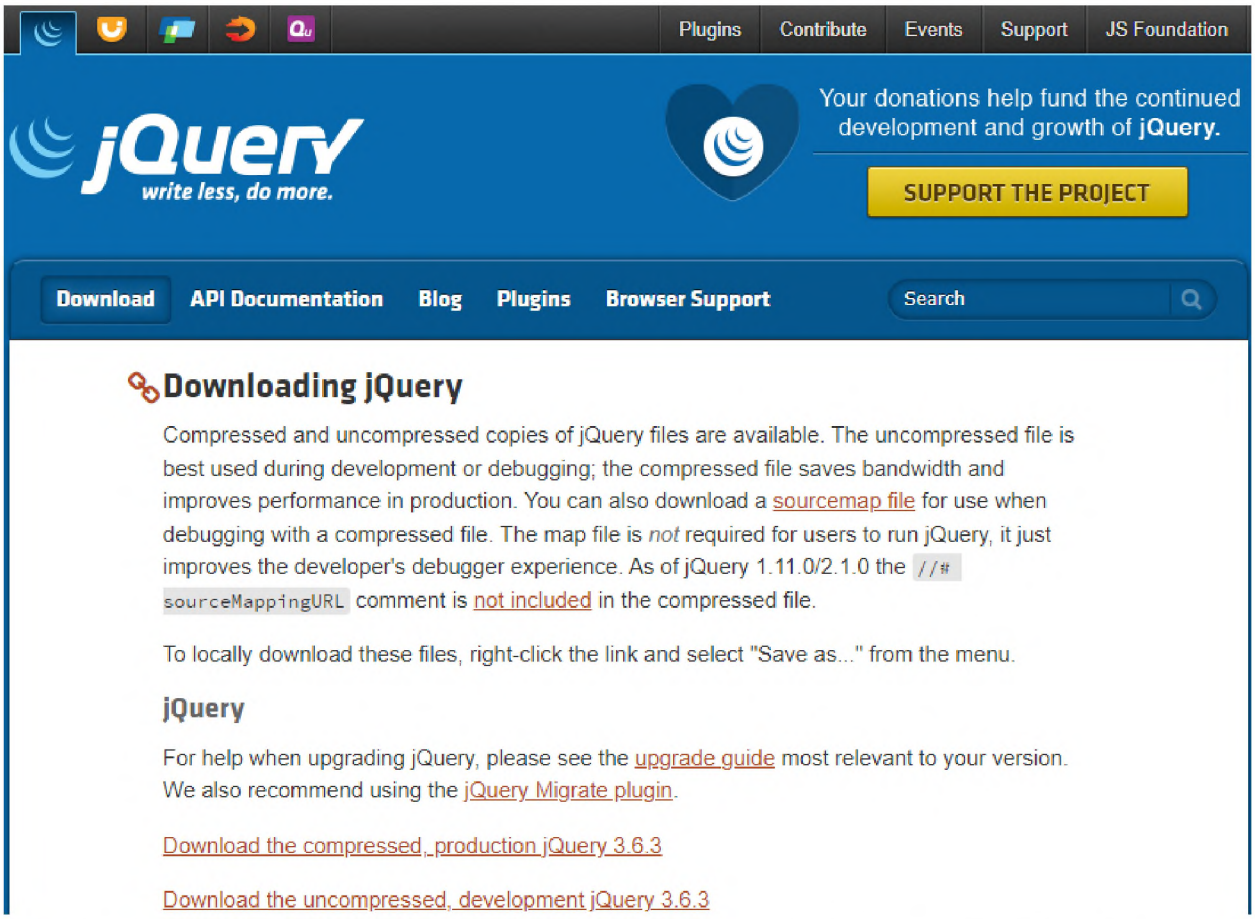


Рисунок 2.6 – Вікно завантаження бібліотеки jQuery із вибором версії

Доступні стиснені та нестиснуті копії файлів (див. рис. 2.6). Нестиснутий файл найкраще використовувати під час розробки або налагодження; стиснутий файл зберігає пропускну здатність і покращує продуктивність у виробництві.

Можна завантажити файл вихідної карти для використання під час налагодження за допомогою стисненого файлу. Файл карти не потрібен користувачам для запуску jQuery, він лише покращує роботу налагоджувача для розробника. Починаючи з jQuery 1.11.0/2.1.0 коментар //# sourceMappingURL не включено до стисненого файлу.

Обираємо завантаження не стиснутого файлу для початку використання. Щоб локально завантажити ці файли, достатньо клацнути посилання правою кнопкою миші та вибрати «Зберегти як...». Результат наведено на рис. 2.7.

Рисунок 2.7 – Код завантаженої бібліотеки та файл у папці

Документацію про попередні версії системи можна переглянути за спеціальним посиланням. На офіційному сайті викладено необхідні матеріали для вивчення можливостей бібліотеки.

Девіз jQuery: «Пиши менше, роби більше». Якщо ви витратили хоч якийсь час, намагаючись додати динамічну функціональність до своїх сторінок, то напевно виявили, що виконання простих завдань за допомогою необробленого JavaScript може призвести до десятків рядків коду (LoC). Автор jQuery спеціально створив цю бібліотеку, щоб зробити типові завдання тривіальними та легкими для вивчення, вирішуючи проблеми, спричинені несумісністю браузера.

Наприклад, всі, хто мав справу з радіогрупами в JavaScript, знає, що виявити, який радіоелемент радіогрупи наразі перевірено, і отримати його

атрибут значення – це нудний урок. Радіогрупу необхідно знайти, а отриманий набір радіоелементів необхідно перевірити один за одним, щоб з'ясувати, який елемент має встановлений атрибут перевірених. Тоді можна отримати атрибут значення цього елемента. Щоб бути сумісним з Internet Explorer 6 і вище, такий код можна реалізувати таким чином (рис. 2.8).

```
var checkedValue;
var elements = document.getElementsByTagName('input');
for (var i = 0; i < elements.length; i++) {
  if (elements[i].type === 'radio' &&
      elements[i].name === 'some-radio-group' &&
      elements[i].checked) {
    checkedValue = elements[i].value;
    break;
  }
}
```

Рисунок 2.8 – Приклад коду з radio-елементами в звичайному JS

Контраст можна побачити, якщо записати ту ж операцію з використанням jQuery (рис. 2.9).

```
var checkedValue =
  jQuery('input:radio[name="some-radio-group"]:checked').val();
```

Рисунок 2.9 – Раціональний код із включенням jQuery

Цей приклад (див. рис 2.8-2.9) показує, наскільки простим і лаконічним може бути код, написаний за допомогою jQuery. Це не єдина справжня сила jQuery: однією з його сильних сторін є можливість отримувати елементи за допомогою складних селекторів, не турбуючись про кросбраузерну сумісність.

Коли ви виконуєте вибір, ви покладаєтеся на дві речі: метод і селектор. Сьогодні останні версії всіх основних браузерів підтримують власні методи вибору елементів, такі як document.querySelector() і document.querySelectorAll(). Вони дозволяють використовувати більш складні селектори замість звичайного вибору за ID або класом. Крім того, нові селектори CSS3 широко підтримуються серед сучасних браузерів.

Той факт, що багато людей все ще покладаються на старіші браузери, які, можливо, доведеться підтримувати, може стати справжньою проблемою, оскільки доведеться мати справу з усіма невідповідностями. Це одна з головних причин використання jQuery, який дозволяє надійно використовувати його селектори, не турбуючись про те, що код не працює в старіших браузерах.

Ще один важливий фактор, на якому слід прийняти рішення, це те, де буде використовуватись jQuery. Ось кілька випадків використання, які можуть допомогти при виборі:

Вебсайти, яким не потрібна підтримка старіших версій Internet Explorer та інших браузерів, можуть використовувати гілку 3.x. Це стосується вебсайтів, які працюють у контрольованому середовищі, як локальні мережі компанії.

Вебсайти, які мають націлюватися на якомога ширшу аудиторію, наприклад урядовий веб-сайт, повинні використовувати гілку 1.x.

Якщо ви розробляєте вебсайт, який має бути сумісним із ширшою аудиторією, але вам не обов'язково підтримувати Internet Explorer 6–7 і старі версії Opera та Safari, вам слід використовувати jQuery Compat 3.x.

Якщо вам не потрібна підтримка Internet Explorer 8 і нижче, але ви повинні підтримувати старі версії Opera та Safari, вам слід використовувати jQuery 2.x.

Мобільні програми, розроблені за допомогою PhoneGap або подібних фреймворків, можуть використовувати jQuery 3.x.

Окремої уваги заслуговують препроцесори до мов, що розглядаються – це Pug для HTML та SCSS для CSS. Процес написання HTML і CSS може виявитися дещо виснажливим і вимагати безлічі одних і тих же завдань знову і знову. Такі різні завдання, як правило, невеликі, знижують ефективність. На щастя, ці та кілька інших неефективних завдань були визнані і виклик ним кинули препроцесори. Що стосується HTML і CSS, одні з найпопулярніших мов препроцесора – це Haml і Sass. Haml перетворюється на HTML, а Sass перетворюється на CSS [44]. В свою чергу Pug є надлаштуванням над Haml, так званим шаблонізатором, але є його дещо покращеною версію і також слідує

принципу «Don't repeat yourself» (DRY; з англ. – «не повторюйся») і сприяє добре структурованій розмітці, допомогаючи досвідченим розробникам працювати більш комфортно та уникати рутини (рис. 2.10).

```
html(lang="ru")
  head
    meta(charset="utf-8")
    title= "Учимся работать с шаблонизатором Pug"
  body
      h1 Hello, World!
```

Рисунок 2.10 – Приклад синтаксису коду на Pug

Pug покращує візуальне сприйняття коду для людини, знайомої з його синтаксисом. Sass це свого роду розширення, створене для спрощення каскадних таблиць стилів (CSS), це скриптова метамова, тобто мова програмування, що описує іншу [44]. Цей модуль також входить в Haml і має схожий з ним та Pug зовнішній вигляд, отже розробники можуть з легкістю розуміти синтаксис та структуру цих мов програмування.

Коли йдеться про Sass, як правило, ми маємо на увазі препроцесор та мову в цілому. Тим не менш, використовуючи Sass як препроцесор, ми можемо використовувати два різні синтаксиси (рис. 2.11):

SASS

```
$color: red
$color2: lime

a
  color: $color
  &:hover
    color: $color2
```

SCSS

```
$color: #f00;
$color2: #0f0;

a {
    color: $color;
    &:hover {
        color: $color2;
    }
}
```

CSS

```
a {
    color: red;
}
    a:hover {
        color: lime;
    }
```

Рисунок 2.11 – Різниця в синтаксисі SASS, SCSS та CSS

– Sass (відступи);

– SCSS (CSS-подібний синтаксис).

Отже, Sass має Ruby-подібний синтаксис, схожий на Pug, тобто для позначення вкладеності використовуються пробіли або таби, а SCSS більше

схожий на CSS, в який він згодом звичайно компілюється, і у файлі з розширенням .css можна з легкістю замінити розширення на .scss і він буде працювати так, як і до цього. Для заміни розширення файлу .scss в .css потрібно провести декілька більше маніпуляцій з кодом, але в цілому це робиться досить швидко та просто. Використання цих препроцесорів може дуже значно полегшити розробку верстальнику і позбавити його зайвої рутини та зробити весь процес розробки більш приємним.

На офіційному сайті розробників React викладено документацію, навчальні матеріали на багатьох мовах, у т. ч. й українською. Крім того, є тренувальні вправи, а також інформація про попередні версії.

React-додаток будується з компонентів, саме в них і відбувається оновлення і декларація елементів сторінки. Тобто, компонент це головне поняття в React, єдина сутність, яку він містить. React-компонент може повертати HTML-код за допомогою JSX. У звичайному JavaScript таке неможливо, це також дуже зручна надбудова над мовою для зручності розробки.

Компоненти реалізують метод render(), який приймає вхідні дані і повертає те, що буде показано користувачу (рис. 2.12).

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Привіт, {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Віталій" />,
  document.getElementById('hello-example'),
);
```

Рисунок 2.12 – Приклад застосування елементів та методу render()

У цьому прикладі (див. рис. 2.12) використовується XML-подібний синтаксис під назвою JSX. Доступ до вхідних даних, які передаються в компонент, можна отримати за допомогою render() та this.props. Результатом

виконання прикладу буде виведення привітання в інтерактивному режимі. Зазвичай, така модель обробки даних поширюється на більш складні мовні інтерактивні взаємодії, і потужність фреймворку виявляєься на повну сутність.

JSX – це препроцесор (розширення JavaScript), що спрощує створення елементів і компонентів React, дозволяє декларативно створювати компоненти користувальницького інтерфейсу. Розширення JSX володіє наступними вагомими можливостями [45]:

– застосування простої декларативною розмітки;

– код розмітки розташований там само, де і код компонента;

– реалізація принципу поділу відповідальностей (наприклад - відділення опису інтерфейсу від логіки стану і від побічних ефектів). При цьому реалізація базується не на використанні різних технологій (наприклад - HTML, CSS,

– абстрагування управління змінами DOM;

– абстрагування від особливостей різних платформ, для яких створюють React-застосунки.

З цим розширенням під час написання коду потрібно набагато менше зусиль, порівняно з класичним JavaScript. JSX трансформується в JavaScript перед запуском у браузері. Він не є обов'язковим під час використання React.

Існує два підходу до розробки – декларативний та імперативний. Суть декларативного підходу в описі кінцевого результату, а імперативного в тому, що ми описуємо конкретні кроки для досягнення результату, тобто яким способом і засобами ми хочемо його досягти. Виявилось, що декларативний підхід добре підходить для розробки інтерфейсів, і саме його притримується React. Завдяки цьому він зміг досягти такої популярності в світі та зайняти провідне місце у веброзробці. Наприклад, можна порівняти тривіальний приклад лічильника, реалізований імперативним способом за допомогою мови розмітки HTML і JavaScript (рис. 2.13) і декларативним методом за допомогою бібліотеки React (рис. 2.14).

```html
<!DOCTYPE html>
<html>

<head>
    <title>Імперативний метод</title>
    <meta charset="UTF-8" />
</head>

<body>
    <div class="root">
        <h1 id="counter-value"></h1>
        <button id="increment-btn">+1</button>
    </div>

    <script>
        const counterValueElement = document.getElementById("counter-value");
        const incrementBtn = document.getElementById("increment-btn");
        let counterValue = 0;

        function increment() {
            counterValue += 1;
            counterValueElement.innerText = counterValue;
        }

        counterValueElement.innerText = counterValue;
        incrementBtn.addEventListener("click", increment);
    </script>
</body>
```

Рисунок 2.13 – Приклад декларативного підходу до веброзмітки

```html
<head>
    <title>Декларативний метод</title>
    <meta charset="UTF-8" />
</head>

<body>
    <div id="root"></div>
    <script src="https://unpkg.com/react@17/umd/react.development.js" crossorigin></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone@7.13.6/babel.min.js" crossorigin></script>
    <script type="text/babel">
        function App() { const [value, setValue] = React.useState(0); function increment() { setValue
        (value + 1); } return (
        <div className="app">
            <h1>{value}</h1>
            <button onClick={increment}>+1</button>
        </div>
        ); } ReactDOM.render(
        <div>
            <App />
        </div>, document.getElementById("root") );
    </script>
</body>
```

Рисунок 2.14 – Приклад імперативного підходу з використанням

Як ми бачимо, імперативний метод є хоч більш компактним і візуально зрозумілішим, але має певні недоліки, наприклад необхідність зберігати елементи на сторінці в змінні, яких зазвичай набагато більше, а от в React

рендер відбувається в самому компоненті без необхідності декларування змінних і їх збереження.

Styled-jsx дозволяє оголошувати вбудовані стилі у компонентах React, подібно до того, як теги <style> працюють у HTML. Сфера застосування цих стилів є гіперлокальною, що означає, що стилі впливатимуть лише на рідні теги та їхні дочірні теги. Next.js також включає styled-jsx за замовчуванням.

Virtual DOM у React представляє легковажну копію звичайного DOM. І відмінною особливістю React є те, що ця бібліотека працює саме з віртуальним DOM, а не звичайним.

Якщо програмі необхідно уточнити інформацію про стан елементів, то вона звертається до віртуального DOM. За рахунок Virtual DOM має збільшуватися продуктивність.

React надає декларативний API і це створює умови, за яких не потрібно турбуватися про зміни, які відбуваються. Це значно спрощує написання додатків.

React надає повний контроль над розміром додатка, дозволяючи включити тільки ті речі, які дійсно потрібні. Він дає більше гнучкості під час переходу від односторонніх додатків (SPA) до мікросервісів, використовуючи частини колишнього додатка.

React чудово підійде для проєктів, де важлива гнучкість при створенні великих екосистем, що мають тенденцію розростатися. Дуже добре підходить для командної розробки. UI-код читабельний і простий у супроводі. Здатний повністю взяти на себе відповідальність за рівень подання в архітектурі MVC (модель, подання, контролер). Розробка користувацького інтерфейсу за цим принципом – це сучасний підхід у сфері надання даних.

Файли CSS можна завантажити в заголовок вашої сторінки для загальних глобальних макетів, шрифтів тощо. Вони чудово працюють.

Модулі CSS – це локальні файли CSS, які можна імпортувати безпосередньо у файли JavaScript. Для цього знадобиться правильно налаштований завантажувач. Next.js увімкне це за замовчуванням.

Отже, можна зробити висновок, що фреймворк React є надзвичайно зручним інструментом у веброзробці і за допомогою неї можна будувати сучасні та швидкі вебдодатки. Компоненти можуть допомогти розбити інтерфейс на незалежні частини і використовувати їх повторно за потреби та комбінувати будь-яким чином. Це клас, що наслідується з кореня бібліотеки.

2.3 Технології візуалізації макету вебсайту та опис бізнес-моделі

Перед початком реалізації вебсайту потрібно створити макет майбутнього сайту. Завдяки макету можна до найменших подробиць продумати, як буде виглядати майбутній продукт: розташування елементів, кольорова гамма, шрифти та загальна структура сторінок. Також це дозволяє уникнути непорозумінь із замовником.

Для реалізації макетів можна використовувати один з наступних інструментів: Adobe Photoshop, Figma, Zeplin, Framer, Sketch. Для порівняння було створено таблицю 2.2. Кожному критерію проставлена оцінка від 1 до 5, після чого порахована сума балів.

Таблиця 2.2 – Оцінювання параметрів графічних редакторів для макетування за 5-бальною шкалою

Критерій	Adobe Photoshop	Figma	Zeplin	Framer	Sketch
Легкість використання	2	5	4	2	4
Командна розробка	3	5	4	2	3
Інтеграція з іншими інструментами	5	4	4	3	4
Прототипування	2	5	1	4	3
Швидкість роботи	3	5	3	4	4
Доступність	3	5	4	4	4
Функціонал для дизайну	5	5	4	4	4
Екосистема плагінів	4	4	3	3	4
Разом балів	27	38	27	26	30

За результатами порівняння (див. табл. 2.2), найкращим варіантом виявився інструмент Figma. Це хмарна технологія для створення макетів, в тому числі вебсайтів. Однією з ключових особливостей Figma є можливість спільної

роботи в реальному часі, що дозволяє дизайнерам, розробникам та іншим учасникам команди одночасно працювати над проєктом з різних пристроїв. Інтерфейс Figma зображено на рис. 2.15.

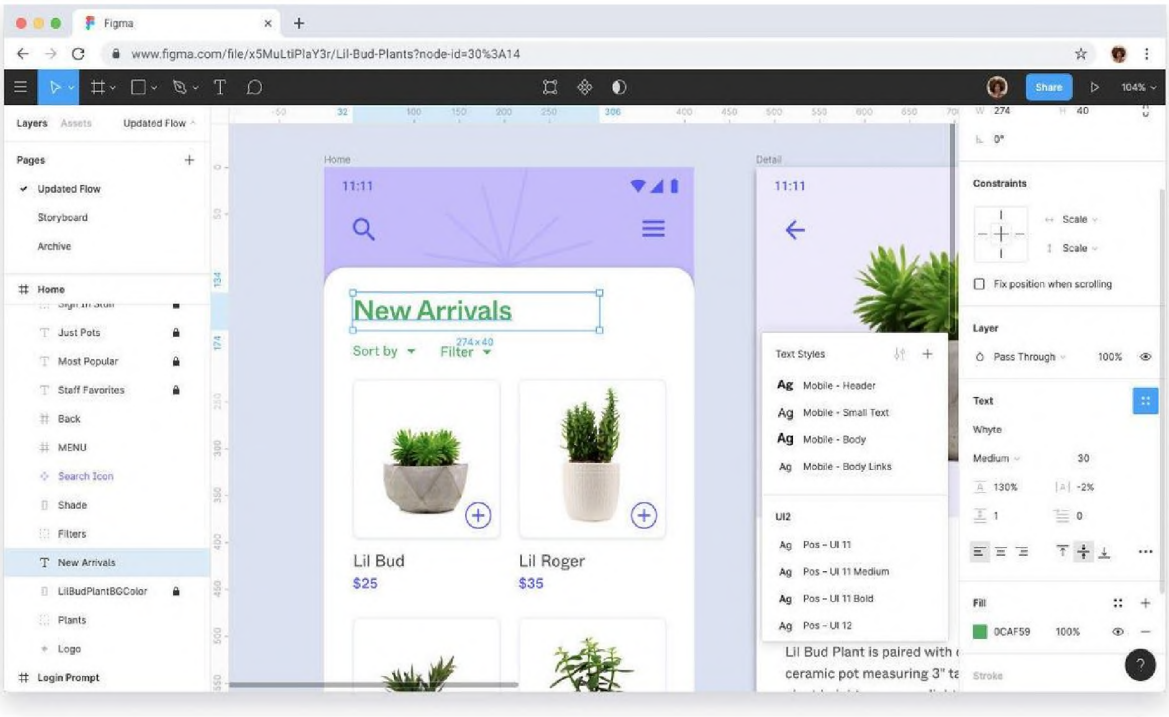


Рисунок 2.15 – Інтерфейс Figma

Для успішного функціонування вебсайту про переваги використання Proton Trade необхідна чітка бізнес-модель. Однією з можливих стратегій стала комісійна модель, де користувачі вебсайту оплачують комісію за використання трейд-бота та отримання прибутку від автоматичної торгівлі. Також можна розглянути модель підписки, де користувачі мають можливість обрати різні плани підписки залежно від їхніх потреб та отримувати додаткові функції та переваги. Завдання частин вебсайту – максимально чітко та зрозуміло й наочно показати вигоди від використання трейд-бота.

Для початку необхідно створити орієнтовний макет різних частин вебдодатку. Загальний план – структура у вигляді лендінгу. Макет комерційного вебсайту повинен бути розроблений таким чином, щоб забезпечити безперебійний і візуально привабливий досвід перегляду (UI-дизайн). Він повинен бути добре організованим, з чіткою ієрархією, яка

дозволяє користувачам легко знаходити інформацію та продукти, які вони шукають. Схема макету показана на рис. 2.16.

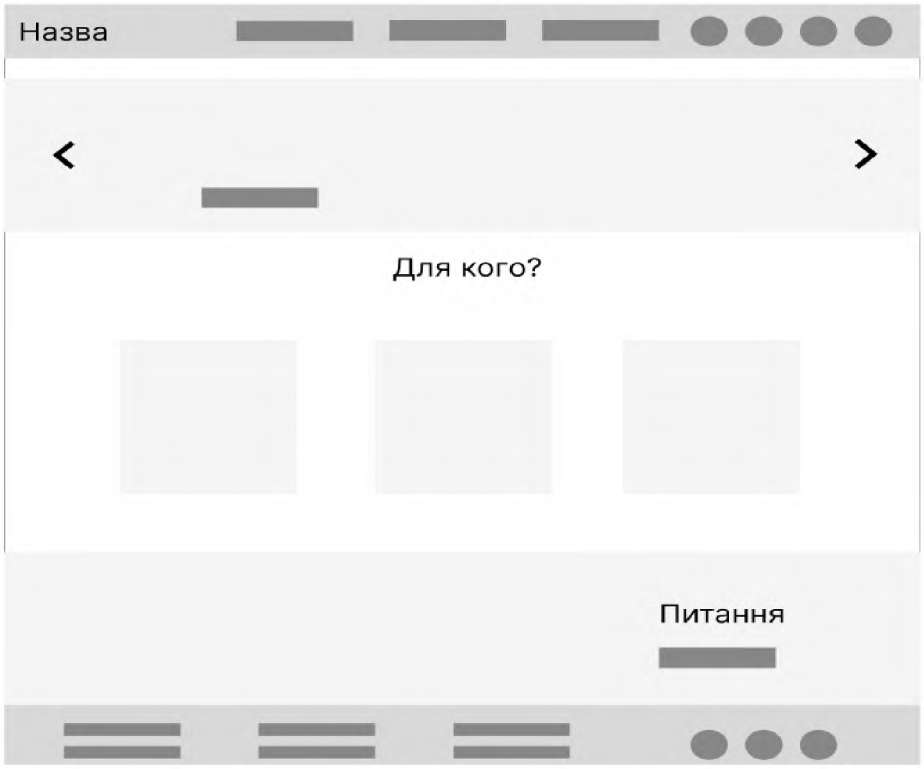


Рисунок 2.16 – Вигляд макету головної сторінки

Макет головної сторінки буде містити динамічні графічні елементи, які при певних діях і інтервалі часу змінюються в розмірах, основну частину з висвітленням можливостей трейд-боту та блок з посиланням на сторінку допомоги, де користувач зможе відправити листа

Візуальні елементи, такі як зображення, кольори та типографіка, повинні використовуватися стратегічно, щоб покращити загальну естетику сайту та відобразити ідентичність його призначення. Сайт за планом виготовлений в стилі мінімалізму. В якості графіки використано популярний сторітеллінг, іконки (зображення формату .png). Графічний проєкт основної частини вебсайту показано на рис. 2.17.

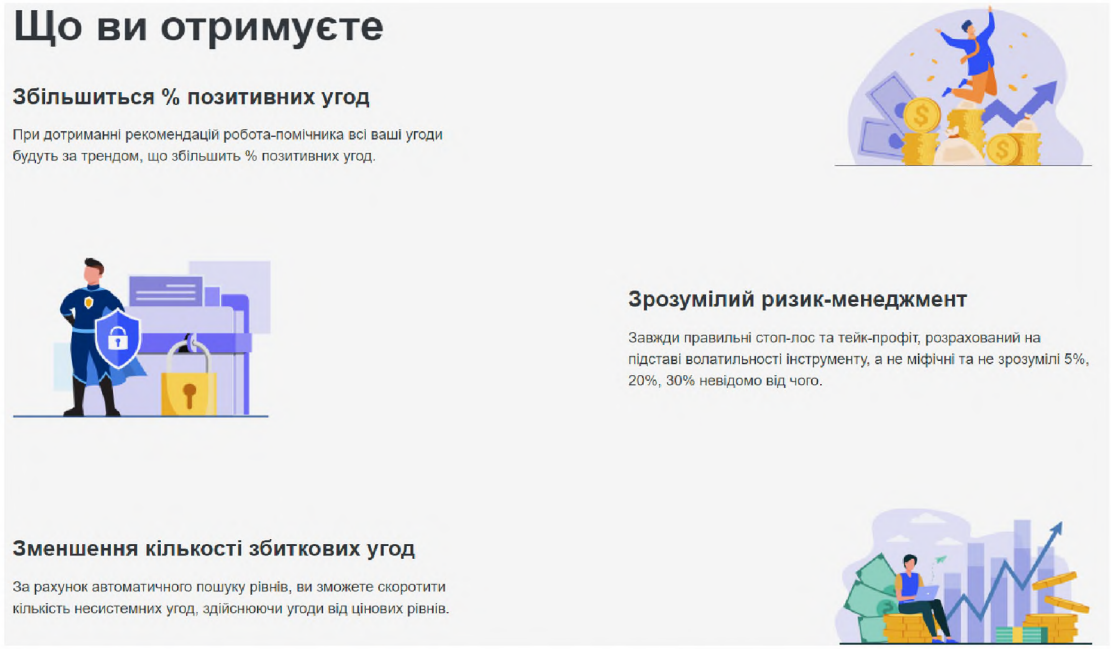


Рисунок 2.17 – Мінімалістичний стиль основної частини сайту і сторітеллінг

Окремим блоком представлено картки функцій робота – помічника. В якості змістового наповнення в текстовій формі представлені пояснення основних функцій робота. При цьому враховано потреби користувача. Кожний інформаційний блок виділений фоновою областю та іконкою-символом (рис. 2. 18)

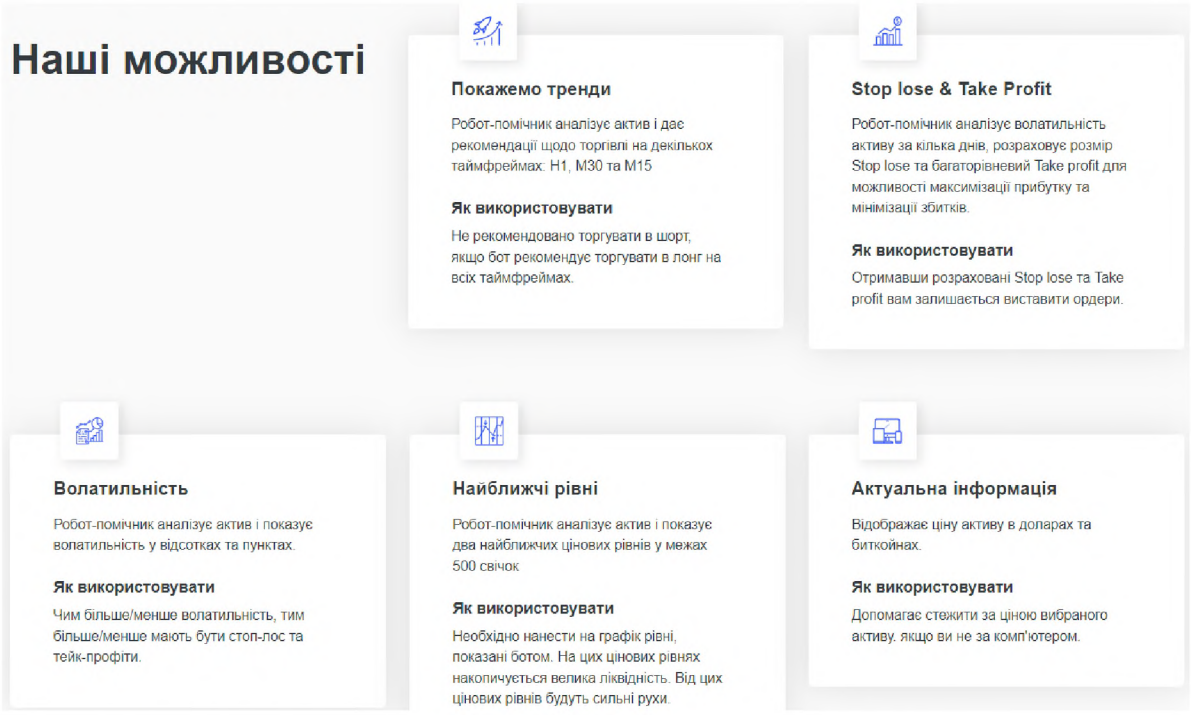


Рисунок 2.18 – Вигляд макету основної частини сторінки

Нижня частина вебсайту містить посилання на партнерів та форму зворотного зв’язку (рис. 2.19).

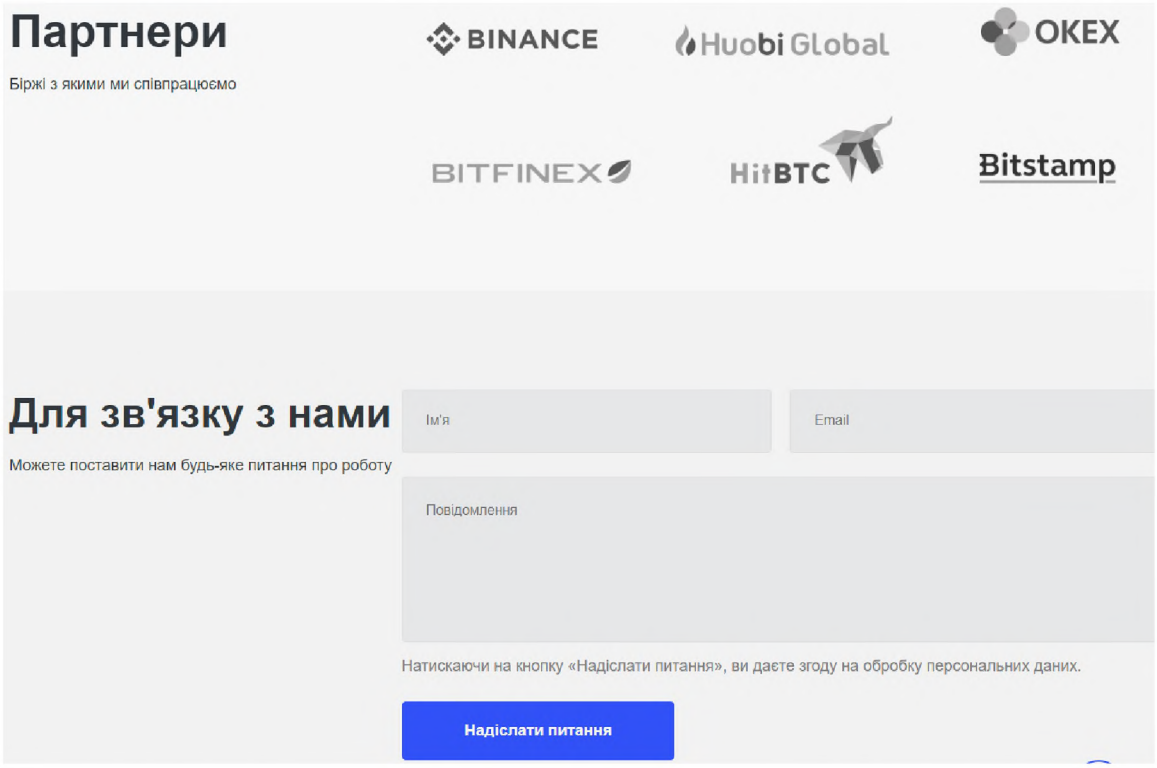


Рисунок 2.19 – Вигляд макету нижньої частини сторінки з формою зворотного зв’язку

Трьом частинам головної сторінки відповідають такі елементи навігації, як горизонтальне меню та стрілки для вертикального пересування. Три пункти свідчать про розбивку основної частини на три основних екранних блоки (рис. 2.20).

Рисунок 2.20 – Вигляд верхньої частини макету з горизонтальним меню

В наступній частині представлено опис структури коду вебсайту як набору програмних файлів та елементів коду, які відповідають за окремі функції виконаного вебсайту. Окремо продумана адаптивність вебсайту до різних екранів.

Всі представлені елементи було заплановано зробити динамічними із використанням таких ефектів, як спливаючі зображення при прокручуванні сайту. Для цього підібрано інструменти технологій CSS, бібліотек JavaScript.

Висновки до розділу 2

В даному розділі сформульовано та поділено на логічні кроки задачу практичної частини кваліфікаційної роботи. Вона полягатиме в створенні прикладу сучасного вебдодатку із використанням сучасних технологій, які спрощують та виводять на якісно вищий рівень роботу розробника.

Було проаналізовано, обрано та описано різноманітний інструментарій для реалізації: інтегроване середовище розробки, фреймворки, бібліотеки, систему автоматичної збірки та графічний редактор. Акцент зроблено на поєднанні та сумісності різних інструментів та внеску в загальний процес розробки вебдодатку.

Показано, що фреймворк React є надзвичайно зручним інструментом у веброзробці, і за допомогою нього можна будувати сучасні та швидкі вебдодатки. Компоненти можуть допомогти розбити інтерфейс на незалежні частини і використовувати їх повторно за потреби та комбінувати будь-яким чином.

Поглиблений аналіз показав, наскільки простим і лаконічним може бути код, написаний за допомогою бібліотеки jQuery. Серед її сильних сторін є можливість отримувати елементи за допомогою складних селекторів, не турбуючись про кросбраузерну сумісність.

РОЗДІЛ 3
ПРАКТИЧНЕ ЗАСТОСУВАННЯ ВЕБТЕХНОЛОГІЙ В РЕАЛІЗАЦІЇ ПРОЄКТУ КОМЕРЦІЙНОГО ВЕБДОДАТКУ

3.1 Технічні прийоми створення основних розділів вебсайту

На початку формується HTML код. Важливою частиною створення будь-якого вебдодатку є семантична оптимізація коду. Семантична оптимізація коду фокусується на використанні відповідних тегів HTML для відображення структури та ієрархії контенту. Це допомагає пошуковим системам зрозуміти контекст і релевантність інформації, що потенційно покращує видимість сайту в результатах пошукової видачі. На рис. 3.1 представлено структуру верхньої частини, в якій використано з поміж інших семантичні теги <header> і <nav>.

```
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="assets/css/style.min.css">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="robots" content="index, follow">
    <meta name="google" content="notranslate">
    <meta name="format-detection" content="telephone=no">
    <meta name="description" content="">

    <title>Proton Trade</title>
</head>
<body>

    <div class="page">
        <header class="header"  id="top">
            <div class="container">
                <div class="header__wrapper">
                    <div class="header__logo logo">
                        <div class="logo__img">
                            <img src="assets/images/logo.png" alt="Logo Proton Trade">
                        </div>
                        <div class="logo__text">
                    </div>

                    <nav class="header__nav">
                        <ul class="header__list">
                            <li>
                                <a class="header__link lift" href="#capabilities" >
                                    Наші можливості
                                </a>
                            </li>
                            <li >
                                <a class="header__link lift" href="#you-get">
                                    Що ви отримуєте
                                </a>
                            </li>
                            <li>
                        </ul>
                    </nav>
```

Рисунок 3.1 – HTML-кодування верхньої частини головної сторінки вебсайту

Використовуючи описові та змістовні імена для елементів HTML, розробники можуть зробити свій код більш зрозумілим і легким для сприйняття. Це не тільки приносить користь іншим розробникам, які можуть працювати над проектом, але й сприяє загальній легкості обслуговування кодової бази. Перше що видно на головній сторінці це header (верхній колонтитул або «шапка»). Після нього зображений банер з назвою теми сайту (див. рис. 3.1). Повний HTML-код вебсторінки представлено в додатку А. Якщо брати до уваги інші файли, а саме зображення, то вони знаходяться в кореневій папці – у папці «images», де знаходяться всі необхідні зображення.

Ще одним ключовим аспектом семантичної оптимізації коду є розмежування проблем, пов'язаних із контентом, представленням і поведінкою. Такий підхід дозволяє розробникам підтримувати чітке розділення HTML, CSS і JavaScript, що полегшує оновлення та модифікацію певних аспектів вебсайту, не впливаючи на інші.

Для управління функціями головної сторінки створені 2 файли: перший це файл з розширенням «.js», який слугує для написання коду; інший файл із розширенням «.css», який призначений для стилю різних елементів коду. Підключення останнього представлено в рядку в частині <head>:

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

SCSS (Sassy CSS) – це препроцесор, який розширює можливості звичайного CSS. Він надає додаткові можливості, такі як змінні, вкладеність та інші, які роблять написання стилів більш ефективним та організованим [46]. Повний лістинг таблиці стилів (компактної версії) представлено в додатку Б.

При верстанні вебсайту використані сучасні функції CSS, які дозволяють досягнути динамічних ефектів. Наприклад, для появи окремих блоків під час прокручування або натискання одного з пунктів меню використано спеціальну функцію transition: 1s; }, записану в класі .demo, а також властивості об'єкта в різних епізодах із вказанням максимального розміру блоку від появи до розкриття (.demo._active). Фрагмент відповідного запису стилів у файлі style.css представлено на рис. 3.2.

```
.demo {
  background: linear-gradient(#4867ff, #4867ff), url(../../assets/images/web.png) center/cover no-repeat;
  background-blend-mode: overlay;
  position: relative;
  opacity: 0;
  margin-left: -100%;
  transition: 1s; }
  .demo._active {
    margin-left: 0;
    opacity: 1; }
  @media (max-width: 768px) {
    .demo {
      padding-bottom: 1rem; } }
```

Рисунок 3.2 – Запис властивостей елементів, які динамічно з'являються на основній частині вебсторінки

У середній частині головної сторінки вебсайту необхідно привернути увагу до партнерських компаній, що вже використовують трейд-бот. Це зроблено у вигляді трансформації зображень на сайті з логотипами цих компаній. Властивості трансформації кожної частини блоку прописані в кількох класах: .partners__company, .partners__descript, та активований .partners__descript._active (рис. 3.3). Повний перелік див. в додатку Б.

```
.partners__company {
  flex-basis: 66%;
  display: flex;
  justify-content: space-between;
  flex-wrap: wrap;
  align-items: flex-end; }
  @media (max-width: 1200px) {
    .partners__company {
      flex-basis: 75%; } }
.partners__img {
  margin: 0 auto;
  opacity: 0;
  transform: scale(0);
  transition: 1.6s; }
  .partners__img._active {
    opacity: 1;
    transform: scale(1); }
```

Рисунок 3.3 – Приклад використання функції трансформації зображень в CSS із заданими властивостями флекс-елементів

На початку було передбачено забезпечити для користувачів доступ до сайту з будь-яких пристроїв, тому елементи є адаптивними. Так, класичний

приклад забезпечення гнучкості блоків за допомогою властивостей display: flex; , flex-wrap: wrap; , flex-basis наведено на рис. 3.4.

```
862  .partners__company {
863    flex-basis: 66%;
864    display: flex;
865    justify-content: space-between;
866    flex-wrap: wrap;
867    align-items: flex-end; }
868    @media (max-width: 1200px) {
869      .partners__company {
870        flex-basis: 75%; } }
871  .partners__img {
```

Рисунок 3.4 – Адаптивні flex-властивості блоків

Для нижньої частини вебсайту показані властивості для зміни порядку розташування блокових елементів при перегляді на малому екрані, зокрема, flex-direction: column;, а також граничні розміри зменшення (рис. 3.5).

```
892  .contact-us__wrapper {
893    display: flex;
894    justify-content: space-between; }
895    @media (max-width: 768px) {
896      .contact-us__wrapper {
897        flex-direction: column;
898        text-align: center; } }
899  @media (max-width: 768px) {
900    .contact-us__caption {
901      padding-bottom: 5rem; } }
902  @media (max-width: 481px) {
903    .contact-us__caption {
```

Рисунок 3.5 – Набір flex-властивостей для блоків нижньої частини вебсайту

Перевірка адаптивності наведена на рис. 3.6 – 3.7.

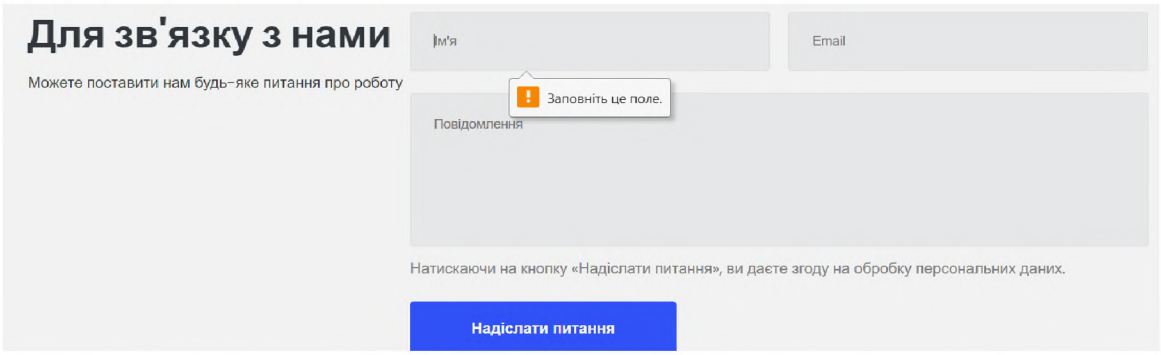


Рисунок 3.6 – Вигляд форми контактів та заголовку на звичайному екрані

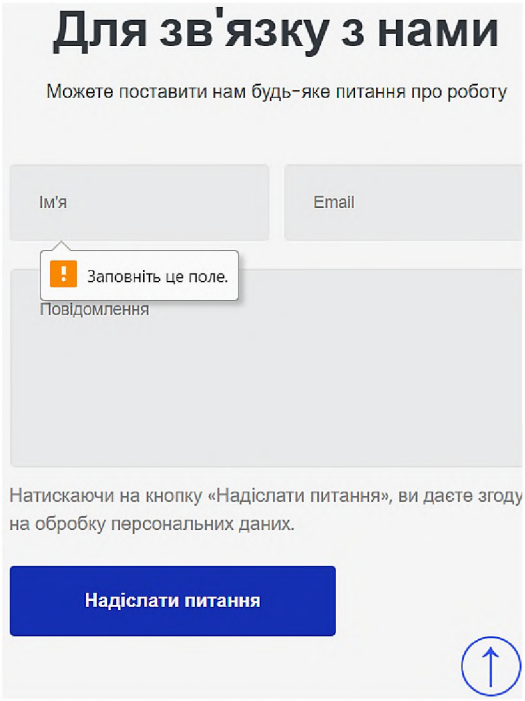


Рисунок 3.7 – Розташування елементів форми на екрані смартфона

Для роботи з об’єктами використано набір функцій JavaScript. Підключення файлу app.js розміщено в нижній частині вебсайту у вигляді скрипта (рис. 3.8), що дозволяє уникнути сповільнення завантаження та роботи з вебсайтом у браузері.

```
302        </div>
303
304        <script src="assets/js/app.js"></script>
305    </body>
306    </html>
```

Рисунок 3.8 – Запис посилання на файл app.js в нижній частині HTML-коду

Повний лістинг цього файлу представлено в додатку В. Для прикладу розглянемо один із методів, який використаний для роботи з елементами. Мова йде про статичний метод Object.defineProperties(), який визначає нові або змінює існуючі властивості безпосередньо об’єкта, повертаючи об’єкт. Синаксис методу записується Object.defineProperties(obj, props). При цьому:

obj - об’єкт, для якого потрібно визначити або змінити властивості;

props - об'єкт, ключі якого представляють назви властивостей, які потрібно визначити або змінити, а значення – об'єкти, що описують ці властивості. Кожне значення в props має бути або дескриптором даних, або дескриптором доступу; не може бути обох одночасно [47].

Відповідний фрагмент запису методу наведено на рис. 3.9 разом із визначенням функції __webpack_require__.t.

```
48  /******/ 	Object.defineProperty(exports, '__esModule', { value: true });
49  /******/ };
50  /******/
51  /******/ // create a fake namespace object
52  /******/ // mode & 1: value is a module id, require it
53  /******/ // mode & 2: merge all properties of value into the ns
54  /******/ // mode & 4: return value when already ns object
55  /******/ // mode & 8|1: behave like require
56  /******/ __webpack_require__.t = function(value, mode) {
57  /******/ 	if(mode & 1) value = __webpack_require__(value);
58  /******/ 	if(mode & 8) return value;
59  /******/ 	if((mode & 4) && typeof value === 'object' && value && value.__esModule) return value;
60  /******/ 	var ns = Object.create(null);
61  /******/ 	__webpack_require__.r(ns);
62  /******/ 	Object.defineProperty(ns, 'default', { enumerable: true, value: value });
63  /******/ 	if(mode & 2 && typeof value != 'string') for(var key in value) __webpack_require__.d(ns,
64  /******/ 	return ns;
65  /******/ };
```

Рисунок 3.9 – Приклад опису методу Object.defineProperties у файлі .js

Використання методів та функцій JS на основі фреймворку React дозволяє управляти та обробляти дані більш коректно, у тому числі завдяки алгоритмам розгалуження та циклам (див. рис. 3.9).

3.2 Методи оптимізації вебсайту та елементи тестування працездатності

При розробленні вебсайту враховано вимоги до пошукової оптимізації вебсайту та збільшення переглядів цільовою аудиторією. Заходи з оптимізації можна поділити на кілька груп.

1. Графіка та мультимедіа. Вставлені візуальні елементи, які доповнюють тематику сторінки, такі як ілюстрації, фотографії та відео. При цьому використані формати, що підтримуються веббраузерами, для оптимального

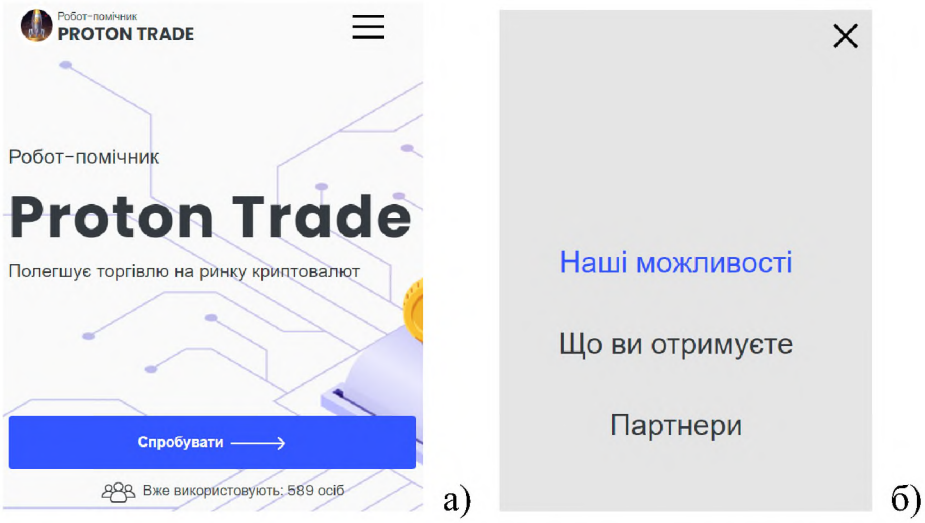
завантаження та відтворення. Окремо обрані спеціальні шрифти з бібліотеки Google, які надають виразності тексту.

2. Оптимізація та тестування на різних пристроях. Готовий вебсайт проходив попереднє тестування з метою усунення як недоліків у написанні коду, так і для правильного відображення на різних екранах. У попередньому розділі було зазначено про використання флекс-елементів для досягнення адаптивності різних блоків. Для збереження функцій горизонтального меню на малих екранах застосовано спеціальне бургер-меню, яке компактно згортає всі посилання в «бургер». Властивості, які формують бургер-меню, наведені на рис. 3.10.

```
.burger {
  display: none;
  z-index: 2;
  width: 3.4rem;
  height: 2.6rem;
  position: absolute;
  top: 2rem;
  right: 5rem;
  font-size: 0;
  color: transparent; }
  @media (max-width: 768px) {
    .burger {
      display: block; } }
  @media (max-width: 481px) {
    .burger {
      top: 2rem;
      right: 1rem; } }
  .burger:before, .burger:after,
  .burger span {
    display: block;
    width: 100%;
    height: 3px;
    background-color: #000;
    position: absolute;
    left: 0; }
  .burger:before, .burger:after {
    content: "";
    transition: transform 0.5s linear; }
  .burger:before {
    top: 0; }
  .burger:after {
    bottom: 0; }
  .burger span {
    top: 50%;
    transform: translateY(-50%);
    transition: opacity 0.2s linear; }
  .burger.active span {
    opacity: 0; }
```

Рисунок 3.10 – Правила стилів для оформлення бургер-меню на вебсайті

Результат роботи властивостей бургер-меню показано на рис. 3.11 при перегляді сайту на смартфоні.



а) б)

Рисунок 3.11 – Вигляд бургер-меню при перегляді на смартфоні в згорнутому вигляді (а) та активованому стані (б)

Адаптивний дизайн спрямований на те, щоб вебсайти були візуально привабливими, незалежно від пристроїв, з яких користувачі отримують доступ до них (стаціонарного ПК, ноутбука, планшета або смартфона [48]. Адаптивний дизайн був розроблений для всього сайту. Приклад для IPad Air – на рис. 3.12.

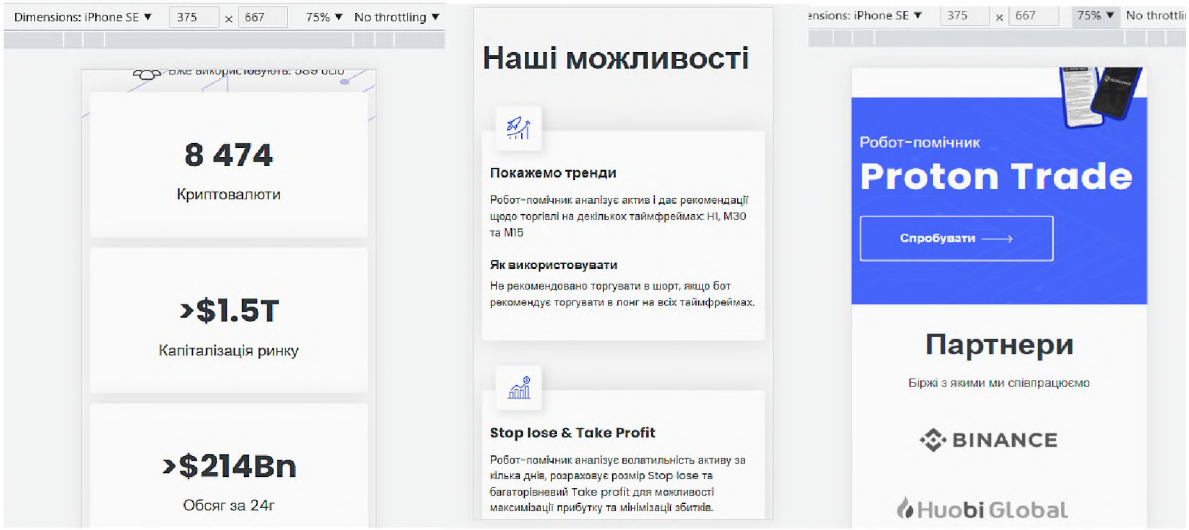


Рисунок 3.12 – Адаптивний формат частин вебсайту на екрані планшету

Перед розміщенням вебдодатку в мережі, було проведене тестування, щоб переконатися в його правильному відображенні та функціональності на

різних пристроях та браузерах за допомогою сервісу [49]. Виявлені проблеми виправлені, а також оптимізоване завантаження сторінки для поліпшення її продуктивності та швидкості завантаження (рис. 3.13.)

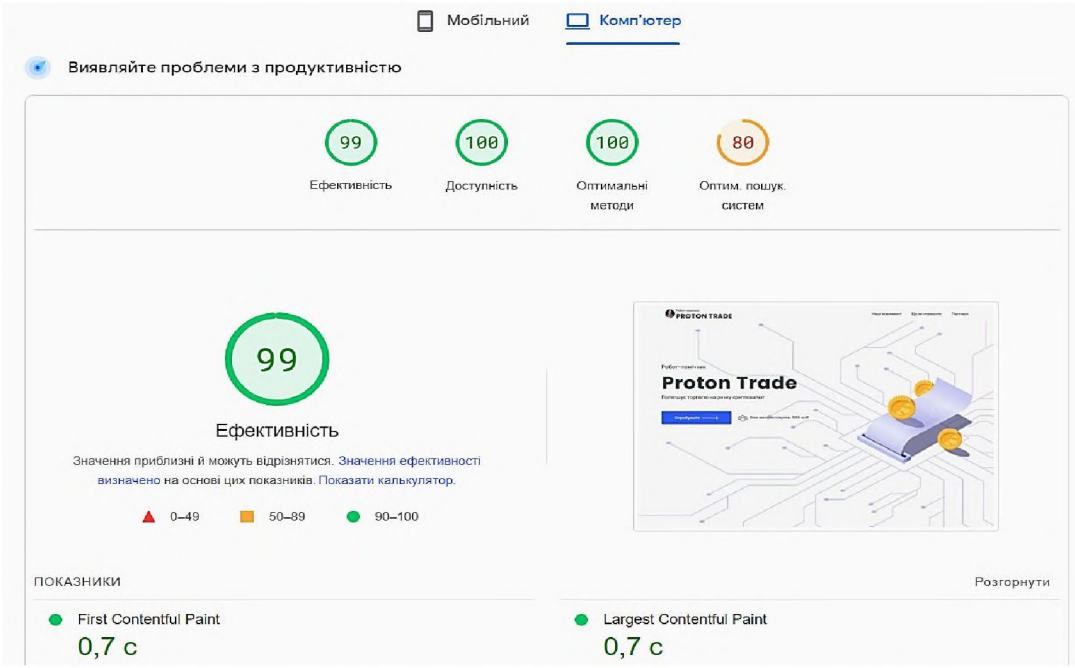


Рисунок 3.13 – Результати технічної вебаналітики сайту

3. Оптимізація контенту та ключові слова. Заголовок і підзаголовок: яскравий та змістовний заголовок, який привертає увагу користувачів, а також підзаголовок, який уточнює основне повідомлення (рис. 3.14).

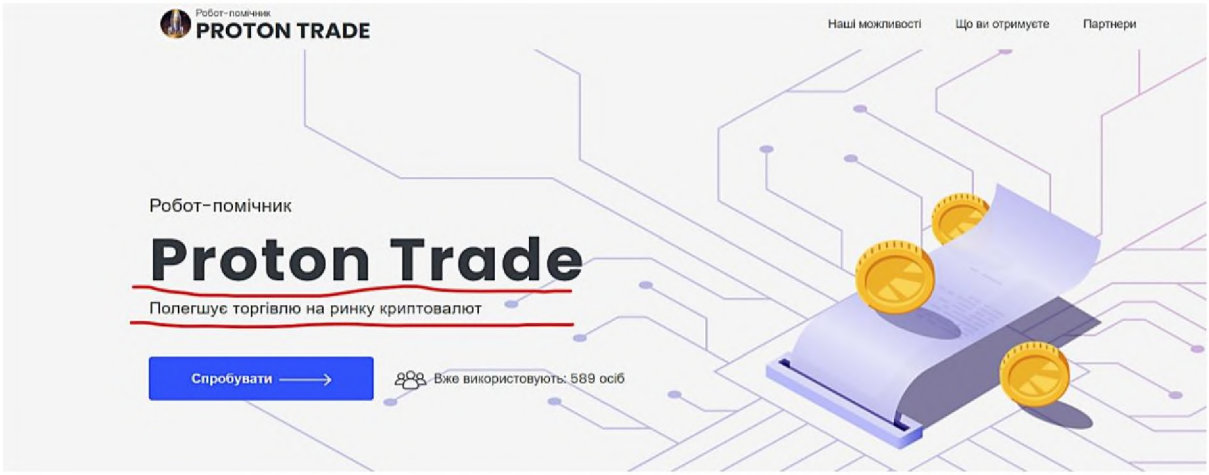


Рисунок 3.14 – Заголовки та підзаголовки на сайті

В основній частині опис трейд-бота включає текст про основні переваги та функціональні можливості Proton Trade. Пояснено, як він працює, які

стратегії використовуються та які можливості надає користувачам. В окремих блоках продемонстровано реальні результати торгівлі за допомогою трейд-бота (рис. 3.15). При цьому виведено тільки необхідну інформацію, виразні шрифти.

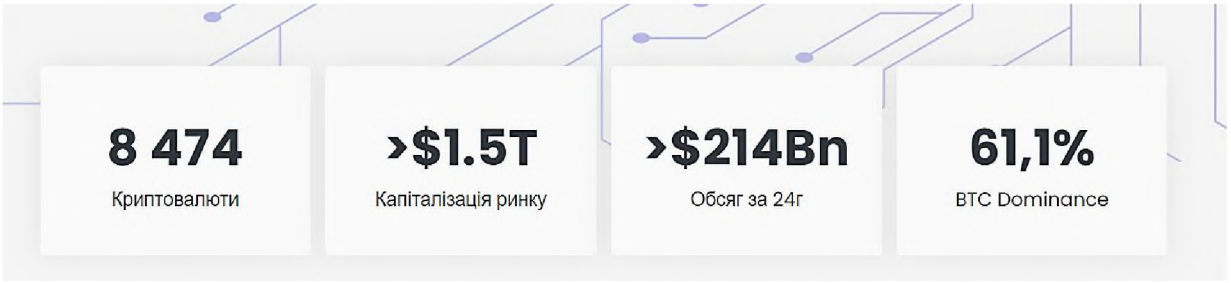


Рисунок 3.15 – Візуальні елементи для демонстрації результатів роботи бота

Так само, в розділі «Що ви отримуєте» наведено в аргументованій формі переваги використання трейд-бота Proton Trade (наприклад, див. рис. 2.18). Готовий код розробленого вебсайту розміщено в репозиторії GitHub [50].

3.3 Перевірка та опис економічної ефективності розробки

Перевірка та опис економічної ефективності розробки проєкту передбачає детальну оцінку різних аспектів для визначення його життєздатності та ефективності у створенні економічних переваг. Враховуючи, що більшість додатків все ж створюються з метою отримання прибутку, проведемо економічний аналіз, який включатиме: витрати на дизайн, розробку та різноманітні побічні витрати. Також врахуємо вартість хостингу, оскільки безкоштовний хостинг не є гарним рішенням для розміщення сайту на постійній основі (особливо сайту нашого типу).

Аналіз витрат. Для розглянутого проєкту розробки, що включає створення вебдодатку з використанням HTML5, CSS3, SCSS, JS, React та деяких прикладних програм, були враховані фактори витрат, описані нижче.

1. Інструменти та програмне забезпечення для розробки. Використовував Visual Studio Code як редактор коду, який є безкоштовним і

широко використовуваним інструментом у спільноті розробників. Він допоміг ефективно писати та керувати кодом.

2. Хостинг та інфраструктура. Firebase пропонує безкоштовний план хостингу, що дозволило мені розгорнути свій вебдодаток без жодних витрат на хостинг. Ця економічна опція забезпечила зручну та надійну платформу для розміщення додатку.

3. Сторонні бібліотеки та пакети. Ретельно відібрав бібліотеки та пакети з відкритим вихідним кодом для свого проєкту, що дозволило уникнути будь-яких витрат, пов'язаних з придбанням або ліцензуванням.

4. Витрати на персонал. Оскільки працював над проєктом самостійно, додаткових витрат на персонал не було. Загалом же використовується тариф на оплату праці програміста, вебдизайнера середнього рівня.

5. Витрати на інтеграцію та API. Розробляючи вебдодаток також врахував будь-які потенційні витрати, пов'язані з інтеграцією сторонніх сервісів або використанням зовнішніх API. Залежно від конкретних може виникнути потреба в інтеграції з платіжними шлюзами, постачальниками послуг або іншими зовнішніми системами.

Інтеграція з цими сервісами часто пов'язана з додатковими витратами, такими як комісія за транзакції, підписка або плата за використання. Важливо оцінити моделі ціноутворення та умови цих інтеграцій, щоб переконатися, що вони відповідають бюджету та вимогам проєкту. З вище описаних витрат, могли бути витрати тільки за оплату за замовлення та комісію.

6. Обслуговування та підтримка. При розрахунках економічної вартості потрібно врахувти потенційні майбутні витрати на обслуговування та підтримку вебдодатку. Це включає такі фактори, як поточне обслуговування сервера (за наявності), виправлення помилок і розширення функцій. Якщо використовувати проєкт для багатьох користувачів, то потрібно врахувати витрати на базу даних.

Враховуючи вищезазначені фактори і використовуючи безкоштовні ресурси, такі як Visual Studio Code, React, при підготовці роботи ефективно

керував витратами. Ця економічна ефективність дозволила зосередитися на успішному виконанні проєкту без фінансових обмежень. Загальна сума витрат становить:

$$В_{сум} = (З_{диз} + З_{прог} + З_{тест}) + В_{н} + В_{ін}, \quad (3.1)$$

де $(З_{диз} + З_{прог} + З_{тест})$ є сумою заробітних плат дизайнера, прогармістів;

$В_{н}$ – накладні витрати;

$В_{ін}$ – інші витрати (всі витрати, що не враховані в попередніх розрахунках).

Розрахунок складових витрат за формулою (3.1) наведено в табл. 3.1.

Таблиця 3.1 – Вартість робіт та передбачених витрат при розробці комерційного вебсайту, станом на початок 2024 р.

№ п/п	Найменування послуги (зміст)	Вартість, грн
1	Планування та дизайн. Збір вимог, структурування та UX дизайну	9217,00
2	Frontend розробка. HTML/CSS кодування, реалізація UI	18536,00
3	Backend розробка. Налаштування бази даних, логіка на стороні сервера	18536,00
4	Інтеграція API (платіжний шлюз, тощо)	3000,00
5	Тестування та QA. Забезпечення якості, виправлення помилок	9217,00
6	Впровадження та запуск. Налаштування сервера, реєстрація домену використовуючи Firebase	370,00
7	Поточне обслуговування. Оновлення, виправлення помилок, технічна підтримка (терміном дії на місяць)	7414,00
8	Сумарні витрати на розробку, встановлення та обслуговування комерційного вебдодатку	66290,00

Хоча етап розробки пройшов успішно, слід визнати, що в майбутньому можуть виникнути непередбачувані технічні проблеми. Ці проблеми можуть включати проблеми сумісності між різними версіями бібліотек або потенційні вразливості безпеки, які можуть з’явитися. Щоб зменшити ці ризики, необхідно залишатися в курсі останніх версій та проводити регулярний моніторинг.

У результаті докладених зусиль додаток було розроблено ефективно, з використанням економічно вигідних ресурсів та з дотриманням найкращих практик. Проактивно реагуючи на потенційні ризики, пом’якшені виклики і створено безпечний, масштабований і зручний вебдодаток.

Загалом, поєднання використання економічно ефективних ресурсів, проведення ретельної оцінки ризиків та впровадження відповідних заходів сприяло успішному завершенню проєкту з розробки комерційного вебсайту.

Висновки до розділу 3

У результаті роботи над останнім розділом, нами було спроєктовано макети сторінок вебдодатку за допомогою графічного редактору Figma. Після цього, базуючись на створених макетах, та поставлених вимогах у попередньому розділі було реалізовано складові частини додатку.

У повному обсязі було реалізовано клієнтський додаток, використовуючи технологію React та допоможні бібліотеки. Розроблена адміністративна панель, створено адаптив для мобільних пристроїв. Розроблені рішення тимчасово розміщено на безкоштовні хостинги dashboard.render та Vercel.

Проведено економічний аналіз, порахована орієнтовна вартість реалізації подібного проєкту, яка склала 66290 грн. Розглянуто можливість розміщення сайту на повноцінному хостингу. Для цього обрано хостинг ukraine.com.ua, та тариф «Кращий» вартістю 285 грн./місяць.

ВИСНОВКИ

Здійснивши аналіз сучасних програмних засобів вебтехнологій, можна сміливо визнати: галузь стрімко й різнопланово розвивається. В ході виконання кваліфікаційної роботи було проаналізовано велику кількість різноманітного інструментарію веброзробника. Вимоги до вебдодатків наростають надзвичайно швидко. Вебдодатки стають не просто елементом, а важливою складовою успішного бізнесу.

Було проаналізовано різноманітні вебсайти та систематизовано їх за характеристиками, призначенням та оптимальним стеком технологій. В результаті дослідження можна зробити висновок, що для кожної задачі потрібен свій інструмент та підхід, який враховує основні фактори дієвості, швидкості та раціональності створення, універсальних рішень не існує.

Описано та порівняно frontend-рішення, серед яких HTML-препроцесори (Pug, Haml), CSS-бібліотеки (Tailwind, Ant Design та інші). Особлива увага була присвячена JavaScript-фреймворкам. Було створено порівняльні таблиці популярних фреймворків React, Angular, Vue, Svelte. Досліджено особливості використання кожного з них. Універсальним рішенням виявився React.

Окремо розглянуті системи керування контентом (CMS): досліджено класифікацію, особливості застосування та галузі використання, для яких вони призначені.

Було також розглянуто backend-інструментарій: проведено порівняння популярних фрейморків (Laravel, Express, Django, Spring), досліджено особливості використання різних баз даних (SQL, NoSQL) та систем управління базами даних. Порівняно продуктивність та швидкодію популярних вебсерверів (Nginx, Apache). Кращі результати показав Apache. Розглянуто способи тестування та відладки серверної частини вебдодатку за допомогою спеціального програмного забезпечення (Postman, Insomnia).

Окрім цього був проведений аналіз допоміжного програмного забезпечення: системи контролю версій та хмарні репозитарії, інтегровані середовища розробки, пакетні менеджери, Docker та контейнеризація.

Проведено аналіз актуальних методологій проєктної діяльності та виділено переваги використання супровідного програмного забезпечення. Сучасна методологія Agile є чудовим вибором для впровадження в діяльність фірми, що займається розробкою вебдодтків. В якості програмного забезпечення можна використовувати Microsoft Project/Jira для масштабних, або Asana/Trello для середніх та малих проєктів.

Перед нами було поставлено задачу спроєктувати та розробити вебдодаток комерційного характеру. Для цього було виконано наступні кроки:

– обрано робочий стек технологій: для вибору найоптимальніших інструментів було проведено багатогранний аналіз переваг та недоліків різних продуктів. В результаті було сформовано наступний стек: Figma – графічний редактор, MongoDB – база даних, Express.js – backend-фреймворк, React – frontend-фреймворк, Git – система контролю версій, GitHub – хмарний репозиторій, Visual Studio Code – редактор коду, yarn – пакетний менеджер, – система автоматизованої збірки.

– Розроблено макети сторінок: продумано та реалізовано зовнішній вигляд всіх сторінок вебдодатку, на основі яких буде створено вже сам сайт;

– Створено серверну частину додатку: налаштована обробка всіх ендпоінтів, реалізована авторизація за допомогою JWT-токенів та можливість завантаження файлів;

– Створено клієнтську частину: на основі раніше створених макетів було розроблено клієнтський додаток. Була створена мобільна версія вебдодатку.

У результаті практичної частини був реалізваний увесь запланований функціонал.

На завершення було проведено економічний аналіз. Приблизна вартість впровадження такого проєкту становить 38730,6 грн. Вивчили можливість

розміщення сайту на повноцінному хостингу. Зробили вибір на користь хостингу ukraine.com.ua та тарифу "Кращий" за 285 грн. на місяць.

Подальше вдосконалення роботи можливо в напрямку розвитку підтримки SEO, оскільки React недостатньо з ним взаємодіє. Також можна працювати в бік розширення функціоналу. Прикладом може бути розробка модулю для тестування та опитування студентів, або блок з відеоматеріалами. Варто задуматись над додаванням в проєкт TypeScript, якщо буде плануватись подальше розширення функціоналу.