Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії)

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: «Удосконалення системи управління веб-сайтами, створеними на базі системи керування вмістом Wordpress»

Виконав: студент 2 курсу
групи ПІ-19мз
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)

Донченко В.В.
(прізвище та ініціали)

Керівник к.т.н., доц. каф. Коваленко О.О.
(прізвище та ініціали)
Рецензент д.т.н., проф. кафедри Василевський О. М.
(прізвище та ініціали)

Вінниця − 2021 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 - «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
"17" лютого 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Донченку Владиславу Вікторовичу

1. Тема роботи: «Удосконалення системи управління веб-сайтами, створеними = на базі системи керування вмістом Wordpress»
керівник роботи: Коваленко Олена Олексіївна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "09" березня 2021 року №64

2. Строк подання студентом роботи_____

3. Вихідні дані до роботи : Операційна система – Windows
Мови програмування – JavsScript
Бібліотеки –Vue.js

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задач дослідження; розробка методу, моделі та алгоритмів; розробка програмних засобів; тестування програми; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): технології; моделі системи; модель взаємодії процесів; розробка інтерейсу веб-додатку, тестування розробленої програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Коваленко О.О., к.т.н, доцент кафедри ПЗ		
5	Глущенко Л.А., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методів його вирішення	10.04.19 – 15.04.21	Вик.
2	Аналіз існуючих аналогів та постановка задач дослідження	16.04.19 – 23.04.21	Вик.
3	Розробка методу динамічного завантаження	24.04.19 – 30.04.21	Вик.
4	Розробка структур і алгоритмів програмного продукту	01.05.19 – 04.05.21	Вик.
5	Розробка програмного забезпечення	05.05.19 – 19.05.21	Вик.
6	Тестування розробленого програмного продукту	20.05.19 – 24.05.21	Вик.
7	Економічна частина	25.05.19 – 27.05.21	Вик.
8	Оформлення матеріалів до захисту МКР	27.05.21 – 01.06.21	Вик.

Студент _____ **Донченко В.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Коваленко О.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська кваліфіаційна робота спрямована на дослідження процесу удосконалення системи управління веб-сайтами, створеними на базі системи WordPress.

У магістерській кваліфікаційній роботі удосконалено систему управління веб-сайтами, на базі системи керування вмістом WordPress, шляхом розрозробки інтерфейсу та удосконаллення API на базі WHMCS. Розроблена система призначення для створення, редагування вмісту, встановлення SSL сертифікатів, отримання статстики, порад по оптимізації та безпеці роботи веб-сайтів, перестановка тем та плагінів.

Створений програмний продукт написаний на мові програмування JavaScript, з використанням фреймворку Vue.js. Веб-додаток являє собою SPA, тому програмний продукт, швидкий та зручний в користуванні та забезпечує користувачу досвід близький до користування настільного застосунка.

ABSTRACT

The master's qualification work is aimed at studying the process of improving the management system of websites created on the basis of the WordPress system.

The master's thesis improved the website management system, based on the WordPress content management system, by developing an interface and improving the API based on WHMCS. Developed a destination system for creating, editing content, installing SSL certificates, obtaining statistics, tips on optimizing and security of websites, permutation of themes and plug-ins.

The created software product is written in JavaScript programming language, using the Vue.js framework. The web application is a SPA, so the software product is fast and easy to use and provides the user with an experience close to using a desktop application.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Кожного дня попит на розробку сайтів зростає, разом з цим зростає необхідність у швидкому створенню адмініструванню та оптимізації сайтів, для того, щоб покращити взаємодію між сайтом та користувачем. На данний момент існує безліч АПІ, які дають можливість реалізувати це, проте не має сервісу, який міг би зібрати це все у одному місці та зпростити завдання власника сайту чи навіть десятків сайтів. Серед головних задач власника сайту можна виділити такі як можливість швидкого оновлення контенту, зміни акцентів на інформації відносно її актувальності; динамічного запровадження програмних засобів для оптимізації сайту та його безпеки. Серед відомих програмних додатків, що дозволяють виконати управління сайтом, його оновлення, надавати чіткі та якісні інструкції щодо виконання процесів управління, комплексних систем, які надавали б повний необхідний перелік функцій управління вмістом та можливості їх ефективної реалізації не виявлено. Кожна з систем має свої переваги та недоліки і потребує удосконалення. Платформа Wordpress є достатньо популярною серед власників та адміністраторів сайтів, але процеси встановлення сайту, формування порад щодо комунікацій та безпеки, оптимізації сайту, зміни контенту, отримання статистики з відвідуванням сайту та інших аналітичних даних потребує свого удосконалення.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження. Метою роботи є спрощення взаємодії між користувачем встановленням та управління сайтом на базі системи керування вмістом Wordpress.

У відповідності до поставленої мети потрібно виконати такі **задачі**:

- виконати аналіз необхідних вдосконалень в системі управління сайтом;
- розробити програмний модуль формування системи сайтів на базі системи керування вмістом Wordpress;

– розробити системи комунікацій з отримання порад по оптимізації та безпеки сайту;

– удосконалити програмний модуль встановлення плагінів на сайт;

– удосконалити програмний модуль отримання коментарів з усіх сайтів, якими управляє користувач;

– удосконалити програмний модуль отримання статистики з відвідування сайту;

– удосконалити програмний модуль отримання статистики з системною інформацією;

– сформувати архітектурну модель базі flux управління групою сайтів.

Виконання перелічених завдань у повному обсязі зможе продемонструвати можливості управління та створення сайтів на базі системи керування вмістом WordPress.

Розроблена система може бути впроваджена в організаціях, де вже використовуються сайти на базі системи керування вмістом WordPress, або для адмінів, які обслуговують групу сайтів.

Об’єктом дослідження є процес удосконалення управління сайтом на базі системи керування вмістом WordPress.

Предметом дослідження є програмні засоби створення та керування веб-сайтами на базі Wordpress.

Методи дослідження:

- аналізу для порівняння технологій та обгрунтованого вибору засобів створення та удосконалення програмних модулів;
- методи веб-програмування для розробки сучасних односторінкових веб-додатків;
- методи розробки сучасних додатків на базі flux-архітектури для створення архітектурної моделі програмного додатку управління вмістом;
- методи тестування для перевірки розроблених програмних модулів.

Наукова новизна одержаних результатів:

1. Подальшого розвитку дістав метод створення та редагування вмісту веб-сайтів на базі системи керування вмістом Wordpress, який за рахунок

управління низкою створених за різними технологіями сайтів та їх деталізації, дозволяє користувачеві підвищити продуктивність управління сайтами, адмініструвати декілька сайтів без погіршення якості управління та швидкості внесення змін.

2. Подальшого розвитку отримали модулі збору статистики та засоби оптимізації, комунікації з користувачем шляхом отримання візуальних зображень параметрів для низки сайтів, їх деталізації, формування відповідних порад з безпеки та оптимізації сайтів, що дозволяє користувачу отримати адаптовані дані для більш якісного управління сайтом.

Практична цінність отриманих результатів. Практична цінність результатів полягає у програмному продукті, що дозволяє спростити процеси користування та управління сайтами на базі системи управління керування Wordpress, удосконалити рівень обслуговування групи сайтів.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботах, опублікованих у співавторстві, здобувачу належить удосконалення створення сайтів на базі системи керування вмістом Wordpress, алгоритми роботи системи та приклади, що ілюструють її роботу по створенню та управлінню сайтами.

Публікації. За тематикою дослідження опублікована 1 наукова публікація у матеріалах наукової інтернет-конференції "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 59).Коваленко О., Донченко В. Системи управління якісними показниками контенту сайтів на базі системи керування вмістом Wordpress. Матеріали наукової інтернет-конференції «Інформаційне суспільство "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення" (випуск 59), 2021.

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану питання

Сьогодні веб-сайт потрібен кожній компнанії, не залежно від роду занять, не важливо, чи компанія займається роздрібною, оптовою торгівлею, чи надає послуги, тому що найшвидший та найбезпечніий спосіб купувати, дізнаватись нову інформацію, навчатись – це інтернет. Попит на веб-сайти зростає щороку, але разом і з цим виникає проблема в їх обслуговуванні та створенні, оскільки при створенні постає питання в використанні системи керування вмістом чи створення на базі сучасних фрейморків.

Створюючи сайт на CMS ми маємо такі переваги як:

- Швидкість створення. раніше створення сайту було надзвичайно тривалим і трудомістким процесом, повним проб і помилок. Сьогодні CMS позбавили вебмайстрів і розробників від маси непотрібних операцій. Максимальний рівень автоматизації, максимальна швидкість створення нових сайтів є головною перевагою таких систем.

- Поширеність CMS. Системи управління сайтом міцно лідирують в сайтобудівництва - переважна більшість веб-сайтів переходить на CMS, і тенденція ця зберігається. За останні роки близько 59% сайтів використовували Wordpress, далі з великим відривом йдуть Joomla (6,5%) і Drupal (4,8%). Продукт 1С-Бітрікс замикає сімку світових лідерів з показником 1,4% - це сотні тисяч сайтів.

- Простота розробки та підтримки. Завдяки ретельно продуманим функціоналом тієї ж 1С-Бітрікс, сьогодні не потрібно бути гуру програмування, щоб запустити найпростіший сайт. Зручний інтерфейс CMS і Google легко виведуть тямущого студента на правильну дорогу.

- Швидкий запуск сайту. Залежно від дизайну і рівня настройки, сучасні сайти на CMS можна запустити в рази швидше, ніж в минулому. Спрощують задачу розробників і дизайнерів готові шаблони (такі, як пропонуються для 1С-Бітрікс на «Маркетплейс»).

- Оновлення контенту. При використанні будь-яких систем управління сайтом на оновлення і редагування вмісту йде значно менше часу. Те ж стосується внесення дизайнерських змін. Будь то нові меню, заголовки і бічні панелі, будь-які корективи легко робляться через зручний і зрозумілий інтерфейс.

- SEO-оптимізація. CMS оптимізовані для пошуку - вони безперервно еволюціонують, щоб сьогодні запропонувати власникам, адмінам, веб-майстрам кастомізовані метадані і дають можливість настроювати URL-адреси. Додаткові плагіни взагалі розширюють арсенал SEO фахівців до небачених раніше меж.

Проте, сайти на базі CMS мають певні недоліки такі як:

- Низька безпека сайту. Поширеність CMS у всьому світі - це одночасно успіх і вразливість. Внутрішній устрій найбільш популярних систем вивчено хакерами уздовж і поперек, що робить ваш сайт потенційною жертвою чергового зловмисника.

- Однотипність сайтів. Зробити веб-сайт на CMS в точності таким, яким ви уявляєте в мріях, буде досить складно. Так, різні системи мають різну ступінь гнучкості, але абсолютно кожній властива деяка «шаблонність», а також свої «дитячі хвороби» - неможливість запровадження деяких функцій, низький рівень динамічності тощо.

- Повільне завантаження. CMS зберігає всі ресурси окремо, зіставляючи їх на льоту при зверненні веб-клієнта до певної сторінці. Це означає повільне завантаження. На щастя, проблему можна дещо пом'якшити шляхом використання Content Network Distribution (CDN).

Отже було визначено, що найпопулярнішою системою керування вмістом на якому створюються веб-сайти є WordPress, адже за допомогою такої CMS

створюються різноманітні сайти. Але така система має ряд недоліків, серед яких є незручність використання системи при обслуговуванні групи сайтів, має повільне завантаження та неефективну систему комунікацій допомоги користувачу. Саме тому, необідно створити додаткові програмні модулі для формування удосконаленого додатку управління вмістом сайтів та їх системою комунікацій.

1.2 Порівняльний аналіз аналогів

Проведемо аналіз декількох аналогів, що надають можливість керувати низкою сайтів на базі системи керування вмістом Wordpress.

InfiniteWP - потужний інструмент управління сайтом Вордпресс, оптимізований для агентств, розробників та фрілансерів.

Використовуючи цей ресурс можна безкоштовно керувати неограниченим кількістю сайтів. Тим не менше, у безкоштовній версії є обмеження. Наприклад InfiniteWP дозволяє створювати резервні копії сайтів за вимогою, але не додавати платне доповнення для зберігання їх у віддаленому сховищі.

Реальна потужність InfiniteWP відкривається з їх преміальним планом, що включає в себе функції: просто розробка сайту, мінливий веб-сайт, міграція, сканування вредоносних програм, моніторинг часу безотказної роботи, початкові клієнти, управління коментарями, публікація публікацій та сторінок, двофакторна аутентифікація, перевірка і робота неработающих інші функції.

Преміум-план дозволяє маркувати плагін своїм власним брендом, щоб клієнти могли переглянути ваш логотип замість InfiniteWP.

InfiniteWP не є SaaS-додатком. Замість цього він дозволяє вам встановити інструмент управління WordPress прямо на вашому сайті. Ви можете встановити його через плагін установщика, через cPanel або вручну завантажити його на свій сервер.

Після налаштування ви можете додати сайти, які хочете керувати, а потім встановити на них клієнтський плагін InfiniteWP. Він з'єднує ваші сайти з адміністративною панеллю InfiniteWP. Це дає вам неограничений контроль над веб-сайтами, якими ви керуєте.

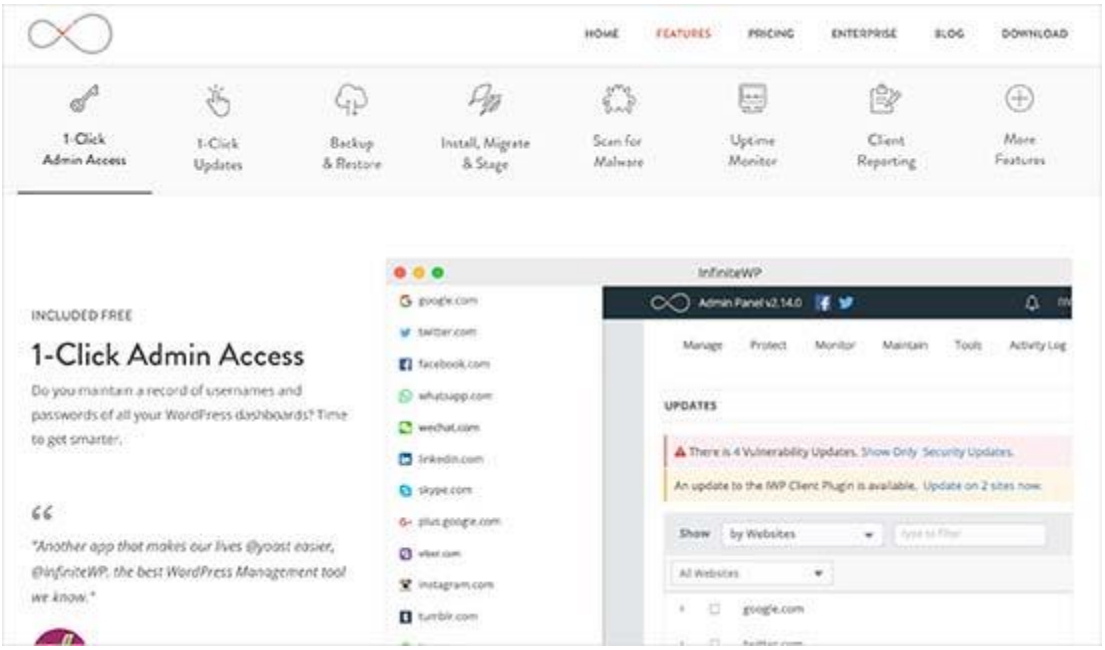

Рисунок 1.1 – Головна сторінка InfiniteWP

CMS Commander - платформний інструмент для управління сайтами Вордпресс. Управління сайтами відбувається з панелями управління CMS Commander, куди підключаються всі ваші сайти через встановлений на них плагін.

Інструмент пропонує всі звичайні функції, такі як оновлення в один клік, управління резервним копіруванням, двофакторну аутентифікацію, базовий сканер шкідливих програм, користувацький бренд та інші опції, що відрізняє CMS Commander від інших інструментів у списках, так це його функції управління контентом. Він дозволяє використовувати сторонні джерела, такі як YouTube, Flickr, Yelp та інші, для додавання контенту. Для аффілізованих маркетологів пропонується інтеграція з популярними партнерами, такими як Amazon, ShareaSale, Commission Junction та інші

Рисунок 1.2 – Головна сторінка CMS Commander

В таблиці 1.1. представлено порівняльні характеристики аналогу та функціоналу маййбутнього програмного продукту.

Таблиця 1.1 – Порівняльні характеристики аналогів

Критерій	Розроблений веб-сервіс	CMS Commander	InfiniteWP
Можливість зробити трансфер веб-сайту	+	-	-
Можливість отримати статистику по відвідуванню сайта	+	-	-
Можливість отримати статистику по залишиних коментарях по сайтах	+	-	-
Можливість встановлення плагінів	+	+	+
Можливість встановлення тем	+	+	-
Можливість управління контентом	+	+	-
Можливість управління резервними копіями	+	+	+

Таблиця порівняльних характеристик (табл.1.1) показала, що розробка сервісу удосконалення сайтів на базі системи управління вмістом Wordpress є доцільною. В результаті отримаємо продукт, який удосконалить існуючий функціонал, при цьому зробить інтерфейс біль зручним у користуванні.

1.3 Постановка задач роботи

Після аналізу розробленого додатка та порівняння аналогу було визначено, що розробка власного сервісу є доцільною. Врахувавши недоліки існуючих аналогів було визначено загальну структуру та архітектуру програмного додатку, що зможе не тільки забезпечити необхідний перелік можливостей, а й не матиме недоліків, які було визначено в ході аналізу існуючих аналогів.

Було визначено завдання, які необхідно виконати для розробки клієнтської частини веб-додатку:

- розробити схеми та алгоритми роботи сайту
- розробити інтерфейс програмного продукту, який буде легким й інтуїтивно-зрозумілим для користувачів
- розробити підсистему комунікації з серверною частиною через JSON-API за принципами REST
- удосконалити статистику відвідувань веб-сайту
- удосконалення отримання системної інформації про сайт
- розробити можливість отримати статистику по залишиних коментарях по сайтах
- розробити можливість встановлення плагінів
- розробити можливість додавання користувачів в адмін панель
- розробити можливість створення сторінок контенту
- розробити можливість встановити тем
- розробити можливість отримання порад про оптимізації

- розробити можливість отримання порад по безпеці сайта
- провести тестування продукту

1.4 Висновки

У першому розділі було розглянуто актуальний стан питання удосконалення веб-сайтів, на базі системи керування вмістом Wordpress, а також конкретні методи реалізації.

Також було проаналізовано перспективи вирішення даного питання шляхом розгляду аналога Hosting Ukraine, було визначенно недоліки та можливі переваги ісуючих реалізацій. Виконано аналіз можливих методів вирішення цього питання, з яких обрано розробку програми з використанням сучасних веб-технологій із монолітною архітектурою.

2. РОЗРОБКА МЕТОДУ, МОДЕЛІ ТА АЛГОРИТМІВ СИСТЕМИ

2.1 Вибір технологій реалізації компонентів системи

Для реалізації веб сервісу було обрано технології, які в повній мірі задовольняють технічні та функціональні характеристики програмного продукту.

Було вирішено, що веб-сервіс буде реалізований як набір SPA - це веб-застосунок чи веб-сайт, який вміщується на одній сторінці з метою забезпечити користувачу досвід близький до користування настільною програмою.

В односторінковому застосунку весь необхідний код - HTML, JavaScript, та CSS - завантажується разом зі сторінкою, або динамічно довантажується за потребою, зазвичай у відповідь на дії користувача. Сторінка не оновлюється і не перенаправляє користувача до іншої сторінки у процесі роботи з нею. Взаємодія з односторінковим застосунком часто включає в себе динамічний зв'язок з веб-сервером. Термін "односторінковий застосунок" був вигаданий Стівом Еном у 2005 році, хоча концепція обговорювалась ще на початку 2003 року, також Стюарт (stunix) Морис описав "автономний веб-сайт" (Self-Contained website) з такими ж цілями та функціоналом в 2002 році та пізніше у цьому році Лукас Бердо, Кевін Гекмен, Майкл Пічі та Еван Є зробили опис у патенті США 8,136,109. Javascript може бути використаний у веб-браузері для відображення інтерфейсу користувача, виконання логіки застосунку, та взаємодії з веб-сервером. Нативні бібліотекі з відкритим кодом доступні для побудови односторінкових застосунків, зменшуючи код, який потрібно написати розробнику Javascript.

JavaScript – мова програмування, яка виконується у браузері й дозволяє керувати властивостями відображуваного документа, інтерактивно змінювати їх, реагувати на дії користувача, виконувати запити до серверу, тощо. Довгий час керувати поведінкою сторінки у браузері можна було лише з використанням

JavaScript. Проте, сучасні браузери також підтримують технологію WASM. З її допомогою можна виконувати у браузері код, написаний на будь-яких інших мовах програмування, таких як C++, C# чи Rust. Ця технологія досить нова та має певні обмеження, такі як неможливість прямої взаємодії з DOM [8]. Тому її використання доцільно коли потрібно виконувати складні обрахунки на стороні користувача, а для програмування простої взаємодії з сторінкою продовжується переважне використання JavaScript.

Для реалізації сервіса було обрано фреймвор Vue.js. Vue.js (читається як "в'ю", з англ. view) — JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Коли Еван Ю працював в Google Creative Labs, в нього виникла необхідність швидко побудувати прототип складного інтерфейсу, і потрібен був інструмент щоб уникнути написання повторюваного HTML. React лише починався, AngularJS та Backbone були занадто громіздкі для прототипування, тому Еван створив свій фреймворк. З того часу Vue.js еволюціонував, і дозволяє писати не тільки прототипи, а й складні веб-застосунки.

Оригінальний реліз Vue відбувся в грудні 2014 року. Інформація про проект було розміщено на Hacker News, Echo JS, та the /r/javascript підкатегорії в день початкового релізу. За один день проект з'явився на перших сторінках цих сайтів.Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

В Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто

замініть шаблон на рендерингову функцію. Рендерингова функція відкриває можливості для потужних патернів базованих на компонентах - для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині.

Одна із найвиразніших особливостей Vue — це ненав'язлива реактивна система. Моделі це просто плоскі JavaScript об'єкти. Це робить керування станами дуже простим та інтуїтивним. Vue надає оптимізований ре-рендеринг з коробки без потреби робити що-небудь додатково. Кожен компонент слідкує за своїми реактивними залежностями під час рендерингу, тому система знає точно коли має відбуватись ре-рендеринг і які компоненти потрібно ре-рендерити.

Vue надає різноманітні шляхи для застосування ефектів переходу, коли елемент додають, оновлюють або видаляють з DOM. Наприклад:

- автоматичне застосування класів для
- інтегрування сторонніх бібліотек для CSS анімацій, таких як Animate.css
- використовувати JavaScript для прямих маніпуляцій з DOM під час переходів
- інтегрування сторонніх JavaScript бібліотек анімацій, таких як Velocity.js

Vue сам по собі не включає роутингу, та є vue-router пакет, який вирішує це питання. Він підтримує зв'язування вкладенних шляхів з вкладеними компонентами і пропонує деталізований контроль над переходами. Vue дозволяє створення додатків за допомогою компонетів. Якщо додати vue-router до цього, все що потрібно зробити це зв'язати ваші компоненти з роутами і дозвольте vue-router вирішувати де їх рендерити.

Користувач взаємодіє з веб-сайтом або веб-застосунком через інтерфейс у браузері. Класичним методом створення веб-інтерфейсів є використання HTML для визначення вмісту веб-сторінок, CSS для визначення презентації веб-сторінок та JavaScript для визначення поведінки веб-сторінок [6]. Браузер будує

сторінку на основі цих інструкцій, що повністю описують її вигляд та поведінку. Основними вимогами до веб-інтерфейсів є їх однаковий зовнішній вигляд і однакова функціональність при роботі в різних браузерах, зовнішня привабливість, адаптивність до різних розмірів екрану.

Hypertext Markup Language використовується для розбиття інформації, що відображається на сторінці, в послідовність елементів, таких як [7]:

- заголовки, абзаци, списки, таблиці;
- зображення, аудіо, відео чи інші типи медіа;
- гіпертекстові посилання на інші документи;
- інтерактивні форми, кнопки, поля для введення інформації.

Cascading Style Sheets являє собою правила, що застосовуються до елементів HTML і керують їх візуальним представленням – колір, розмір, вирівнювання, шрифт, відступи, рамки, видимість та інші [7]. Правила можуть застосовуватись як до одного елемента за його ідентифікатором, так і до групи елементів за їх класом або назвою тегу. Проте, спосіб використання данного способу є застарілий, так як не задовольная компонентній структурі, крім цього портібно придумувати класси для розмітки та розробникові доведеться стежити за різноманітністю та неповторюваність классів, для того щоб уникнути конфліктів у подальшому було обрано StyledComponents.

StyledComponents - це один із нових способів використання CSS у сучасному JavaScript. Він призначений бути наступником модулів CSS, способом писати CSS, який має масштаб до одного компонента, а не витікати до будь-якого іншого елемента на сторінці.

Для розробки серверного застосунку використовують багато різноманітних технологій і мов програмування. Така програма повинна приймати HTTP-запити від клієнтів, видавати їм HTTP-відповіді, зазвичай разом зі сторінкою, що містить дані для відображення. Організацією і зберіганням даних на сервері займається система управління базою даних, яка надає можливості створення, збереження, оновлення та пошуку інформації з

контролем доступу до неї. Для роботи данної системи необхідно API, яке було удосконалене на основі WHMCS API. WHMCS - універсальне рішення для автоматизованого управління рахунком, клієнтами та їх підтримкою для онлайн-бізнесу найрізноманітніших типів. Розробкою WHMCS займається приватна компанія, розташована у Великобританії (компанія № 06265962). Зявився цей продукт у 2005 році, як одна з перших спроб об'єднати автоматизовану систему навчання та управління клієнтами та систему підтримки одночасно в єдиному цілому. З того моменту це було зроблено багато, як в одному, так і в інших облаштуваннях і сьогодні можна сміливо затвердити, що WHMCS є одним із найпотужніших та найпопулярніших рішень серед компаній, що надають послуги хостингу і не тільки. WHMCS є найпопулярнішою системою навчання клієнтів та підтримки у світі серед хостинг-провайдерів (і не тільки серед них). Ця система використовується в більш ніж 90 країнах світу.

Усі основні цілі та завдання розробки та вдосконалення WHMCS формуються на основі відгуків та бажаних її користувачів.

Оновлення з новими функціями та вдосконаленнями виходять із частотою не реже одного разу в 3 місяці.

WHMCS володіє величезним співтовариством, завдяки котрому на сьогоднішній момент розроблено загальноширше число модулів розширення функціональних продуктів.

До появи першого сертифікованого реселлера в Рунете кваліфікована технічна підтримка показує лише розробники та лише на англійській мові. Однак тепер ви можете розробити на кваліфікованій російській підтримці та бути впевненими, що всі питання, пов'язані безпосередньо з технічною стороною, не будуть лише викладені без уваги, але і будуть створені безпосередньо професійними командами розробників WHMCS. Крім того, наша компанія працює з цими розробниками, тому всі ваші побажання та запрошення будуть розглядатися одними з перших.

2.2 Розробка моделей сторінок веб-сайту та інтерфейсів користувача

Моделі веб-сайту формуються як окремі веб-сторінки , кожна з яких має свій функціонал і представлення основних функцій та змісту.

Сторінка аутентифікації користувача (рисунок 2.1) дозволяє виконати такі дії:

Користувачу сервісу потрібно ввести дані акаунта, які були створенні на сторінці створення акаунта, або авторизуватись на сторонньому сервісі Github, Gmail, Facebook, для того щоб мати змогу користуватись веб-сервісом. Також є можливість відновити акаунт, якщо користувач забув пароль, проте цей акаунт повеннен бути створеним у самій системі.

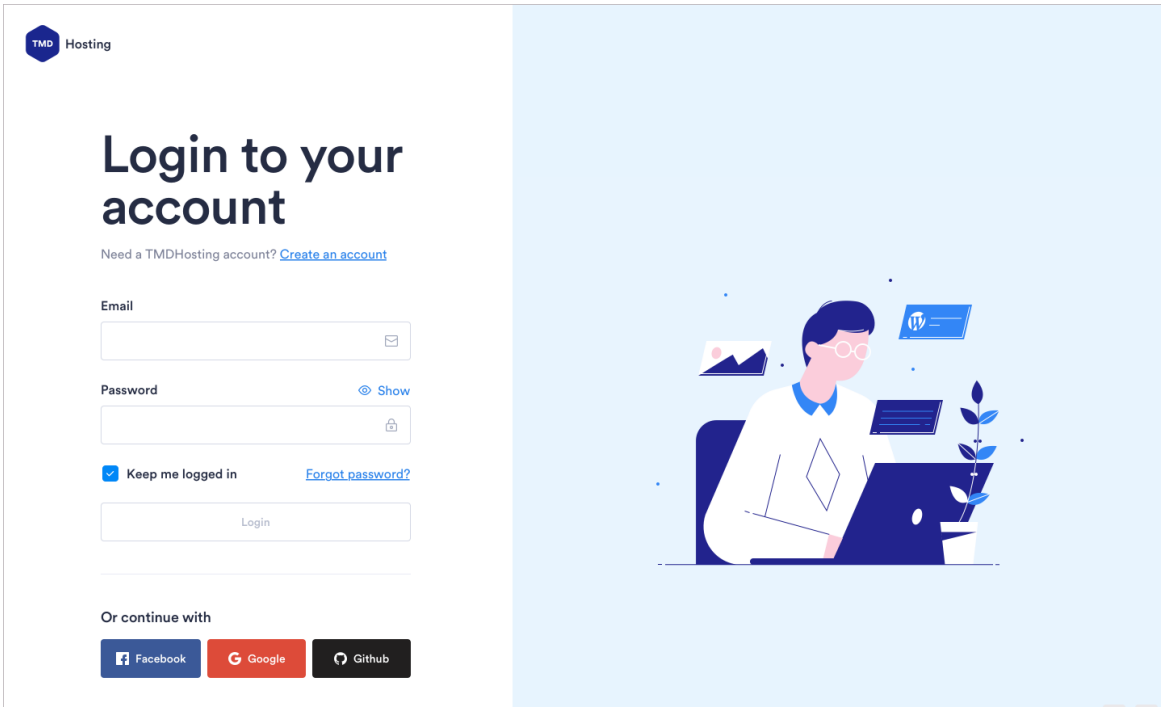

Рисунок 2.1 – Сторінка входу користувача

Панель приладів повинна містити основні показники, можливість їх розшифрування та зміни адміном.

Сторінка панель приладів (рисунок 2.2) – це головна сторінка веб-ресурсу після входу користувача. Вона дозволяє змінювати плагіни сайтів, дивитись

повідомлення, які прийшли на кожен веб-сайт, також там сформована статистика за такими показниками як унікальні відвідувачі, загальна кількість запитів, відсоток кешу та загальна кількість поданих дани. Якщо у користувача не має жодного веб сайту, тоді у на головній реалізується інтерфейс за допомогою якого він може це легко зробити.

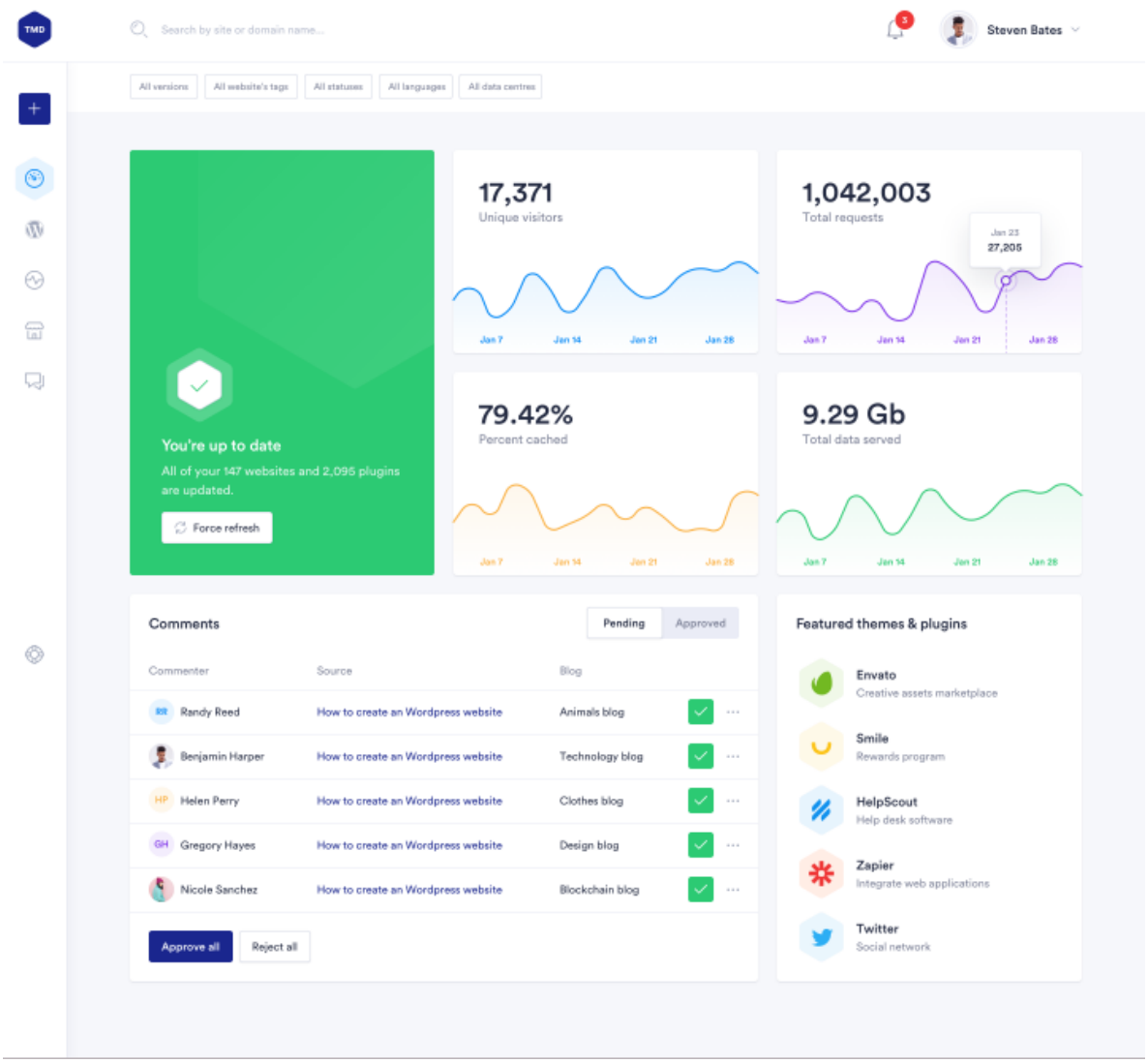

Рисунок 2.2 – Панель управління

Сторінка Управління веб сайтами створених чи перенесених із інших хостингів передбачає використання окремих блоків, які містять інформацію про кожен веб-сайт і також при натисканні на копку Manage Website, користувач

переходить в на сторінку управління визначеним сайтом, крім цього є можливість сортування сайтів.

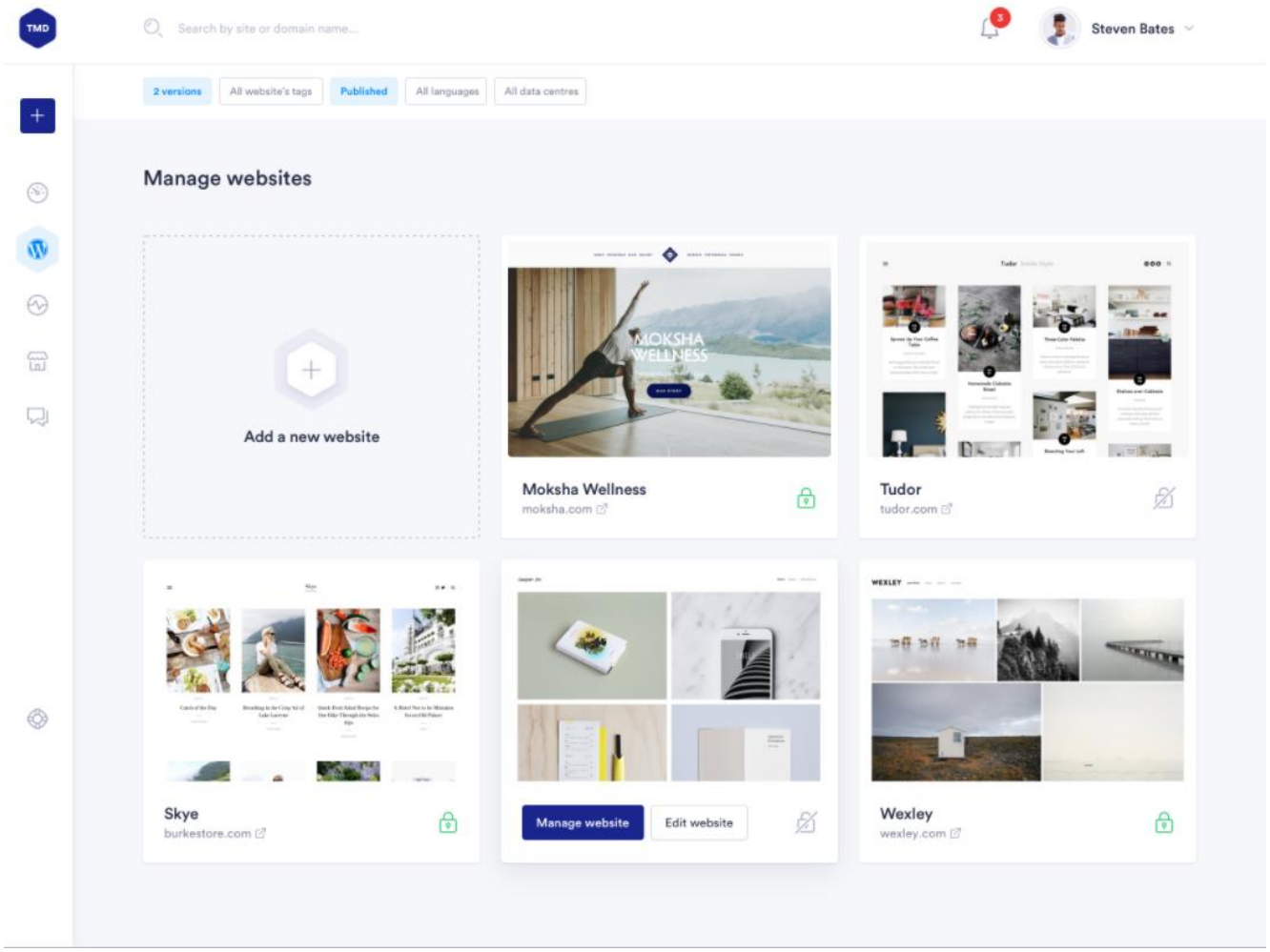

Рисунок 2.2 – Управління веб сайтами

Сторінка аналітики – поділенна на дві вкладки, перша – це веб-аналітика, друга – це ресурсна аналітика.

Ресурсна аналітика (рисунок 2.3) –містить сторінки графіки з інформацією про використання ресурсів, таких як використання процесора, використання пам'яті, стан процесів, використання дискового вводу-виводу, стан операцій вводу-виводу диска. На данній сторінці користувач отримує статистику по усім сайтам.

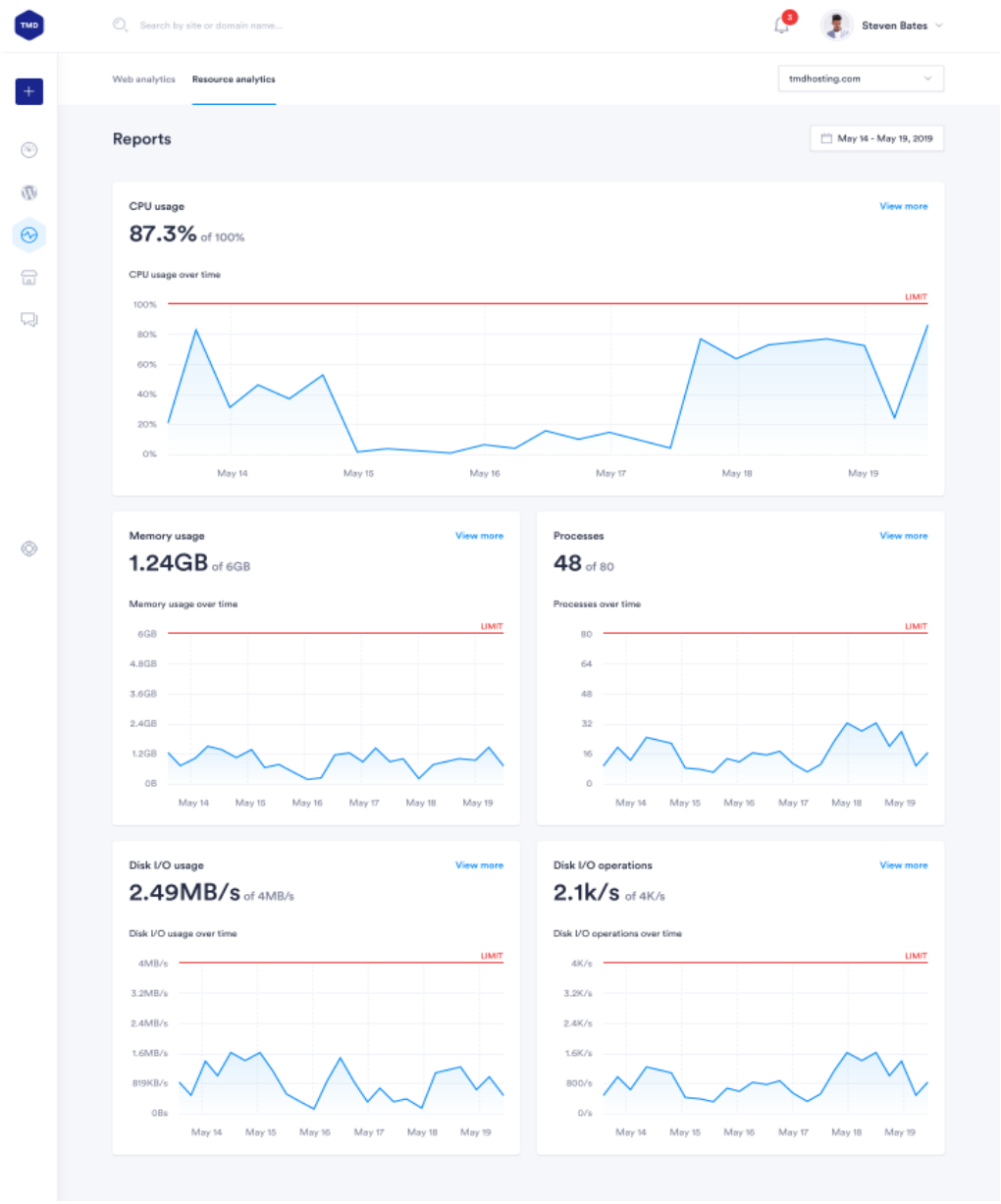

Рисунок 2.3 – Ресурсна аналітика

Веб аналітика (рисунок 2.4) – являє собою графіки з інформацією про кількість відвідувань, з інформацією про кількість унікальних відвідувачів, середньостатистичний час, який проводять відвідувачі.

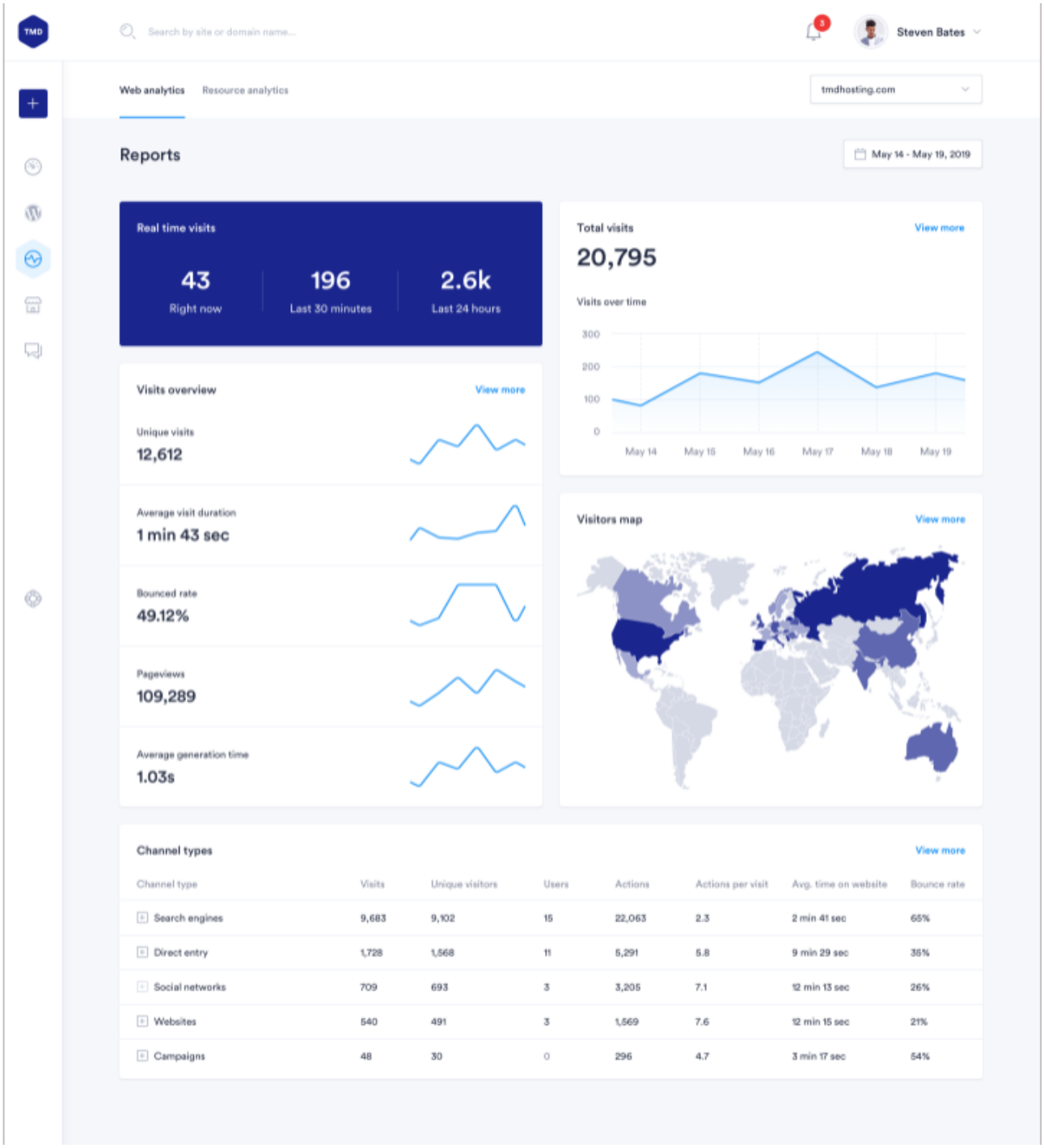

Рисунок 2.4 –Веб аналітика

Сторінка створення веб-сайту – на сторінці користувач пошагово створює сайт. На першому етапі користувач оберає між існуючими доменами, або створює власний, під час створення власного домена, користувач вводить домене

ім'я, а система в списку виводить доступні для придбання, після чого користувач перейде на інший крок (рисунок 2.5).

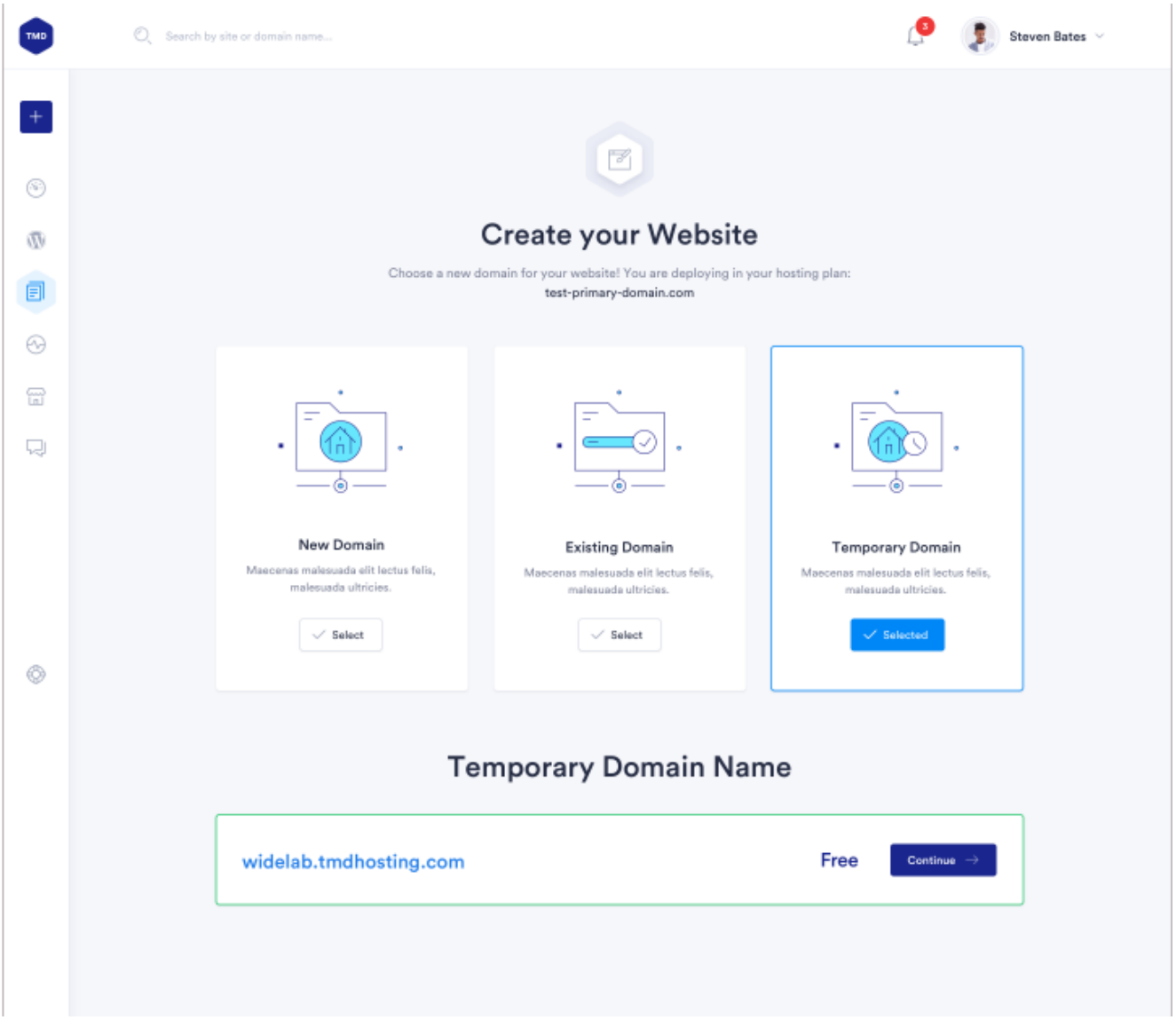

Рисунок 2.5 – Сторінка створення веб-сайту (Крок 1)

На другому етапі, користувач обирає спосіб створення веб-сайту на базі Wordpress, серед можливих варіантів: Drag and Drop Builder, побудова на базі Woocomerce, та звичайних Wordpress. Drag and Drop Builder, тобто користувач крім звичайного блога, чи сайту новин, може створити повноцінний інтернет-

магазин в декілька кліків чи просто звичний для постійних користувачів Wordpress (рисунок 2.6).

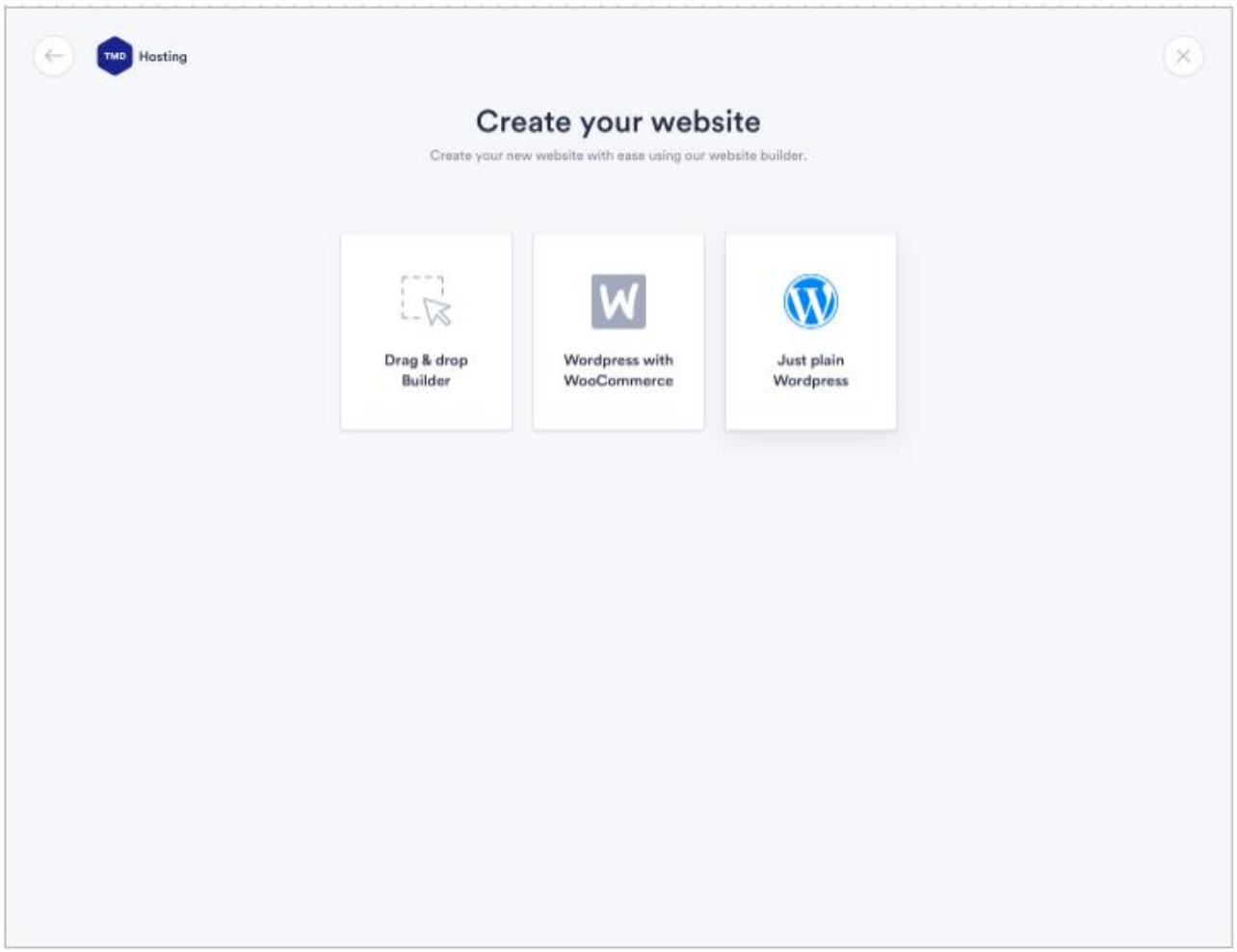

Рисунок 2.6 – Сторінка створення веб-сайту (Крок 2)

На третьому етапі, користувач обирає тему вебсайту (рисунок 2.7). На сторінці також присутній пошук, який допомагає користувачеві швидко знайти потрібну в списку тему, або відфільтрувати за категоріями, крім цього можна подивитись теми у режимі попереднього перегляду –цей режим допоможе з’ясувати, як буде виглядати його сайт у поточному оформленні в режимі декстоп, та мобільній версії, при натисканні кнопки яка визначає тему сайту, користувач атомматично переноситься на сторінку.

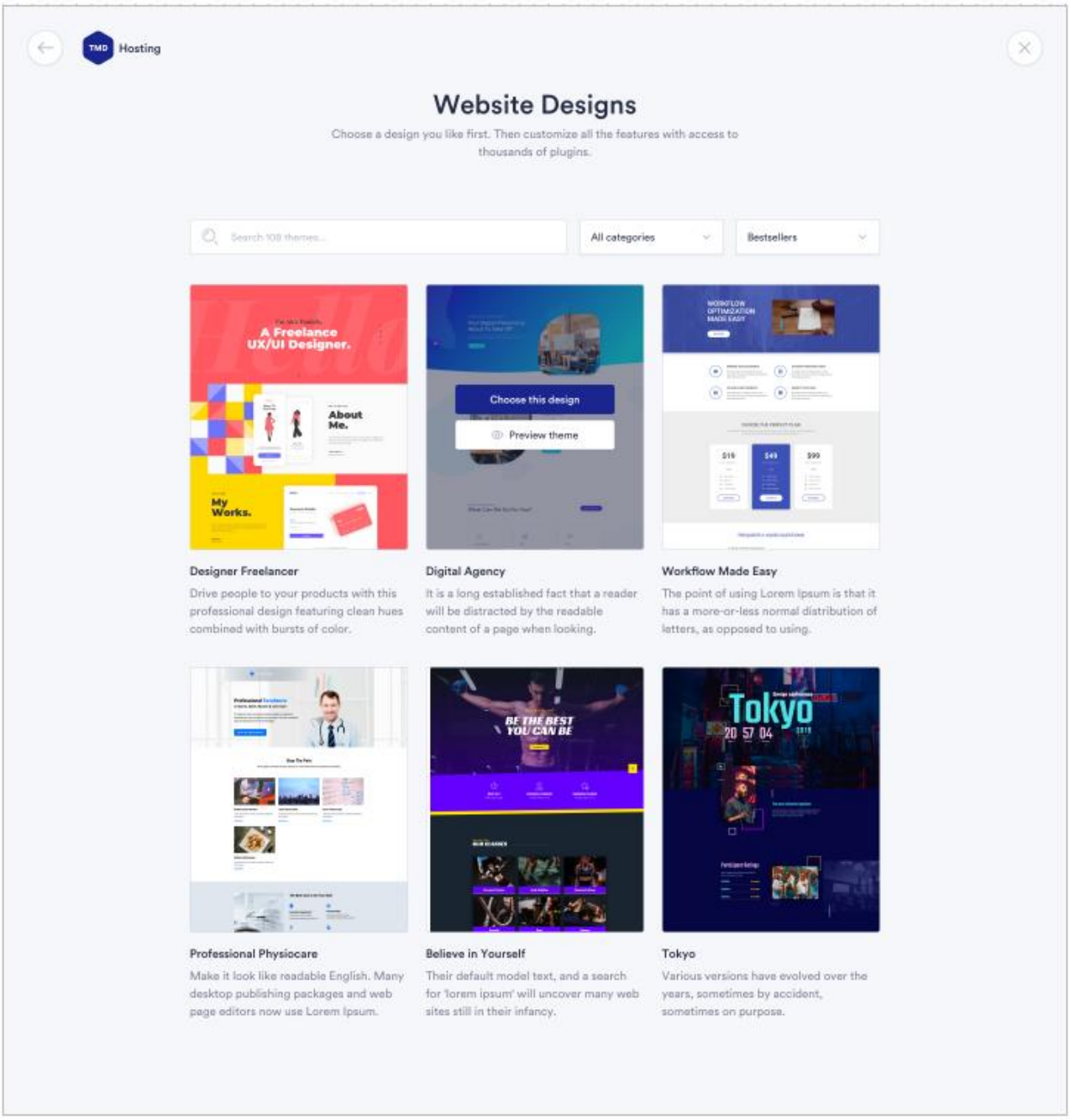

Рисунок 2.7 – Сторінка створення веб-сайту (Крок 3)

На четвертому етапі користувач заповнює почергово форму, яка в свою чергу розділенна на чотири етапи, на першому користувач формує базові відомості про сайт – такі як назва, дані користувача, поточний домен та оформлення сайту будуть підтягнуті з попередніх кроків, далі користувач

обирає, потрібні плагіни для сайту, далі буде оплата та публікація сайта, який користувач сконструював (рисунок 2.8).

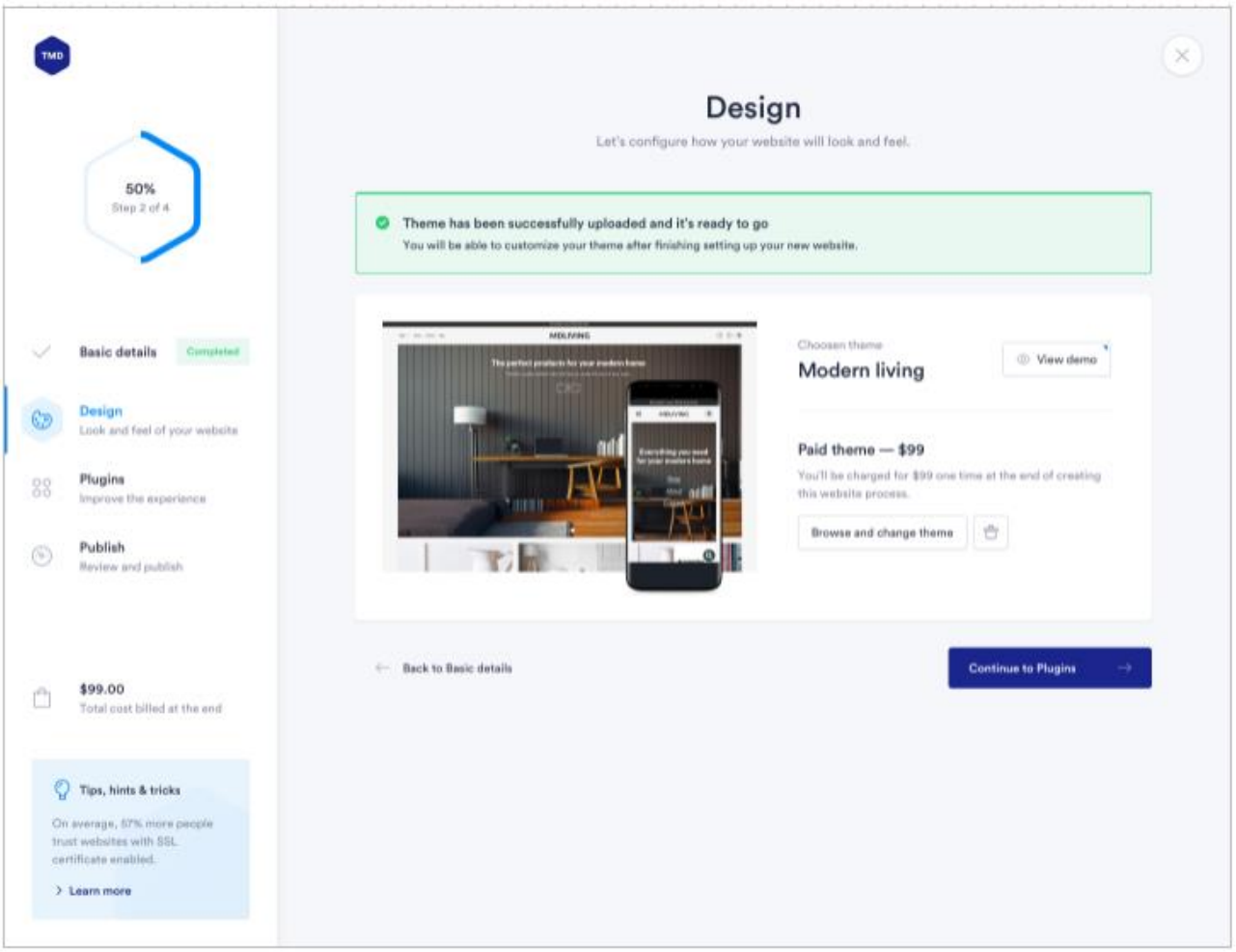

Рисунок 2.8 – Сторінка створення веб-сайту (Крок 4)

Після створення, такого веб сайту можна перейти безпосередньо до його управління, одразу при переході користувач потрапляє на сторінку зі вкладками, активна вкладка – панель приладів. Ця сторінка аналогічна до головної сторінки додатку, проте вона містить інформацію конкретно про створений сайт, його швидкість, резервні копії, плагіни для оновлення та усі наявні коментарі по сайту, з можливістю оновити усі або частково визначені, також є можливість

відредагувати веб-сайт, перейти на нього та дізнатись наявність SSL сертифікату (рисунок 2.9).

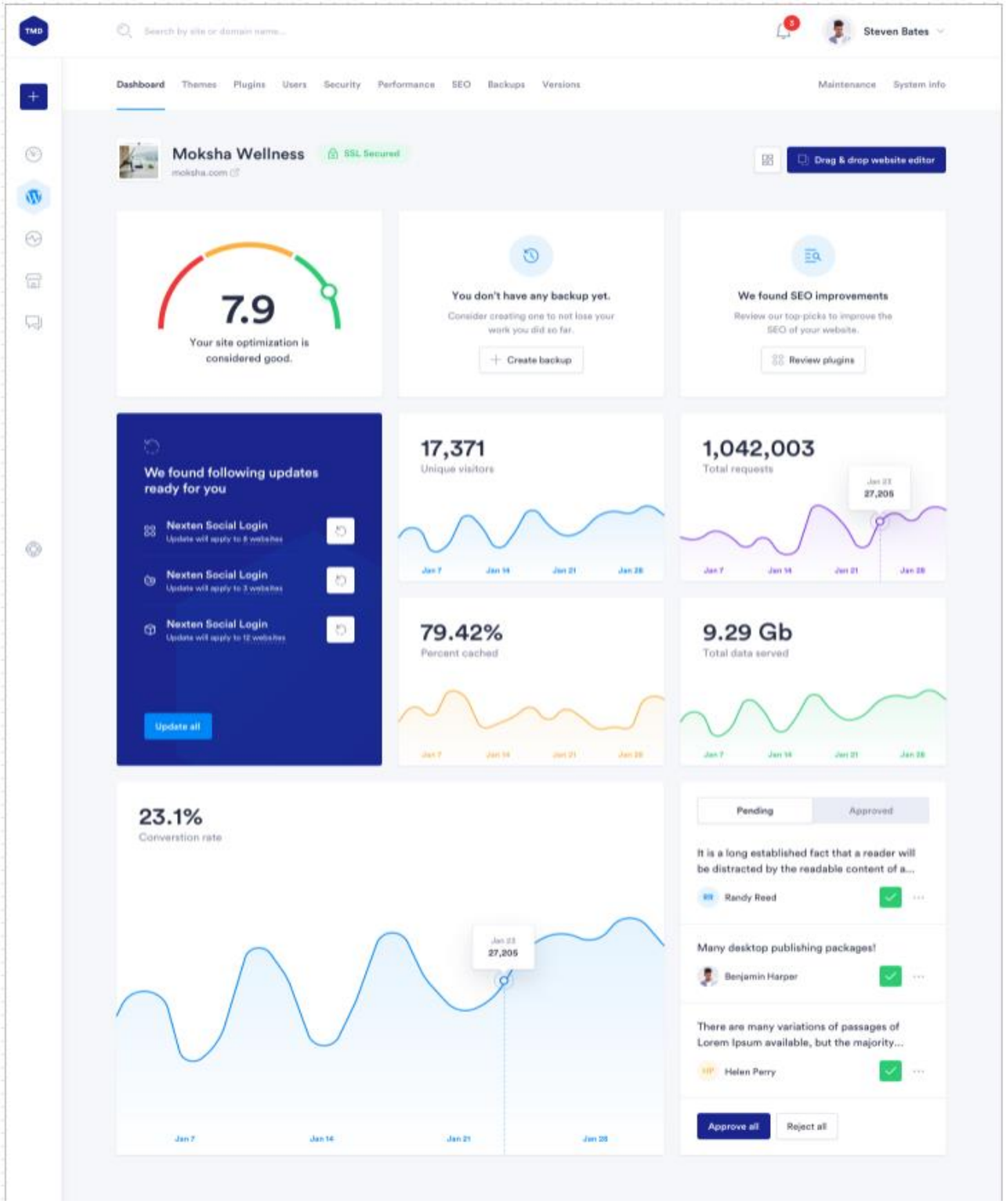

Рисунок 2.9 – Сторінка панель приладів, створенного веб-сайту

Вкладка теми – користувач має змогу редагувати обрану тему, або обирає нову серед доступних, крім цього є можливість перейти до каталогу тем та вибрати інше формлення веб сайту, з можливістю пепереднього перегляду (рисунок 2.10).

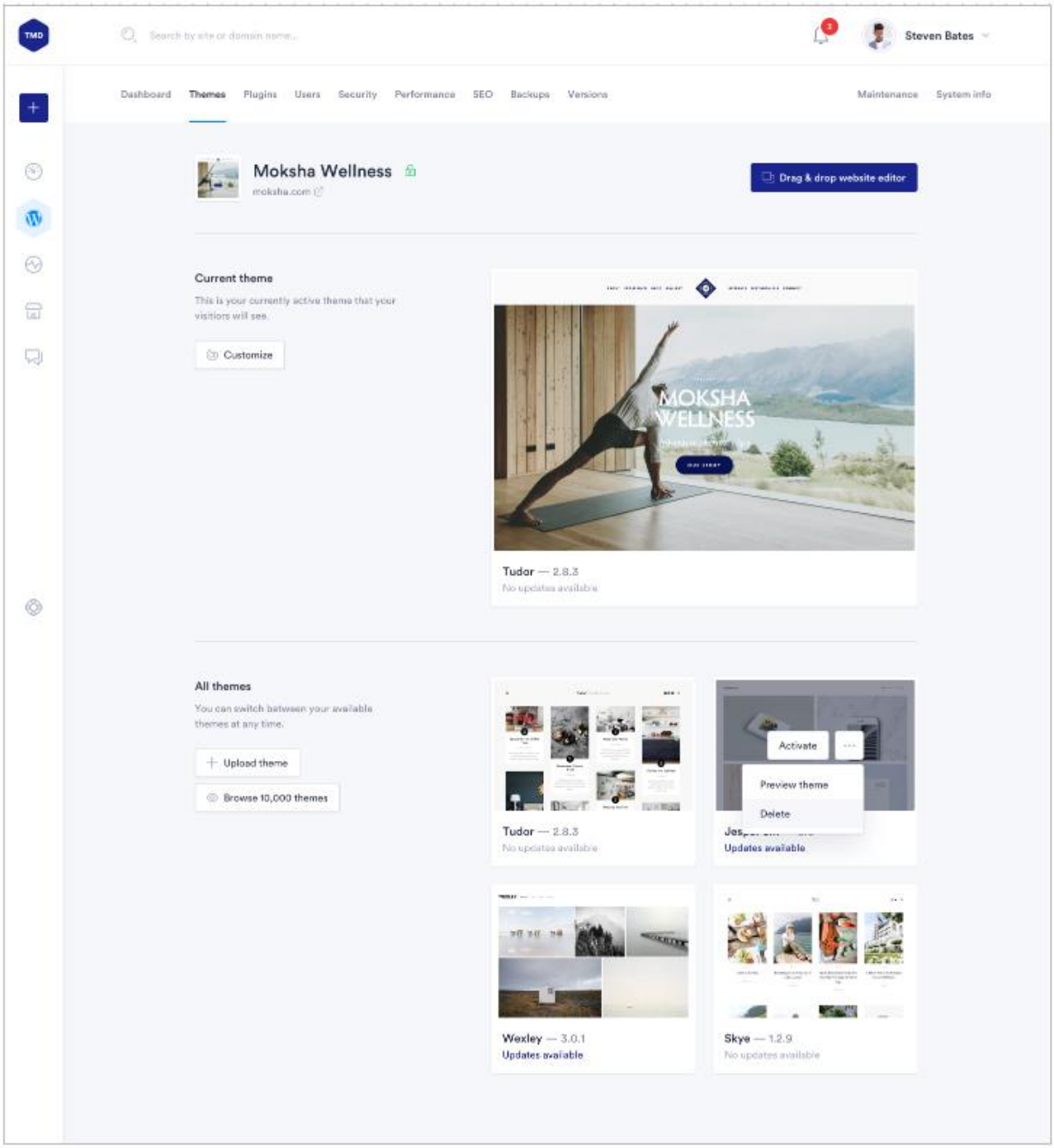

Рисунок 2.10 – Сторінка тем для сайту

При переході до вибору тем, користувач потрапляє до каталагу із фільтрацією та пошуком (рисунок 2.11).

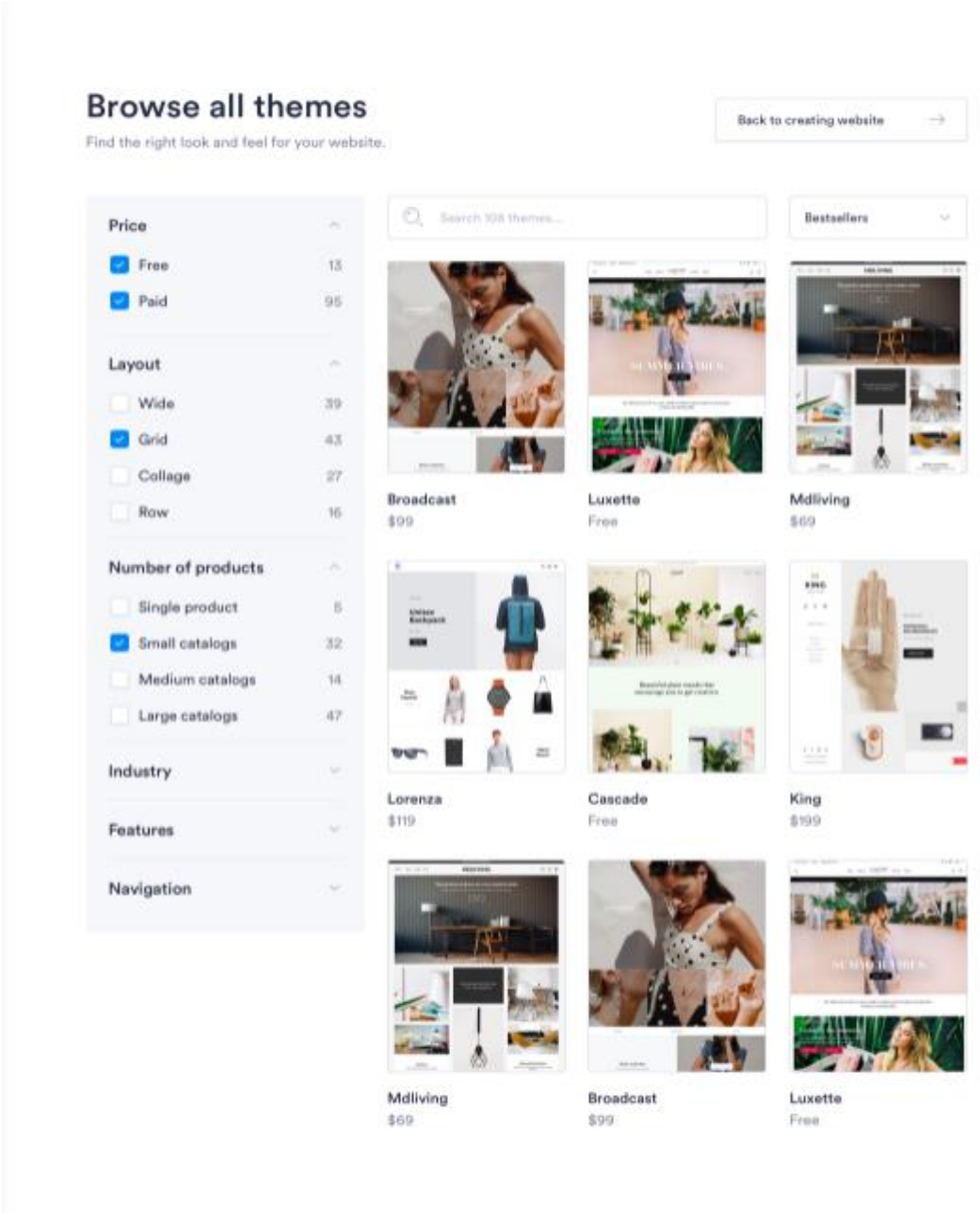

Рисунок 2.11 – Сторінка тем для сайту (Каталог)

Вкладка плагіни – дозволяє включати та виключати уже встановлені, які відображаються у списку, також є можливість перейти до каталогу плагінів.

Каталог при доданні плагін автоматично встановлюється, та додається в поточний список встановленних, крім цього існує можливість видалення (рисунок 2.12).

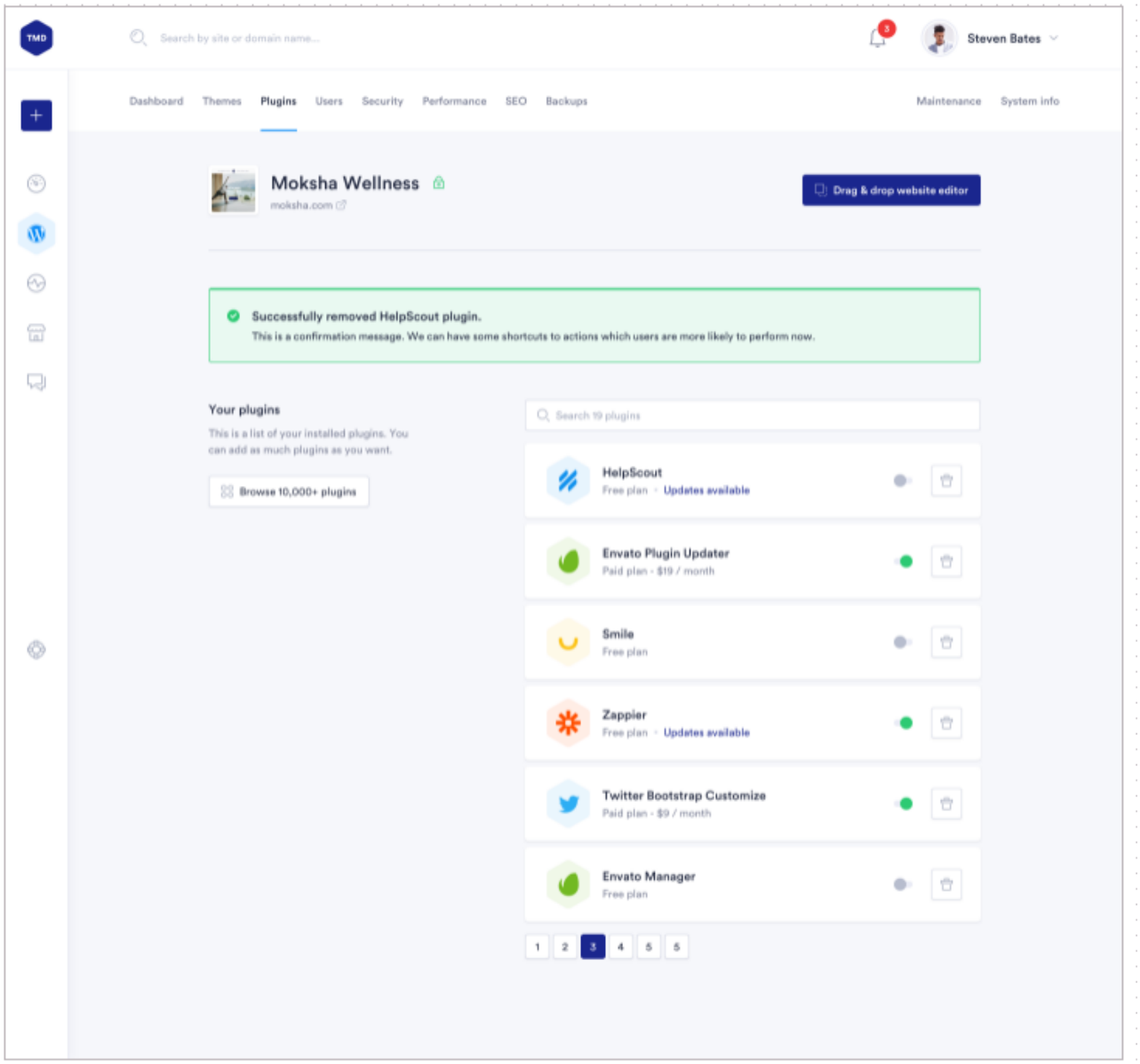

Рисунок 2.12 – Сторінка плагінв для сайту

Вкладка безпека – на цій вкладці користувач, натиснувши кнопку перевірити веб-сайт, отримує перелік можливих проблем з сайтом, які потребують вирішення, поради розділені по пріоритетності, для більш

зрозумілого сприйняття, виділені зеленим, жовтим та червоним кольором, відвідно загрози чи виконаних вимог (рисунок 2.13).

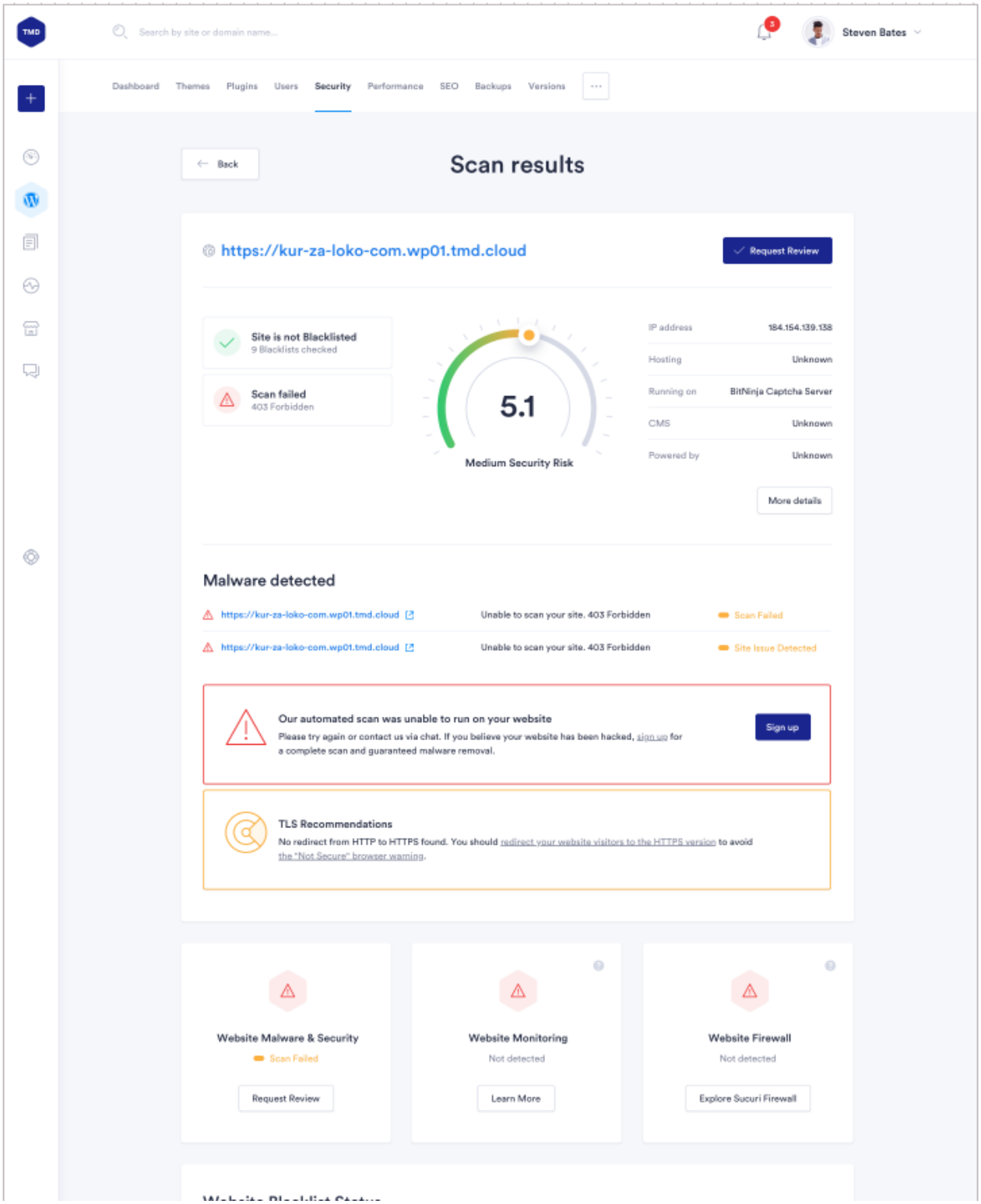

Рисунок 2.13 – Сторінка плагінв для сайту

Вкладка продуктивність – інтерфейс у вигляді окремих вкладок, на кожній з яких відображається, коротка інформація про кожне сканування з можливістю адміністрування та очищення (рисунок 2.14).

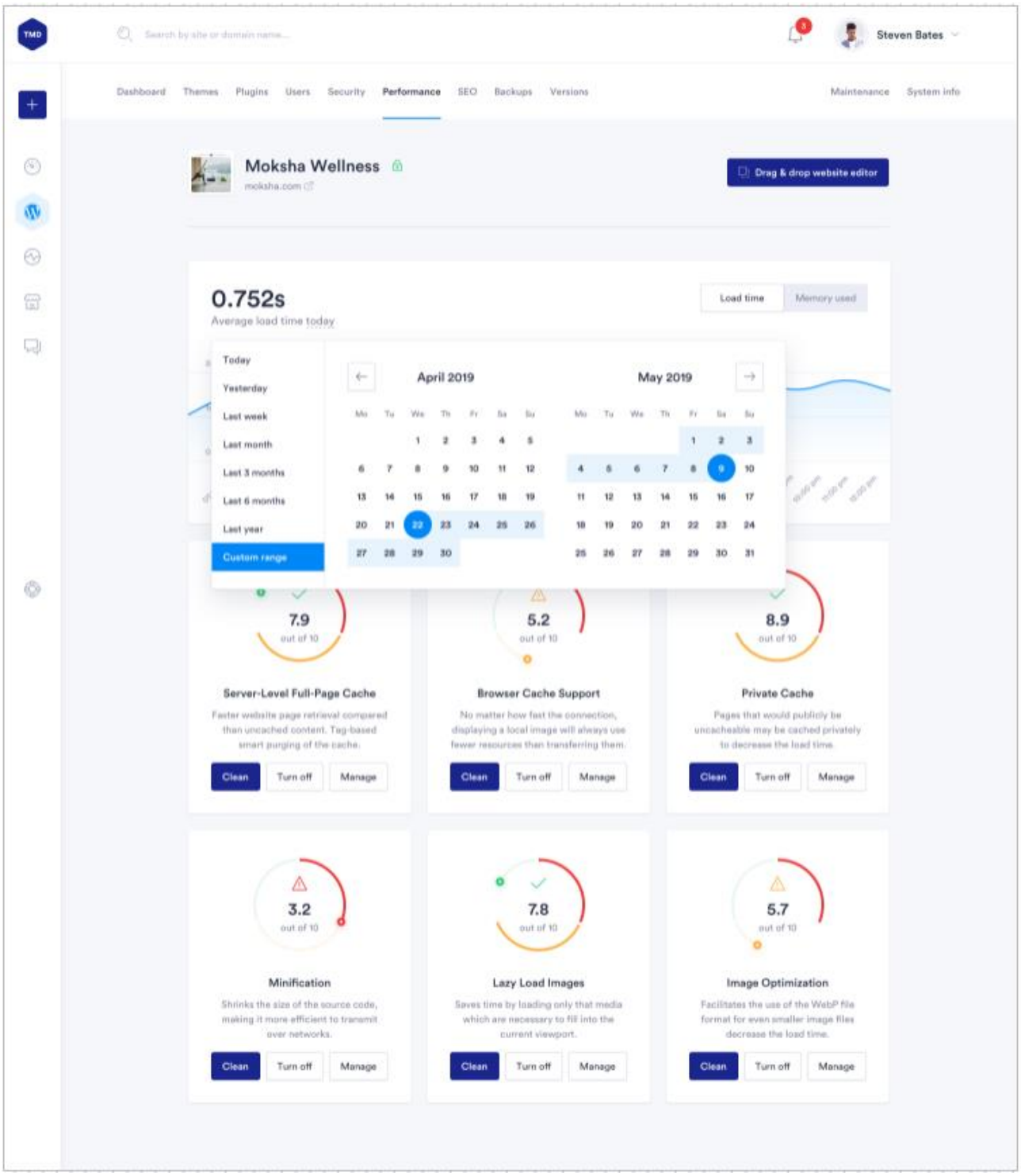

Рисунок 2.14 – Сторінка продуктивності

Вкладка CEO – при переході відбувається запит на сканування, після отримання данних, користувач отримує зауваження, помилки по сайту, які в свою чергу розбиті по секціям (рисунок 2.15).

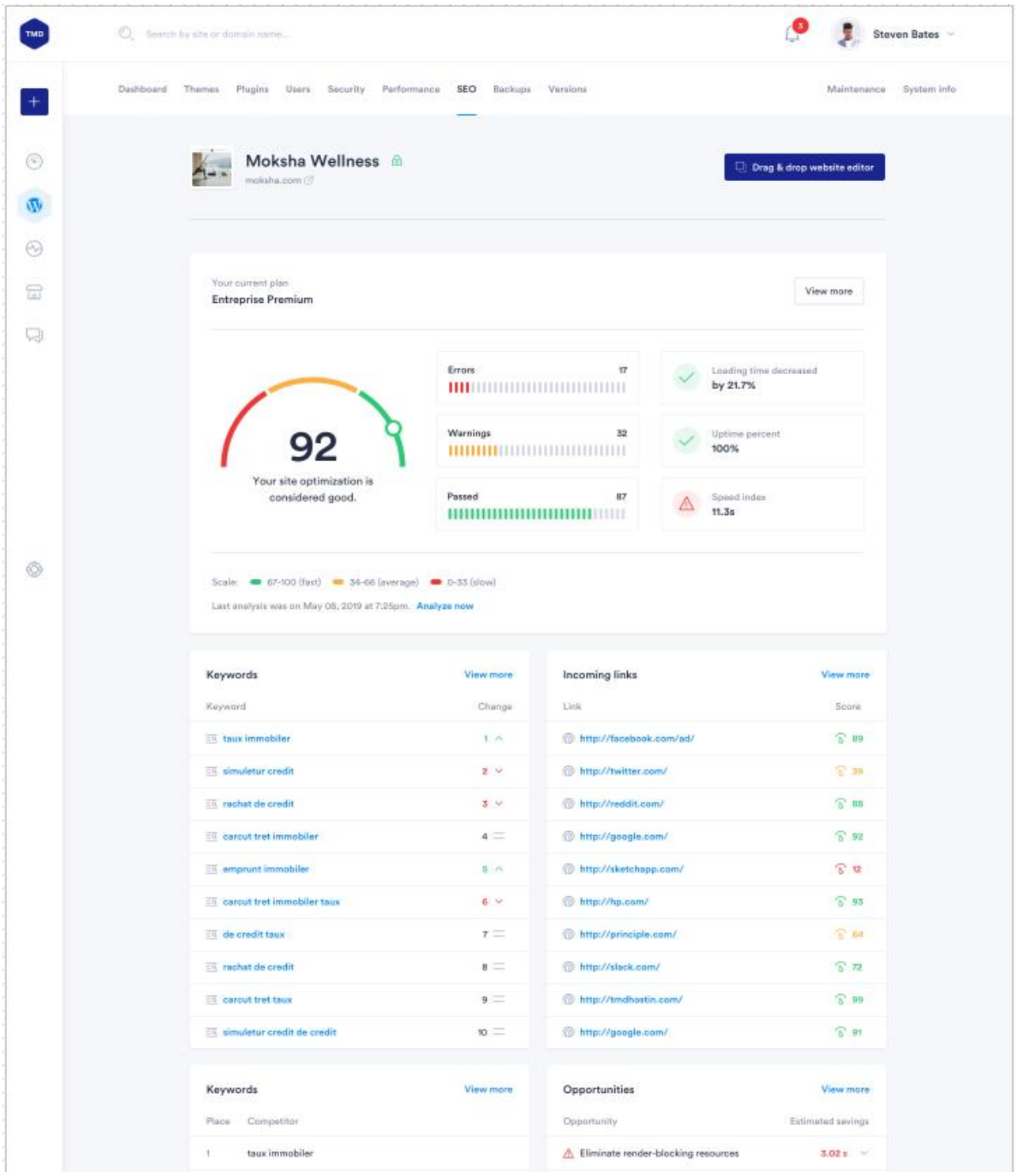

Рисунок 2.14 – Сторінка CEO

Вкладка «Резервні копії»– на ній відображаються списком, усі резервні копії, та користувач, який їх зробив з можливістю застосування чи видалення копії (рисунок 2.15).

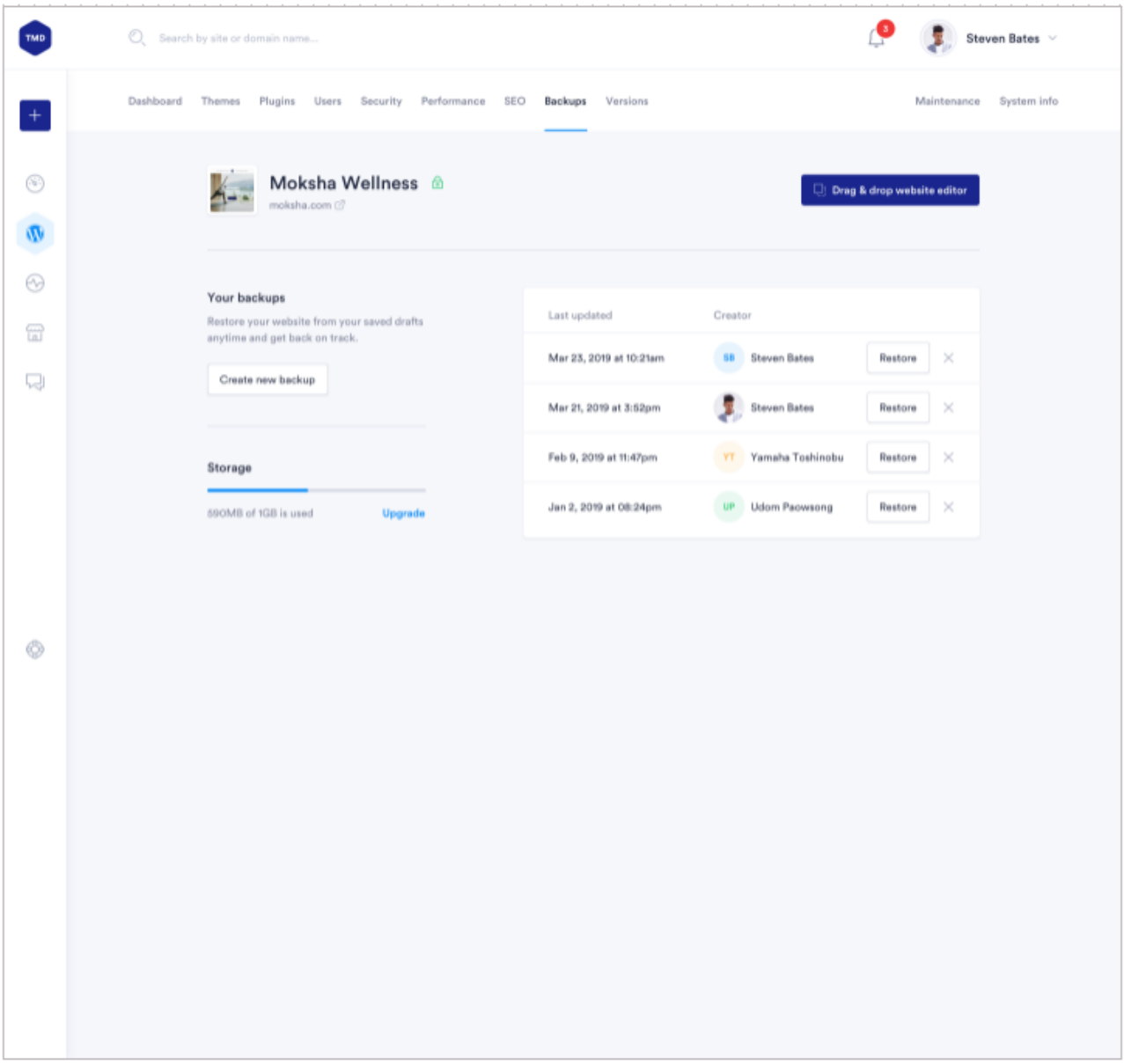

Рисунок 2.15 – Сторінка резервних копій

Вкладка системна інформація –ця вкладка виконана також у плитковому ситлі, на кожній з яких відображається інформація, яка розбита на змістовні розділи. З цієї вкладки можна дізнатись таку важливу інформацію як кількість

зайнятого місця, версію бази данних, кількість встановлених плагінів на цей момент та створених адміністраторів (рисунок 2.16).

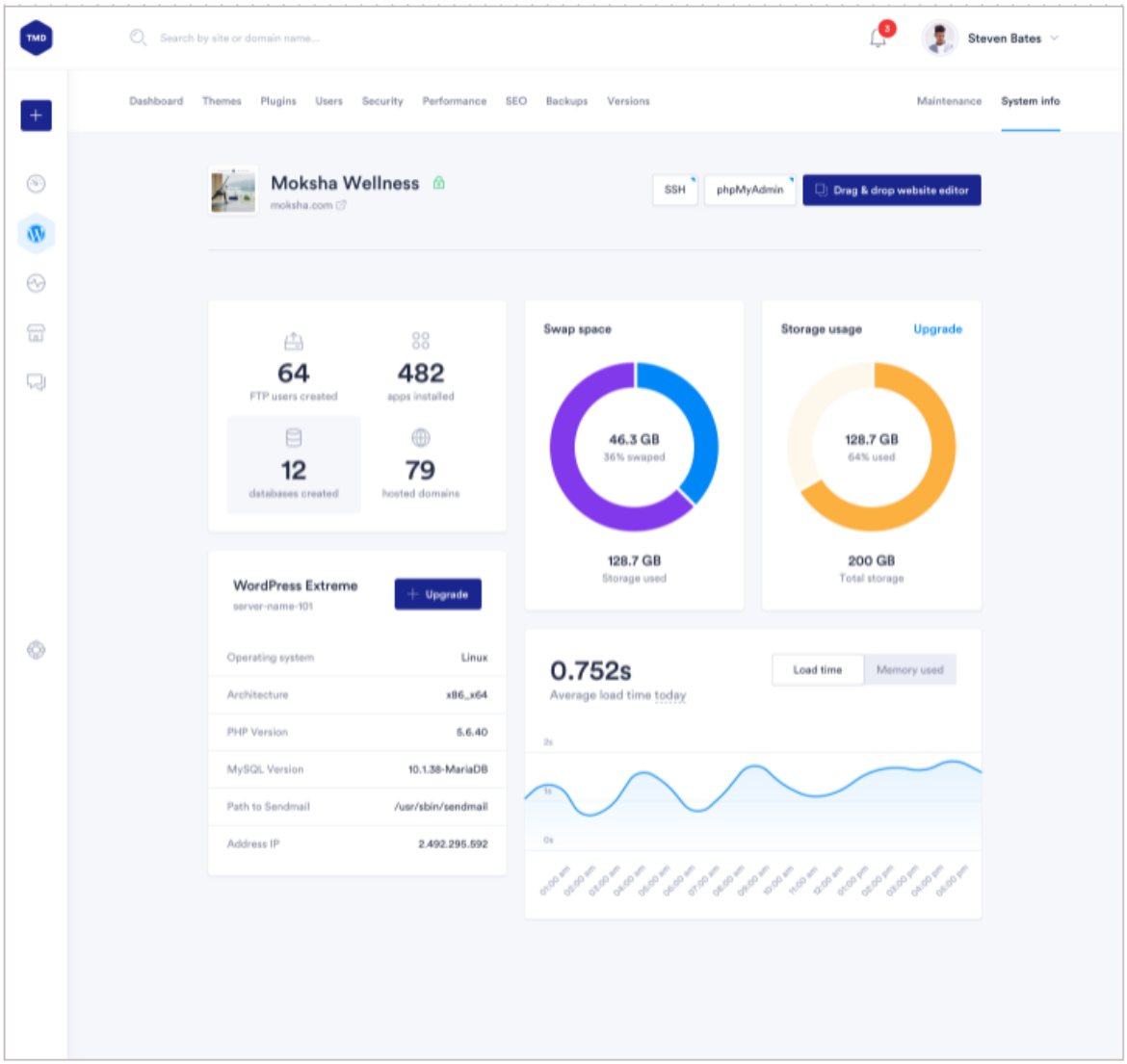

Рисунок 2.16 – Сторінка системної інформації

Отже, для клієнтської частини веб-сервісу розроблено низку веб-сторінок користувача, що забезпечують виконання усіх необхідних функцій розроблюваної системи. Інтерфейс сторінок користуваа складається з представлень інформації, які в свою чергу умовно формують сторінки продукту.

2.3 Розробка методу та моделі роботи веб-сервісу

В основі сучасної розробки веб-інтерфейсів лежить компонентний підхід.

Загальна модель веб-інтерфейсу представляє собою взаємозв'язану систему компонентів, що будуть представлені в системі управління. Компонент – це частина програми, яка може бути використана багато разів. Окремі компоненти формують представлення результатів. Для можливості зберігання даних та комунікацій з сервером використовується store, сукупність цих астракцій реалізує flux-архітектура.

Flux-архітектура — архітектурний підхід або набір шаблонів програмування для побудови користувацького інтерфейсу веб-додатків, що співпрацює з реактивним програмуванням та побудована на однонаправлених потоках даних. Відповідно до запропонованого підходу розробників Flux є архітектурним рішенням.

Основною відмінною рисою Flux є одностороння спрямованість передачі даних між компонентами Flux-архітектури. Архітектура накладає обмеження на потік даних, зокрема, виключаючи можливість поновлення стану компонентів відокремленно. Такий підхід робить потік даних передбачуваним і дозволяє легше простежити причини можливих помилок в програмному забезпеченні.

У мінімальному варіанті Flux-архітектура може містити три шари, які взаємодіють один по одному:

- Actions (дії)
- Store (сховище)
- Views (представлення)

Дії (від анг. actions) — вираз подій. Диспетчери передають дії нижчого рівня компонентів (сховищ) по одному. Нова дія не передається поки попередня повністю не оброблено компонентами. Дії через роботу джерела дії, наприклад, користувача, надходять асинхронно, але їх диспетчеризація є синхронним

процесом. Крім імені, дії можуть мати корисне навантаження, що містить пов'язані з дією дані.

Диспетчер (англ. Dispatcher) – призначений для передачі дій сховищ. У спрощеному варіанті диспетчер може бути єдиним на весь додаток. У диспетчері сховища реєструють свої функції зворотного виклику (callback) і залежності між сховищами.

Сховище (англ. Store) є місцем, де зосереджено стан (англ. State) додатків. Інші компоненти, згідно Flux, не мають значного (з точки зору архітектури) стану. Зміна стану сховища відбувається строго на основі даних дії і старого стану сховища.

Подання (англ. View) – компонент, звичайно відповідає за видачу інформації користувачеві. У Flux-архітектурі, яка може технічно не торкатися внутрішнього облаштування уявлень взагалі, це – кінцева точка потоків даних. Для інформаційної архітектури важливо тільки те, що дані потрапляють в систему (тобто, назад в сховища) тільки через дії.

З огляду на архітектуру, сервіс можна розподілити на такі модулі як:

- Модуль панель приладів.
- Модуль список сайтів.
- Модуль сторінка сайту панель приладів.
- Модуль сторінка сайту теми.
- Модуль сторінка сайту плагіни.
- Модуль сторінка сайту користувачі.
- Модуль сторінка сайту безпека.
- Модуль сторінка сайту продуктвність.
- Модуль сторінка сайту оптимізація.
- Модуль сторінка сайту оптимізація.
- Модуль сторінка сайту сео.
- Модуль сторінка сайту резервні копії.

- Модуль сторінка сайту системна інформація
- Модуль сторінка створення веб-сайту.
- Модуль аналітика.
- Модуль створення контенту.

Кожен з модулів може бути представленийяк сторінка сайту. Така функціональна сторінка є моделлю дії та представлення інформації.

Для відображення роботи між сервером та компонентами розроблювальної програми, побудовано модель взаємодії (рисунок 2.17).

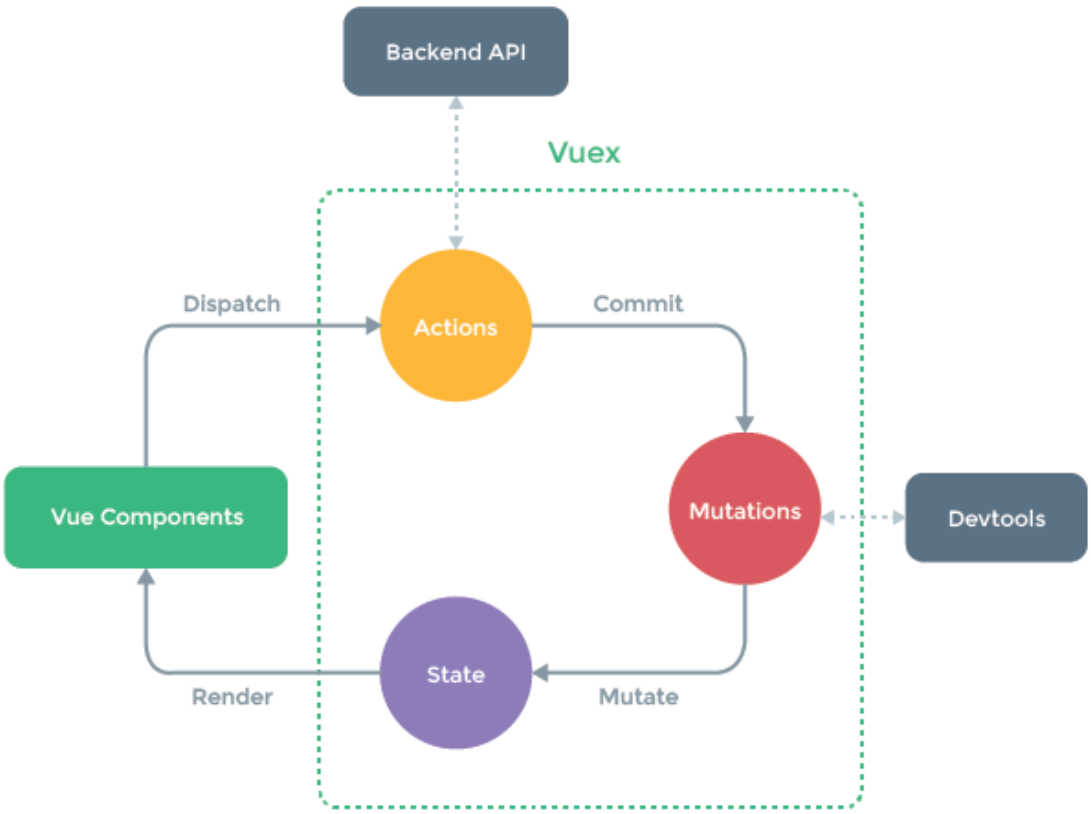

Рисунок 2.17 – Модель взаємодії компонентів програмного продукту

Розробка методу веб-додатку, можна поділити на два етапи, пеший – це можливість користувача аторизуватись в веб додатку (рис 2.18).

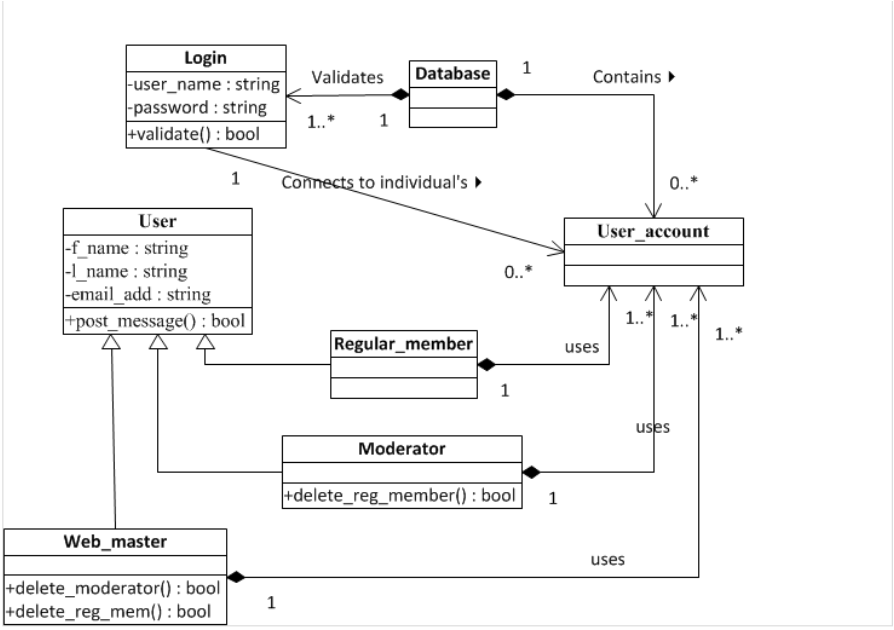

Рисунок 2.17 – Модель авторизації користувача

Модельна Архітектура реалізує вимоги до функціональності та продуктивності системи, а також такі вимоги, як надійність, масштабованість, портативність та доступність (рис 2.18).

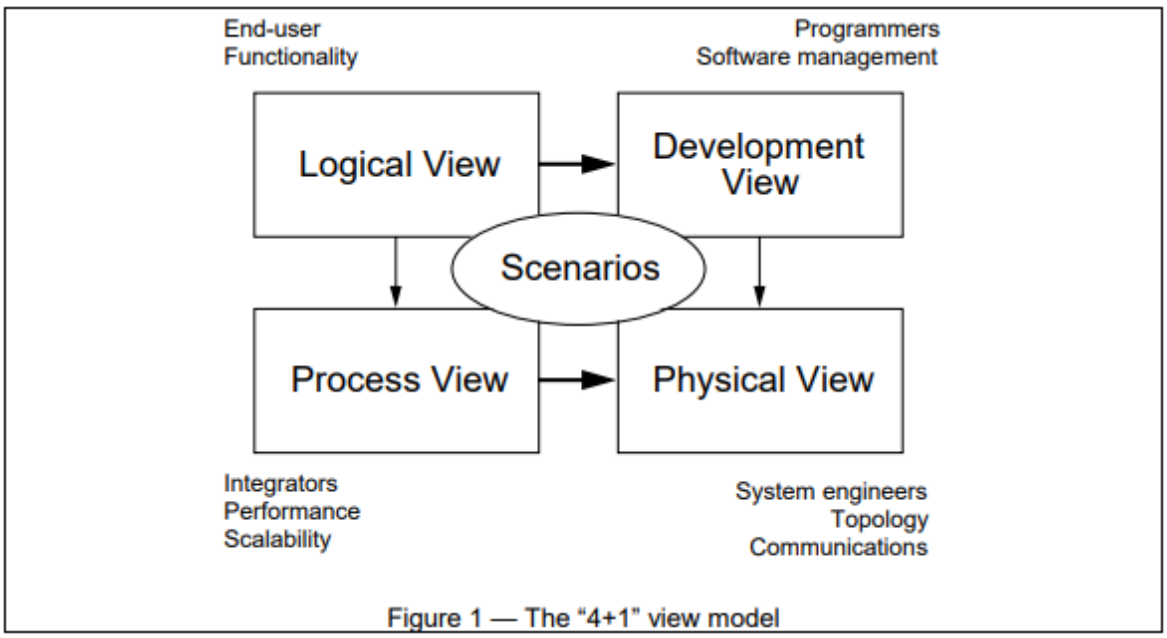

Рисунок 2.18 – Базова модель архітектури

2.4 Розробка основних алгоритмів роботи програми

Розробка системи полягає в тому, що необхідно розробити алгоритми для всіх її модулів, частин і функцій, розроблюваний сервіс виконує широкий спектр задач, проте основна його задача створити веб-сайт на базі Wordpress, тому побудуємо вибору ім'я сайта (рисунок 2.18).

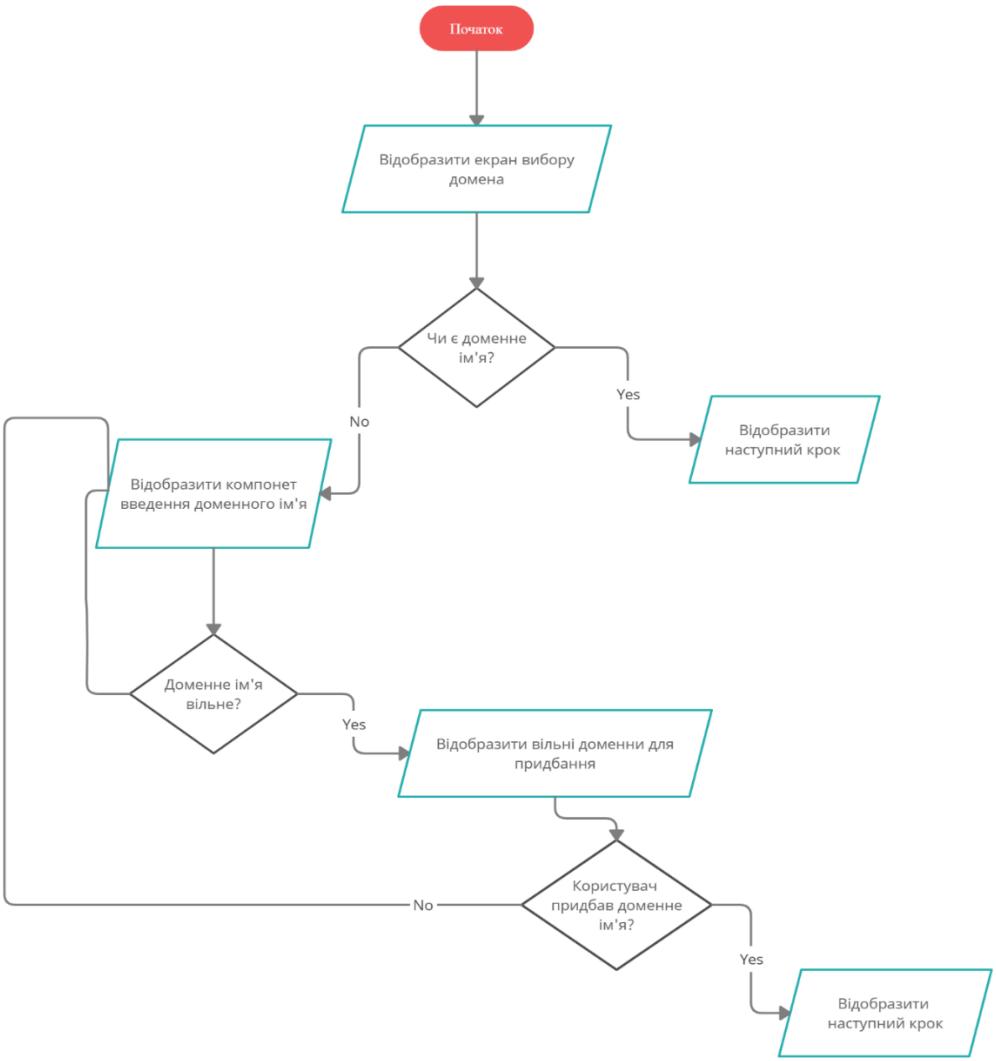

Рисунок 2.18 – Алгоритм Створення веб-сайту (етап 1)

Алгоритм Створення веб-сайту на базі Wordpress (етап 1) складається з таких кроків:

Крок 1. Початок.

Крок 2. В даному кроці відбувається рендерінг екрана на якому користувач бачиь три варіанти, створити новий, обрати існуючий, чи тимчасовий домен

Крок 3. На данному кроці, якшо користувач має доменне ім'я сайту, він одразу може перейти до наступного кроку.

Крок 4. Якщо користувач не має доменного ім'я йому буде запропоновано, поле ввода в яке він введе домен який побажає отримати.

Крок 5. Відбувається асинхронний запит який поверене можливі варіанти які, користувач може придбати.

Крок 6. Користувач купує доменне ім'я та переходить на інший крок, або може повернутись на до вводу нового домену якщо, данний можливі варіанти його не задовільняють.

Крок 7. Кінець взаємодії з данним єтапом та перехід на інший.

Алгоритм Створення веб-сайту на базі Wordpress (етап 2) складається з таких кроків як (рисунок 2.17):

Крок 1. Початок.

Крок 2. Відбувається рендирінг вікно з можливими варіантами створення сайту, такими як: конструктор, інтернет магазин на базі Woocommerce та звичайний Wodpress.

Крок 3. Після вибору методу створення відбувається запит на сервер, який поверне список оформлень

Крок 4. Відбувається рендерінг офрмлень які доступні для сайту.

Крок 5. Користувач обирає офрмлення для свого сайту серед доступних

Крок 6. Відбувається отримання данних зі сховища, а саме отримання стану який зберігає інформацію про обране оформлення, доменне ім'я.

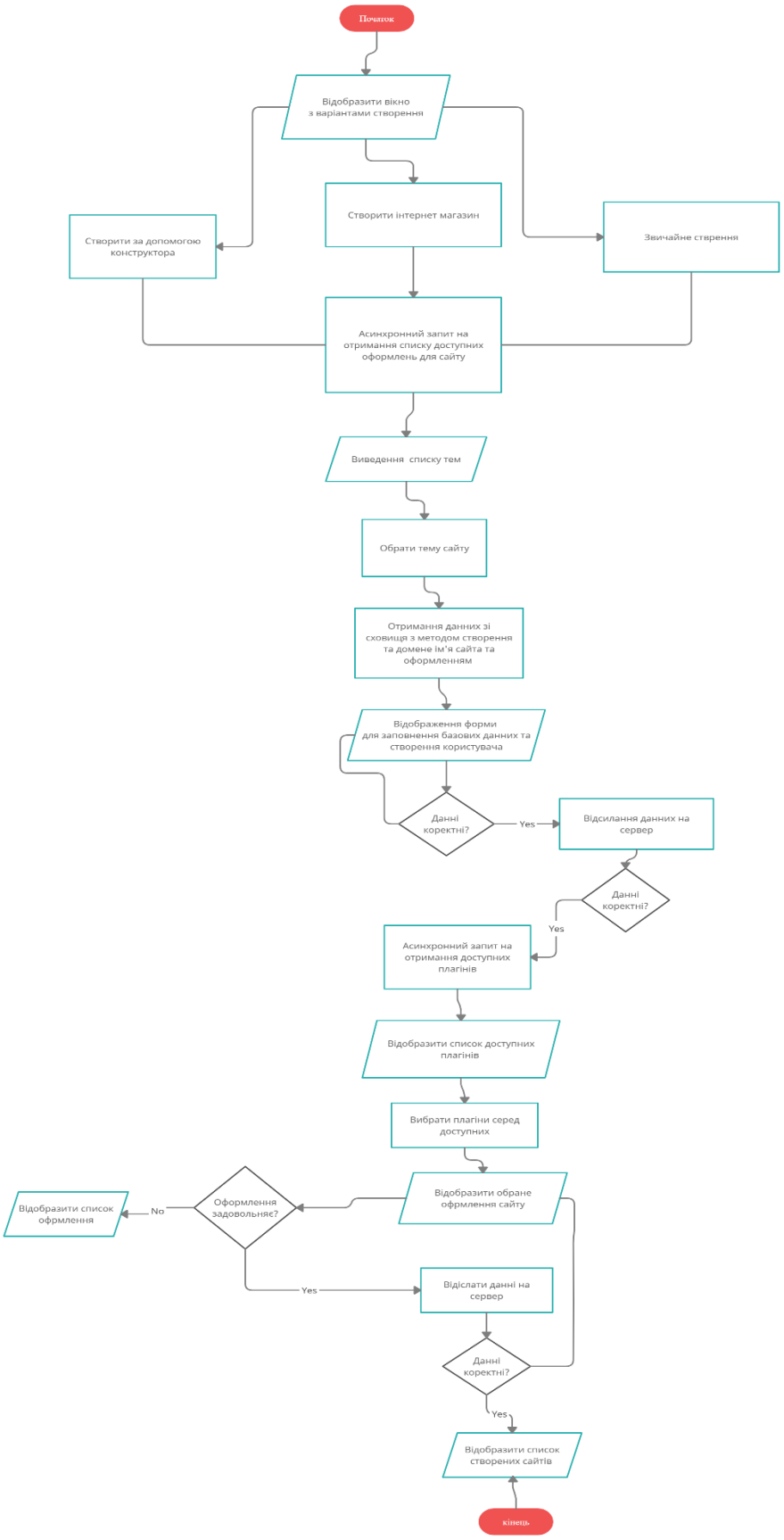

Рисунок 2.18 – Алгоритм Створення веб-сайту (етап 2)

Крок 7. Відобаражено представлення зліва якого розташований сайтбар, який в свою чергру має підпункти такі як: дизайн, плігани, оформлення, та публікація, з права форма, яку користувач заповнює данними.

Крок 8. Відбувається перевірка на клієнті.

Крок 9. Якщо данні коректні відбувається запит на сервервер

Крок10. Відбувається запит на отримання доступних плагінів.

Крок 11. Відбувається рендерінг доступних плагінів

Крок 12. Після вибору плагінів, відмалюється обране користувачем на попередніх кроках.

Крок 13. Якщо користувача влаштовує раніше обрана тема, відбувається перехід до завершального кроку.

Крок 14. Відсилання усіх рані зібраних данних на сервер

Крок 15. Перевірка на коректність данних

Крок 16. Відображення списку сайтій, серед яких користувач зможе знайти створений та адмініструвати його.

2.5 Висновки

У другому розділі було проаналізовано розробку клієнтську частину веб-сервісу, який удосконалює створення сайтів на базі системи управління системи керування вмістом Wordpress, виявлено основні функції такої системи, розроблено інтерфейс користувача вихідного програмного продукту, що складається з десяти основних сторінок, елементу навігації та сторінок адміністрування. Також розроблено ключові алгоритми роботи системи, такі як алгоритм надсилання розв’язку, алгоритм автоматичного оцінювання розв’язку, алгоритм формування таблиці з результатом змагань, алгоритм імпортування задачі з архівного файлу. Разом з цим побудовано діаграми компонентів та послідовності для наочного відображення взаємодії між основними модулями додатку, серверною частиною та іншими використаними сторонніми сервісами.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів реалізації програмного засобу.

Перед розробкою будь-якої програми необхідно обрати засоби його реалізації. Правильний вибір таких засобів дуже важливий, оскільки від нього залежить складність розробки, ефективність і швидкість роботи програми, безпека зберігання даних.

Для реалізації серверного застосунку потрібно обрати середовище його роботи (операційну систему), мову програмування з необхідними бібліотеками й фреймворками, а також технології баз даних для зберігання даних.

Windows, Mac OS і Linux – популярні сучасні операційні системи, що отримують оновлення, підтримку та регулярний випуск нових версій [10].

ОС Windows випускається компанією Microsoft та є найбільш попопулярною системою у всьому світі. Так, за даними сайту NetApplications, на березень 2020 року Windows була встановлена на 89.21% комп’ютерів, Mac OS – 8.94%, а Linux – 1.36% [11]. Також існує спеціальна версія цієї операційної системи, спроектована для роботи в ролі сервера.

Mac OS – серія пропрієтарних графічних операційних систем корпорацій Apple Inc. Перший випуск відбувся у 2001 році. Є спадкоємцею Mac OS 9 - так званого залишкового релізу «класичної» Mac OS - основної операційної системи корпорації Apple з 1984 року. OS X входить у сімейство операційних систем Apple OS X, а також до складу ОС для мобільних пристроїв - iOS. У macOS використовується ядро Darwin, закріплене на мікроядрі Mach, що містить код, написаний самою компанією Apple та код, отриманий з ОС NeXTSTEP і FreeBSD. Apple macOS випускається для комп'ютерів Macintosh (Макінтош) на базі процесорів PowerPC та Intel (починаючи з версії 10.6,) macOS підтримує

лише комп'ютери Mac на базі процесора Intel. Mac OS - друга за популярністю у світі операційна система. Її ринкова частина у червні 2010 року - 6,8% .

Linux – загальна назва UNIX-подібних операційних систем на основі однойменного ядра. На відміну від інших популярних операційних систем, вихідні коди доступні всім для використання, зміни та поширення абсолютно вільно й безкоштовно. ОС UNIX вважається досить ефективною у роботі з мережевою платформою для будь-якого типу серверу: локального, корпоративного або глобального [12]. Вона має усі необхідні інструменти для розміщення веб-сайтів та перевірену часом безпеку.

У таблиці 3.1 зведено результати порівняльного аналізу операційних систем з огляду на розробку веб-додатків.

Таблиця 3.1 – Порівняння операційних систем

Характеристика	Операційна система		
	Windows	Mac OS	Linux
Безкоштовна	-	+/-	+
Відкритий вихідний код	-	-	+
Наявність серверної версії	+	-	+
Наявність технічної підтримки	+	+	-
Низькі технічні вимоги	-	-	+

Для створення веб-ресурсів зазвичай застосовують реляційну базу даних. Реляційна база даних – тіло зв’язаної інформації, що зберігається в двовимірних таблицях, нагадуючи адресну чи телефонну книгу [13]. Серед таких баз даних найбільше поширення отримали PostgreSQL та MySQL.

PostgreSQL – Об'єктно-реляційна система управління базами даних (СКБД). Є альтернативний варіант як комерційний СКБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СКБД з відкритим кодом (MySQL, Firebird, SQLite). Порівняно з іншими проєктами з відкритим кодом, такими як Apache, FreeBSD або MySQL, PostgreSQL не контролюється якою-небудь компанією, її розробка може бути використана співробітниками багатьох людей та компаній, які хочуть використовувати цей СКБД та впроваджувати в нього найновіші досягнення.

Сервер PostgreSQL написав на мові C. Зазвичай розповсюджується у вікні набору текстових файлів із сирцевим кодом. Для встановлення потрібно відкомпілювати файли на кожному комп’ютері та скопіювати в якийсь каталог. Весь процес детально описаний у документаці.

PostgreSQL широко розповсюджена система управління базами даних з відкритим сирцевим кодом. Прототип був випущений в Каліфорнійському університеті Берклі в 1987 році під назвою POSTGRES, після чого активно розвивався та доповнювався. У червні 1990 року з'явилася друга версія із переробленою системою правил маніпулювання та роботи з таблицями, в 1991 році - третя версія, із доданою підтримкою одночасної роботи кількох менеджерів із збереження, покращеним механізмом записів та допоміжною системою внутрішніх правил. У цей час POSTGRES використовується для реалізації великих систем, таких як: аналіз системи фінансових даних, пакет моніторингу функціональності потоків, база даних відстеження астероїдів, система медичної інформації, кілька географічних систем.

MySQL – найпопулярніша база даних у світі, має безкоштовну і платну версії. Найбільше відома за високу швидкодію обробки великої кількості даних

та простоту використання. Однак, для досягнення своєї швидкості, ця база даних має певні розходження зі стандартом SQL, ігнорує деякі перевірки цілісності.

Unix, Linux, Mac OS чи Windows із веб-сервером та встановленим PHP 7. З метою розробки серверної частини веб-ресурсів можна використовувати будь-яку мову програмування, однак є деякі з них найкраще підходять для цієї задачі. Одними з найпоширеніших серверних мов програмування є JavaScript (Node.js), Php, C#.

З метою створення клієнтської частини, обрано мову програмування JavaScript, так як це єдина мова програмування яку існує у браузері.

JavaScript – інтерпретована прототипна мова програмування з динамічною типізацією, що спочатку задумувалась для додання інтерактивності на сторінки у веб-браузері. Однак, з часом її почали використовувати і для написання серверних застосунків. Використання однієї мови для кожного шару додатку – це чудовий спосіб зменшити складність програмного забезпечення, дозволяє уникати невідповідностей та полегшує повторне використання коду [17]. Проте ця мова має деякі проблеми, допущені при початковому проектуванні, які зберігаються і до сьогодні. Наприклад, в мові немає підтримки цілих чисел, використовується слабка типізація та агресивне приведення типів, тощо.

Для створення SPA, знадобіться маршрутизаця, сховище для данних, та налагодження звязків між ними, тому для реалізації данного інтерфейсу на помві програмування JavaScript нам знадобиться фреймфорк або біліотека.

Фреймворк – програмна платформа, яка визначає структуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних компонентів великого програмного проекту.

Vue.js (читається як "в'ю", з англ. view) — JavaScript-фреймворк що використовує шаблон MVVM для створення інтерфейсів користувача на основі моделей даних, через реактивне зв'язування даних.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє декларативно зв'язувати рендеринг DOM з основними екземплярами даних в

Vue. Всі Vue шаблони валідні HTML, і можуть бути розпарсені браузерами та HTML парсерами. Всередині Vue компілює шаблони в рендерингові функції віртуального DOM. В поєднанні з реактивною системою, Vue здатний розумно обчислити кількість компонентів для ре-рендингу та застосувати мінімальну кількість маніпуляцій з DOM, коли стан застосунку зміниться.

В Vue ви можете використовувати синтаксис шаблонів або напряму писати рендерингові функції використовуючи JSX. Для того, щоб це зробити просто замініть шаблон на рендерингову функцію. Рендерингова функція відкриває можливості для потужних патернів базованих на компонентах - для прикладу, нова транзитна система тепер повністю базована на компонентах, що використовує рендерингові функції всередині.

Angular (зазвичай так називають фреймворк Angular 2 або Angular 2+, тобто вищі версії) — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular — це AngularJS, який переосмислили та який був повністю переписаний тією ж командою розробників.

React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб

піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Кожен фреймвок повинет реалізувати flux-архітектуру, та має здатність ефективно перемальовувати компоненти програми відповідно до зміни стану веб-додатку, крім цього може мати набір готових рішень для реалізації. Результати порівняльного аналізу мов програмування для реалізації веб-додатку наведено у таблиці 3.2.

Для розробки серверної частини сайту було обрано мову програмвання PHP, а саме її фреймворк Symfony.

Symfony — відкритий каркас вебзастосунків написаний на PHP і набір багаторазових компонентів/бібліотек для найзагальніших веб-задачі. Випускається під ліцензією MIT. Symfony є вільним програмним забезпеченням. Веб-сайт першої версії symfony-project.com був запущений 18 жовтня 2005 року.

Таблиця 3.2 – Порівняння мов програмування

Характеристика	Мова програмування		
	React	Vue.js	Angular
Фреймворк	-	+	+
TypeScript при розгортці проекта	-	-	+
Підтримка CLI	-	+	+
Реактивна компонентна структура	+	+	+
Підтримка Virtual DOM	+	+	-

Данний фреймворк підходить підзадачі удосконалення розробки веб-сайтів на базі системи керування вмістом Wordpress, оскільки розширює функціонал системи, яка уже написанна на данному фреймворку.

3.2 Вибір середовища розробки

Для ефективної розробки програмного забезпечення застосовуються інтегровані середовища розробки (IDE – Integrated Development Environment). Інтегроване середовище розробки – це комплексне програмне рішення для розробки програмного забезпечення, що складається з редактора вихідного коду, інструментів для автоматизації компіляції та відлагодження програм. Більшість сучасних середовищ розробки мають вбудовану функцію автодоповнення коду.

З метою вибору середовища розробки для порівняння було обрано середовища Visual Studio 2017, Webstorm.

Microsoft Visual Studio – серія продуктів корпорації Microsoft, що підтримують розробку для .NET, а також розробку на C++, Python, Node.js, JavaScript/TypeScript. Microsoft Visual Studio надає засоби для розробки консольних додатків, програм з графічним інтерфейсом, зокрема з підтримкою технологій Windows Forms та WPF, а також веб-сайти, веб-додатки тощо.

Webstorm – інтегроване середовище розробки для JavaScript, HTML та CSS від компанії JetBrains, розроблена на основі платформи IntelliJ IDEA. WebStorm є спеціалізованою версією PhpStorm, пропонуючи підмножину з його можливостей. WebStorm постачається з перед-установленим плагінами JavaScript (такими як для Node.js), котрі доступні для PhpStorm безкоштовно.

WebStorm підтримує мови JavaScript, CoffeeScript, TypeScript та Dart. WebStorm забезпечує автодоповнення, аналіз коду на льоту, навігацію по коду, рефакторинг, зневадження та інтеграцію з системами управління версіями. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (у тому числі, рефакторинг коду JavaScript, що міститься в різних

файлах і теках проекту, а також вкладеного в HTML). Підтримується множинна вкладеність (коли в документ на HTML вкладений скрипт на Javascript, в який вкладено інший код HTML, всередині якого вкладений Javascript) — в таких конструкціях підтримується коректний рефакторинг.

Таблиця 3.3 – Порівняння середовищ розробки

Критерій	Visual Studio 2017	WebStorm
Вбудована підтримка модульного тестування	+	+
Вбудована підтримка контролю версій	+	+
Кастомізація гарячих клавіш	-	+
Автодоповнення коду	+	+
Можливість додання розширень	+	+
Наявність анотацій для контролю версій	-	+

У результаті аналізу було прийнято рішення про використання Webstorm, так як задяки розвинутому аналізатору написання коду та розширеними можливостями налаштування гарячих клавіш.

3.3 Розробка модулів ресурсу

Розроблювана програма складається з сімнадцяти модулів. Основними сценаріями взаємодії кінцевих користувачів є створення сайту на базі системи

управління Wordpress. Кожен сценарій реалізується через звя'язов із серверною частиною, оскільки на кожному єтамі є необхідність отримання та запис данних.

Symfony використовує шаблон проектування Model-View-Controller. Розробників Symfony надихнули такі фреймворки, як Ruby on Rails, Django і Spring Framework. Вихідний код програми розділяється на моделі, що відповідають за роботу з базою даних та валідації; представлення, які виконують необхідні перетворення для відображення даних з моделей користувачу; контролери, що виконують обробку запитів від користувача та є свого роду вхідними точками програми. Для реалізації основних модулів веб-ресурсу необхідно створити класи наступних моделей: Worker, Submission, Problem, Group, Membership, User, Website. Також потрібні класи таких контролерів: ApiController, SubmissionsController, ProblemsController, GroupsController, MembershipsController, ArchivesController, StandingsController. Для кожного з контролерів також потрібно розробити файли представлень для відповіді на його запити.

3.3 Висновки

У третьому розділі було проведено аналіз різноманітних технологій та засобів для розробки веб-ресурсів, такі як мови програмування, операційні системи й бази даних. Під час аналізу визначено переваги і недоліки розглянутих засобів, обрано з них ті, що найкраще підходять для реалізації цього дипломного проекту. У результаті було прийнято рішення використовувати Linux в якості операційної системи. Ця ОС має всі потрібні можливості для швидкого впровадження та безпроблемної довготривалої роботи веб-ресурсу, а також має безкоштовну ліцензію і не використовує багато системних ресурсів. В якості мови програмування обрано JavaScript, оскільки ця мова має підтримку Linux і є об’єктно-орієнтованою, а доступний фреймворк Vue.js забезпечує простоту у створенні односторінкових додатків.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Опис методів тестування

Тестування програмного забезпечення – процес аналізу програмного засобу та супутньої документації з метою виявлення дефектів і підвищення якості продукту. Основними характеристиками сучасного тестування є використання гнучких методологій, глибока інтеграція з процесом розробки, широке використання автоматизації, набор різних технологій та інструментів [19]. Техніка тестування включає як процес пошуку помилок чи інших дефектів, так і випробування програмних складових з метою оцінки. Можна виділити кілька методів тестування, що відрізняються предметом дослідження та відомими даними про систему.

Метод статичного тестування передбачає перевірку документації і результатів розробки програми, наприклад технічного завдання, специфікації, вихідного коду програми. Проводиться аналіз дотримання стандартів програмування, відповідності заданим критеріям і вимогам.

Тестування методом «білої скриньки» полягає у дослідженні внутрішньої поведінки програми, коректність побудови окремих модулів чи їх складових, правильність взаємодії модулів між собою. Для такого тестування необхідно знати та розуміти внутрішню структуру програми. Тестування методом «білої скриньки» має деякі недоліки, наприклад повне тестування окремих модулів програми може не гарантувати відповідність усім вимогам, а також складно виявити деякі типи помилок в ізольованому середовищі.

Інший підхід забезпечується методом тестування «чорної скриньки», коли той, хто проводить тестування, не знає нічого про внутрішню структуру програми і все тестування проходить через інтерфейс користувача. В такому випадку досліджується робота кожної функції сервісу разом, в середовищі, що максимально наближено до користувача. Такий метод дозволяє забезпечити

повну перевірку всіх функціональних вимог до програми, визначити помилки, що не можливо знайти іншими методами. Однак, неможливо провести вичерпне тестування всіх можливих сценаріїв роботи, оскільки навіть в невеликих програмах їх кількість може бути дуже великою. Тому, таким методом зазвичай перевіряють критичну функціональність продукту за прогнозованими сценаріями використання.

Для пришвидшення тестування варто автоматизувати його там, де можливо. Автоматизоване тестування – це набір підходів й технологій, які дозволяють виключити людину з виконання деяких задач в процесі тестування. Автоматизація має такі переваги, як набагато швидше проведення тестування, виключення впливу людського фактору на результат, можливість обробки значно більшої кількості інформації [19]. Також такий підхід дозволяє використати концепції керованої тестами (TDD) чи поведінкою (BDD) розробки. Це схожі підходи до розробки програмного засобу, які починаються з написання тестів чи опису поведінки програми, після чого відбувається написання самої програми до того моменту, поки тести не проходитимуть успішно. RSpec – фреймворк для автоматичного тестування на Rails з підтримкою BDD, що допомагає писати ефективні тести для веб-додатків. RSpec пропонує особливі концепти, з використанням яких можна описувати специфікацію та поведінку певної частини коду в близькому до природної мови форматі, забезпечуючи при цьому її автоматичне тестування [20].

4.2 Тестування роботи клієнтської частини веб-ресурсу

Оскільки кожен з методів тестування має свої преваги й недоліки, необхідно використати їх комбінацію для якісного тестування програмного продукту. Для виконання тестування методом «білої скриньки» використаємо можливості фреймворку Enzyme та Jest.

Jest - фреймворк для тестування JavaScript, які фокусується на простоті. Однією з його унікальних особливостей є можливість створення snapshot-тестів, як альтернативний засіб перевірки модулів програми.

Mocha - фреймворк для тестування JavaScript, який фокусується на гнучкості. Завдяки їй можна вибирати різні бібліотеки для виконання загальних функцій, таких як spying (наприклад, Sinon) або assertions (наприклад, Chai). Інша унікальна особливість Mocha в тому, що вона дозволяє виконувати тести як в браузері, так і в Node.js.

Для тестування більшості компонентів Vue потрібно налаштувати їх до DOM (неважливо, віртуальному чи реальному), щоб мати можливість повністю перевірити їх роботоздатність. Ця концепція також фреймворк-агностична. Тому були створені фреймворки для тестування компонентів, які пропонують користувачеві можливість зробити цей надзвичайний спосіб, а також підтримати /Для кожного з модулів та його складових частин, таких як контролери, моделі, сервісні класи і декоратори, необхідно написати

Також виконаємо тестування методом «чорної скриньки», а саме перевіримо роботу працюючого веб-сервісу через його інтерфейс користувача. Для цього потрібно відкрити головну сторінку, заповнити відповідні поля форми та зареєструватись на сайті для подальшої роботи (рисунок 4.2).

При кліку на створення веб-сайту протестовано відображеня екрану підбору доменну, показу оформленнь сайту та форма з покроковим заповненням.

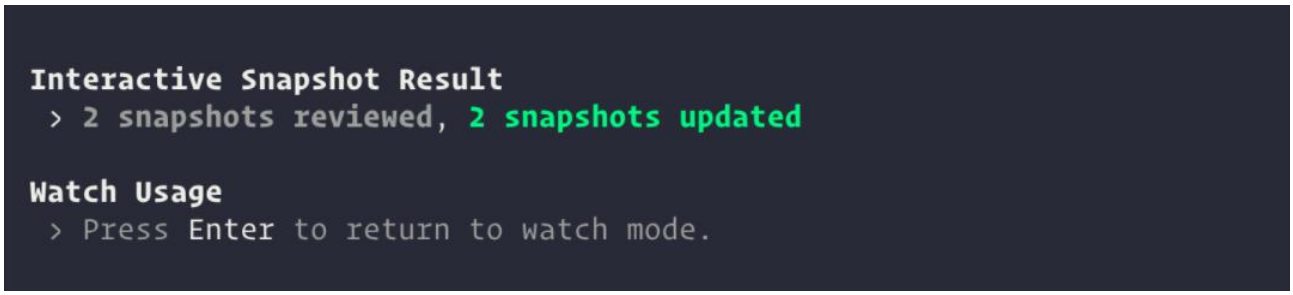

Рисунок 4.1 – Результат тестування Jest

Після створення веб-сайту ми можемо перейти на сторінку управління сайтми та продовжити адміністрування.

Також виконаємо тестування методом «чорної скриньки», а саме перевіримо роботу працюючого веб-ресурсу через його інтерфейс користувача. Для цього потрібно відкрити сторінку (рисунок 4.2).

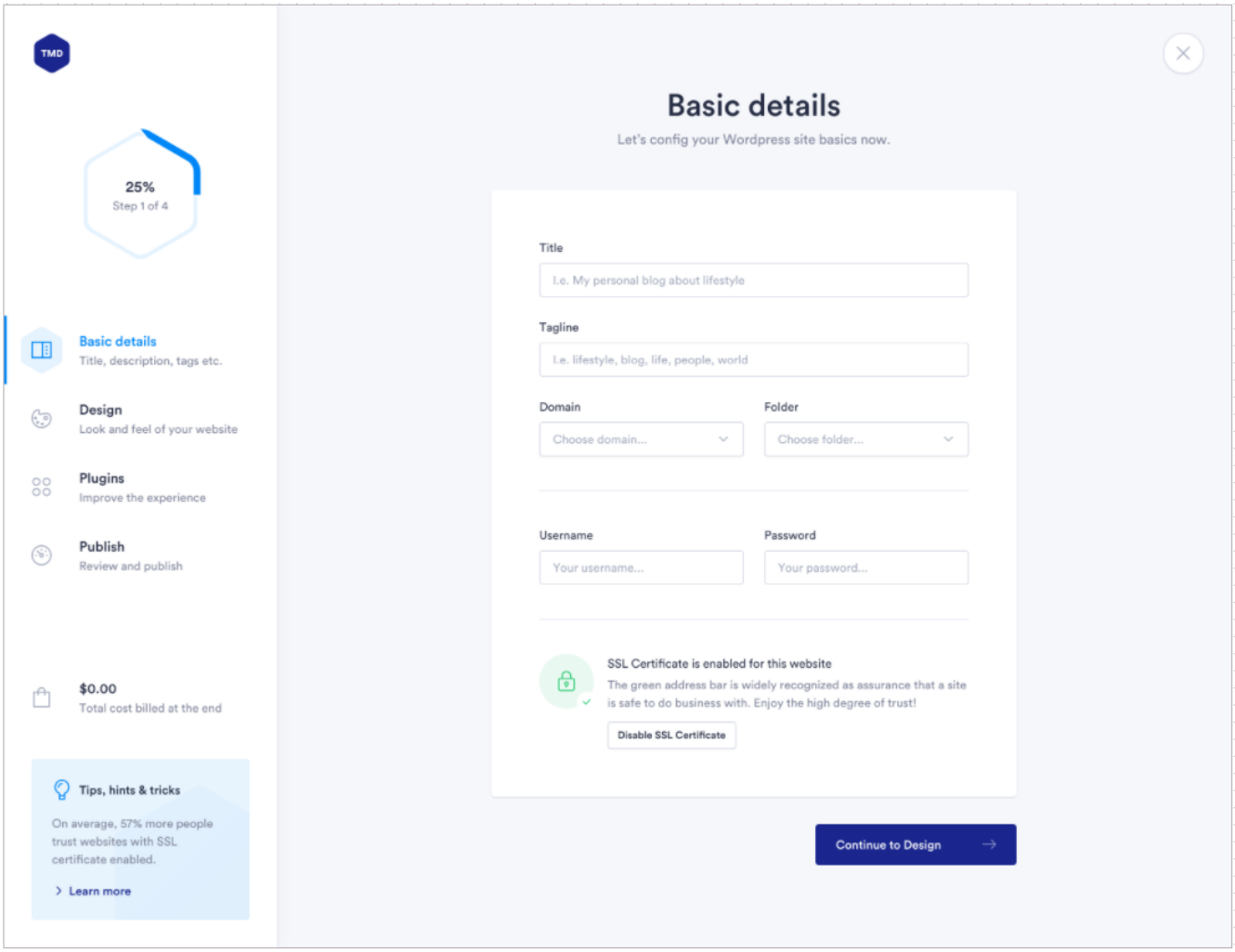

Рисунок 4.2 – Заповнення інформації для створення сайту

Заповнивши правильно усі поля форми а саме: заголовок створенного сайту, популярні теги за якими можливий мопуш по тегам, обрати домен сайту та створити першого користувача для доступу в адмін панель сайту, також є

можливість вимктути або увімкнути сертифікат безпеки, отримаємо перхід на другий крок, на якому користувач має змогу відредагувати обраний дизайн або продовжити з оформленням, яке користувач попередньо обрав (рисунок 4.3).

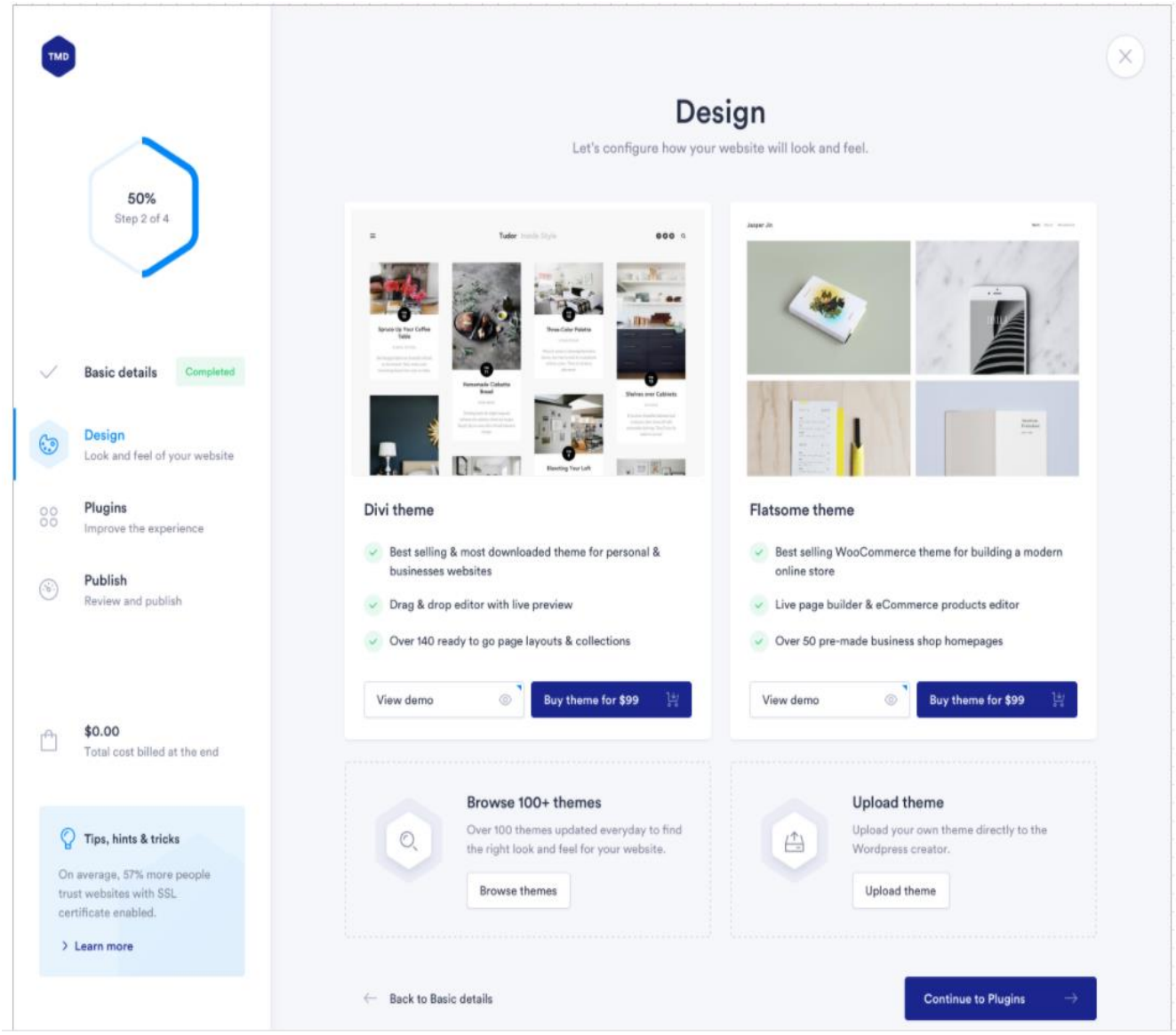

Рисунок 4.3 – Обрання дизайну сайту

Обравши дизайн сайту, користувач має змогу обрати потрібні плагіни для свого сайту а перехід користувача від одної вкладки до іншої відбувається атоматично після заповнення данних та натискання кнопки продовжити з назвою наступного кроку (рисунок 4.4).

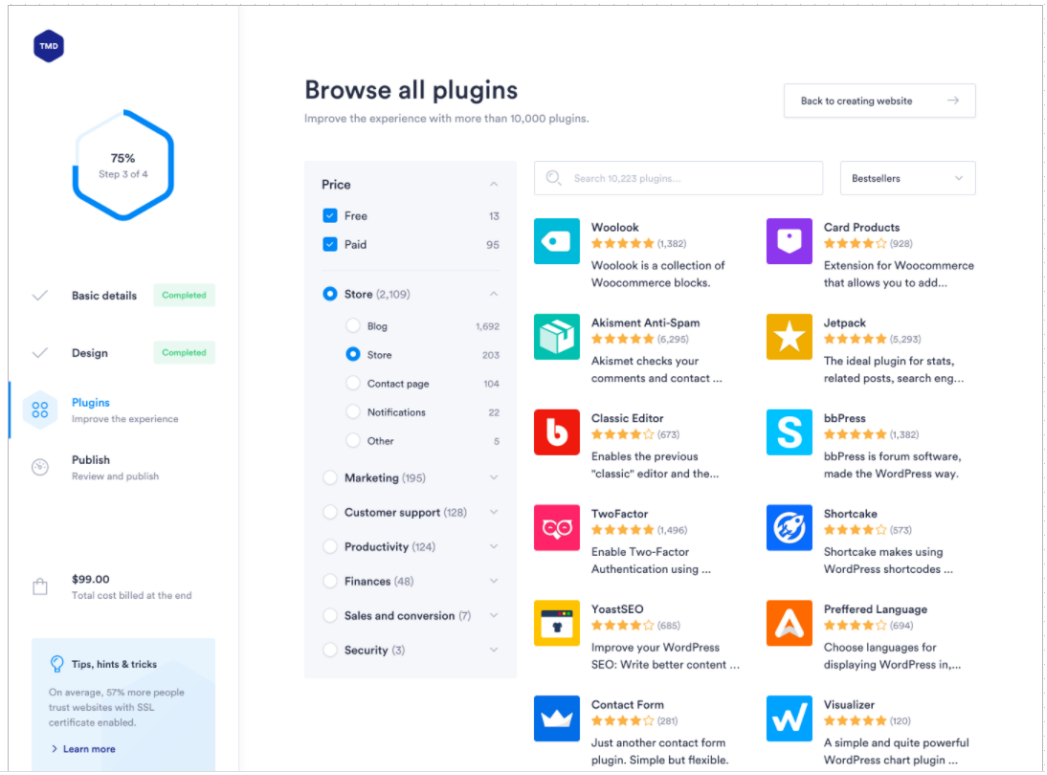

Рисунок 4.4 – Обрання дизайну сайту

Обравши плагіни, користувач переходить на єтап публікації (рисунок 4.5).

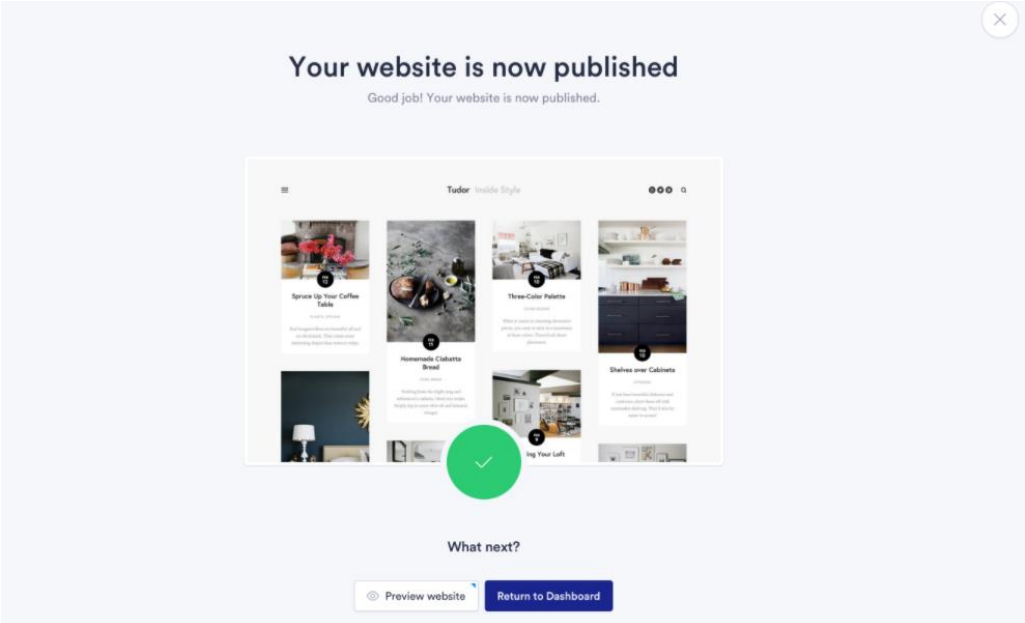

Рисунок 4.5 – Завершення створення сайту

Після відправки данних на сервер, користувач зможе повернутись до сторінки панель приладів чи продовжити роботу зі створенним веб-сайтом.

4.3 Висновки

У четвертому розділі було описано різні методи тестування та інструменти для його проведення в автоматичному режимі, проаналізовано їх переваги й недоліки, сфери використання. Також виконано тестування розробленої клієнтської частини веб-ресурсу методами «чорної скриньки» та «білої скриньки». Така комбінація дозволить провести ефективну перевірку усіх компонентів в автоматичному режимі, а також впевнитися в правильній роботі системи в цілому та її основних компонентів за рахунок більш детального аналізу в наближених до реальних умовах експлуатації.

При проведенні тестування отримана повна відповідність вхідних даних і вихідних результатів. Помилок при роботі програми не виявлено і це підтверджує нормальний режим роботи. У результаті доведено повну працездатність веб-ресурсу та його відповідність поставленому технічному завданню.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами Коваленко Олена Олексіївна (к.т.н., доц. кафедри ПЗ ВНТУ) та Глущенко Лариса Дмитрівна (к.т.н., доц. кафедри ПЗ ВНТУ).Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	Коваленко Олена Олексіївна	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	3
6	4	4
7	4	3
8	3	4
9	4	3
10	4	4
11	3	3
12	4	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 42$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

Веб-додаток на тему «Удосконалення системи управління веб-сайтами, створеними на базі системи керування вмістом Wordpress. Основне використання данного веб-додтку – це сфера наддання полуг хостингу, данний веб-додаток може стати доповненням до панелі управління хостингом, яку зазвичай надає компанія, яка надає послуги у цій сфері своїм користувачам.

Серед аналогів на ринку існує InfiniteWP, CMS Commander.

Переваги даного продукту:

- Приємний UI.
- Можливість отримати статистику відвідувань по кожному сайту та загальну.
- Можливість отримати статистику залишених коментарів по кожному сайту та загальну.
- Можливість встановлення сайту в декілька кліків.
- Можливість інтеграції веб-додатку як доповнення до існуючих панелей управління.

З технічних переваг є найпопулярніша система керування вмістом Wordpress, яка є найпопулярнішою на ринку, та має можливість використання як для сайту новин, блогу так і для інтернет-магазинів. Ще одна перевага це використання сучасного фреймворку Vue.js, за допомогою якого враження від користування веб-додатку близька до стільникової програми.

Данний продукт є актуальним у наш час, так як попит на створення сайту щороку зростає, а данний веб-додаток надає можливість керувати низкою сайтів.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 21 день;

t - число днів роботи розробника, t = 60 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	8500	404,76	5	2023,80
Інженер-програміст	6200	295,23	60	17714,2
Всього:				19738

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0{,}1 \cdot 19738 = 1973{,}8 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп} = (19738 + 1973{,}8) \cdot \frac{22}{100} = 4776{,}59 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=4 міс.).

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп’ютер	17000	25	4	1416,66
Всього:				1416,66

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	150	1	150
Пачка паперу	уп.	100	1	100
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				280,5

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=2,1 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0{,}8$).

$$В_е= 2{,}1 \cdot 0{,}6 \cdot 200 \cdot 0{,}8 = 201{,}6\ (грн.)$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (19738 + 1973{,}8) = 21711{,}8\ (грн).$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 19738+1973,8+4776,59+1416,66+280,5+201,6+21711,8 = 50098,95 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{50098,95}{1} = 50098,95 \text{ (грн.)}$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{50098,95}{0,9} = 55665,5 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 1000 користувачів, а

прибуток, що отримував розробник до впровадження результатів наукової розробки – 400 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2020} = 30 \cdot 1000 + (400 + 30) \cdot 200 = 116000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2021} = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150) = 180500 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2022} = 30 \cdot 1000 + (400 + 30) \cdot (200 + 150 + 100) = 223500 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

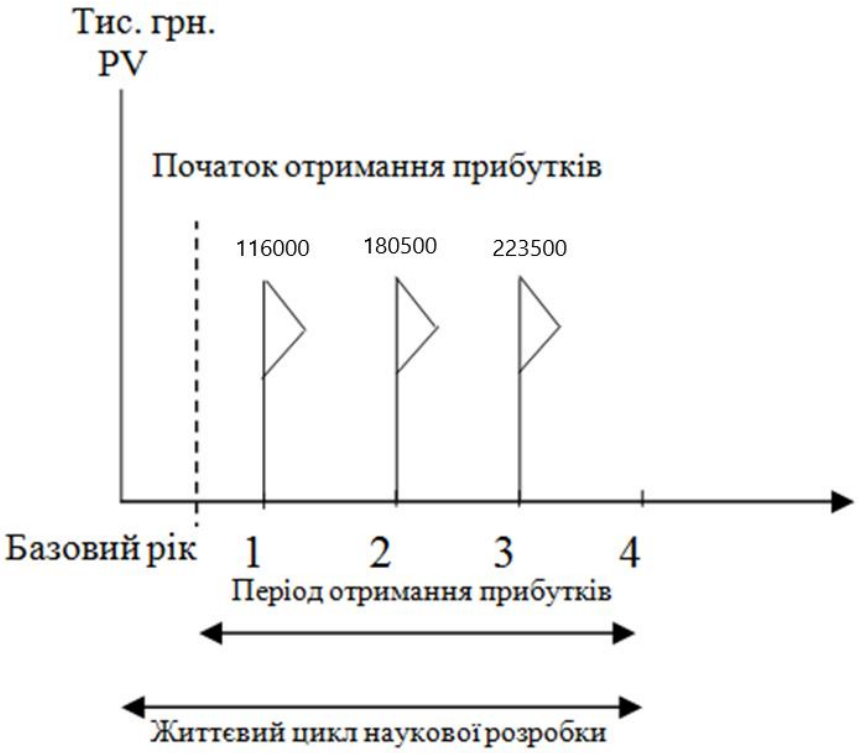

Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{55665,5}{(1+0,1)^0} + \frac{116000}{(1+0,1)^2} + \frac{180500}{(1+0,1)^3} + \frac{223500}{(1+0,1)^4} = 439799,08 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 439799,08 - 55665,5 = 384133,58$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{384133,58}{55665,5}} - 1 = 0,99 \text{ або } 99\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку

інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в$ = 99% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{0{,}99} = 1{,}01 \text{ років}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було удосконаленно систему управління сайтами на базі системи керування вмістом Wordpress.

Виконано аналіз основних аналогів, виявлено їхні переваги та недоліки. На основі отриманих результатів було прийнято рішення про створення власного додатку, який би вирішував проблеми, що присутні в аналогах.

Були вирішенні такі задачі:

– розроблено можливість встановлення сайту на базі системи керування вмістом Wordpress;

– розроблено можливість отримання порад по оптимізації сайту;

– розроблено можливість отримання порад по безпеці сайту;

– удосконалено можливість встановлення плагінів на сайт;

– удосконалено можливість отримання статистики з відвідування сайту;

– удосконалено можливість отримання статистики з сиситомною інформацією;

– спроектовано архітектуру проекту на базазі flux;

– проведено тестування програмного продукту.

В результаті створеного веб-додатку подальшого розвитку дістав метод створення веб-сайтів на базі системи керування вмістом Wordpress, який за рахунок управління низкою створених за різними технологіями сайтів та їх деталізації, дозволяє користувачеві підвищити продуктивність управління сайтами, адмініструвати декілька сайтів без погіршення якості управління та швидкості внесення змін та розвитку отримали модулі збору статистики та засоби оптимізації, комунікації з користувачем шляхом отримання візуальних зображень параметрів для низки сайтів, їх деталізації, формування відповідних

порад з безпеки та оптимізації сайтів, що дозволяє користувачу отримати адаптовані дані для більш якісного управління сайтом.

Запропонований метод, моделі управління на основі низки односторінкових сайтів, аріхітектурін моделі flux та моделі для програмної реалізації стали основою для програмного продукту, що дозволяє більш ефективно керувати групою сайтів, здійснювати зміни вмісту з більшою продуктивністю та якістю, що може бути використано адміністраторами сайтів в організаціях, що використовують динаміні сайти на платформі WordPress.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні системи і технології в статистиці: Навч. посібник / За ред. д-ра екон. наук, проф. В. Ф. Ситника. — К.: КНЕУ, 2003. — 267 с.

2. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ- Плюс, 2007. – 336 с., ил. 3.

3. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов – Москва: «БИНОМ», 2007 – 484 ст.

4. Алгоритми. Побудова і аналіз / Т. Кормен, Ч. Лайзерсон, Р. Ривест, К. Штайн. – Москва: И.Д.Вильямс, 2013. – 1328 с. – (3).

5. TypeScript [Електронний ресурс]. – Режим доступу https://www.typescriptlang.org/.

6. Vue.js [Електронний ресурс]. – Режим доступу https://vuejs.org/.

7. Rect.js [Електронний ресурс]. – Режим доступу https://react.org/.

8. Тестування [електронний ресурс] // Режим доступу: http://www.victoria.lviv.ua/html/wp/l-testing.html – Назва з екрану.

9. Wbstorm - [Електронний ресурс] - Режим доступу https://www.jetbrains.com/ru-ru/webstorm/.

10. Symphony [Електронний ресурс]. – Режим доступу https://symfony.com/.

11. Jest [електронний ресурс] // Режим доступу: https://jestjs.io/ – Назва з екрану.

12. Enzyme [електронний ресурс] // Режим доступу https://enzymejs.github.io/enzyme/ – Назва з екрану.

13. Кормен Т. Алгоритмы: построение и анализ, 2-е издание.: Пер. с англ. / Т. Кормен – М.: Издательский дом «Вильямс», 2005. – 1296 с.

14. Степанченко И.В. Методы тестирования программного обеспечения: учебное пособие / И.В. Степанченко. – Волгоград : ВолгГТУ, 2006. – 74с.

15. Казарин О.В. Теория и практика защиты программ / Олег Казарин – М.: МГУЛ, 2004. – 450 с.- ISBN 5-93517-178-6.

16. Тестування [електронний ресурс] // Режим доступу: http://www.victoria.lviv.ua/html/wp/l-testing.html – Назва з екрану.

17. PHP [електронний ресурс] - // Режим доступу https://uk.wikipedia.org/wiki/PHP/.

18. ООП [електронний ресурс] // Режим доступу - https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

19. Інтерфейс [електронний ресурс] // Режим доступу: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81

20. Лисенка Г.Л. Методичні вказівки до оформлення курсових проектів(робіт) у Вінницькому національному технічному університеті / Уклад. Г. Л. Лисенко, А.Г. Буда, Р.Р. Обертюх, – Вінниця: ВНТУ, 2006. – 60с.

ДОДАТКИ

ДОДАТОК А Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
"17" лютого 2021 року

Технічне завдання
на магістерську кваліфікаційну роботу «Удосконалення системи управління веб-сайтами створеними на базі системи керування вмістом Wordpress»
за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____

____р.

к.т.н., доцент Ковалевський О.Ю.

Виконав:

_____

студент гр. ПІ-19мз Донченко В.В.

В
і
н
н
и

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Удосконалення системи управління веб-сайтами, створеними на базі системи керування вмістом Wordpress».

Веб-додаток – застосовується у процесі створення та використанні сайта, на базі системи керування вмістом Wordpress.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №64 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є удосконалення процесу керування та створення веб-сайтів на базі системи керування вмістом Wordpress, за рахунок використання розробленого веб-додатку, який дозволяє автоматизувати стоврення веб-сайтів та удосконалити систему управління.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Інформаційні системи і технології в статистиці: Навч. посібник / За ред. д-ра екон. наук, проф. В. Ф. Ситника. — К.: КНЕУ, 2003. — 267 с.

2. Дари К., Бринзаре Б., Черчез-Тоза Ф., Бусика М. AJAX и PHP: разработка динамических веб-приложений. – СПб.: Символ- Плюс, 2007. – 336 с., ил. 3.

3. Кузнецов С.Д. Основы баз данных, 2-е издание / С.Д. Кузнецов – Москва: «БИНОМ», 2007 – 484 ст.

4. Алгоритми. Побудова і аналіз / Т. Кормен, Ч. Лайзерсон, Р. Ривест, К. Штайн. – Москва: И.Д.Вильямс, 2013. – 1328 с. – (З).

5. Кормен Т. Алгоритмы: построение и анализ, 2-е издание.: Пер. с англ. / Т. Кормен – М.: Издательский дом «Вильямс», 2005. – 1296 с.

4. Технічні вимоги

Оскільки система складається з 2-х частин (клієнт, сервер), розглянемо вимоги до кожної з них.

1. Клієнт. Оскільки клієнт виконаний в форматі сайту і не має важких обчислень на стороні клієнта, варто звертати увагу лише на вимоги браузера. Розглянемо мінімльні системні характеристики необхідні для роботи з сайтом: Windows 7, Windows 8, Windows 8.1, Windows 10 або пізнішої версії. Процесор Intel Pentium 4 або більш пізніша версія з підтримкою SSE3.
2. Сервер. Мінімальні вимоги: 1 VCPU, 2 GB RAM, 20 GB DISK LOCAL
3. DOCKER. Серверна частина працює на ОС Linux засобами Docker. Якщо маються на увазі апаратні вимоги, то докер сам по собі не має мінімальних апаратних вимог. Це лише утиліта для віртуалізації. Мінімальні системні вимоги залежать від програми, яка буде розвернута в контейнерах. З апаратних вимог лише одна – підтримка віртуалізації. Що стосується системних (програмних) вимог, то для систем на базі ОС Linux потрібно лише: бітність 64, ядро не старіше 3.10.
4. NODE. Якщо маються на увазі апаратні вимоги - все залежить від сторонніх бібліотек які використовуються в самому додатку. Розроблюваний додаток не має важких залежностей.

5. Конструктивні вимоги.

Система повинна бути зручною у використанні та обслуговуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз завдання і вибір методів його вирішення	10.04.19 – 15.04.21	Вик.
2	Аналіз існуючих аналогів та постановка задач дослідження	16.04.19 – 23.04.21	Вик.
3	Розробка методу динамічного завантаження	24.04.19 – 30.04.21	Вик.
4	Розробка структур і алгоритмів програмного продукту	01.05.19 – 04.05.21	Вик.
5	Розробка програмного забезпечення	05.05.19 – 19.05.21	Вик.
6	Тестування розробленого програмного продукту	20.05.19 – 24.05.21	Вик.
7	Економічна частина	25.05.19 – 27.05.21	Вик.
8	Оформлення матеріалів до захисту МКР	27.05.21 – 01.06.21	Вик.

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б Лістинг головної сторінки

```
<template>
                <Container>
                  <AppSpinner
                    v-if="isLoading('getClientWebsites')"
                    :loading="true"
                    margin="250px auto"
                    :size="40"
                  />
                  <SectionsContainer v-else>
                    <TittleWrapper v-if="untilTabletPortrait">
                      <AppTitle
                        text-align="left"
                        font-size="24px"
                        :line-height="1.33"
                      >
                        {{ $t('dashboard') }}
                      </AppTitle>
                    </TittleWrapper>
                    <AppOverlay v-show="showBlurringBlock" />
                    <SectionUpdates
                      v-if="clientPublishedWebsites.length !== 0"
                      :site-id-list="clientPublishedWebsites.map(website => website.id)"
                      :grid-row="untilTabletPortrait ? 'initial' : '1 / 3'"
                    />
                    <CreateWebsite v-else @click="handleCreateWebsite">
                      <template>
                        <NoWebsiteMessage>
                          <h2>{{ $t('noWebsiteCreated') }}</h2>
                          <span class="inactive-text">
                            {{ $t('startWithCreatingFirstWebsite') }}
                          </span>
                        </NoWebsiteMessage>
                      </template>
                    </CreateWebsite>
                    <template v-if="!untilTabletPortrait">
                      <ChartStatistic
                        v-for="(statistic, index) in statistics"
                        :key="index"
                        :store-action="statistic.storeAction"
                        :store-getter="statistic.storeGetter"
                        :title="statistic.title"
                        :no-data-title="statistic.noDataTitle"
                        :no-data-description="statistic.noDataDescription"
                        :chart-stroke="statistic.chartStroke"
                        :chart-offset-fill="statistic.chartOffsetFill"
```

```
            :analytic-key="statistic.analyticKey"
            :total-value-key="statistic.totalValueKey"
            :measure="statistic.measure"
            :prevent-request="statistic.preventRequest"
            :interactive="true"
          />
        </template>
        <ChartStatisticWrapper v-else>
          <ChartStatistic
            v-for="(statistic, index) in statistics"
            :key="index"
            :store-action="statistic.storeAction"
            :store-getter="statistic.storeGetter"
            :title="statistic.title"
            :no-data-title="statistic.noDataTitle"
            :no-data-description="statistic.noDataDescription"
            :chart-stroke="statistic.chartStroke"
            :chart-offset-fill="statistic.chartOffsetFill"
            :analytic-key="statistic.analyticKey"
            :total-value-key="statistic.totalValueKey"
            :measure="statistic.measure"
            :prevent-request="statistic.preventRequest"
            :interactive="false"
          />
        </ChartStatisticWrapper>


        <SectionComments :grid-column="sectionCommentsPosition" />
        <SectionThemesAndPlugins min-height="488px" />
      </SectionsContainer>
    </Container>
  </template>


  <script>
    import styled from 'vue-styled-components';
    import { mapGetters } from 'vuex';
    import AppSpinner from 'components/Display/Loading/AppSpinner';
    import Container from 'components/Layout/Contrainers/ContainerPageBody';
    import CreateWebsite from 'views/Dashboard/CreateWebsite/CreateWebsite';

    import AppOverlay from 'components/Display/Transition/AppOverlay';
    import SectionUpdates from 'views/Dashboard/Sections/SectionUpdates';
    import SectionComments from 'views/Dashboard/Sections/SectionComments';
    import SectionThemesAndPlugins from
  'views/Dashboard/Sections/SectionThemesAndPlugins';
```

```
import ChartStatistic from 'views/Chart/ChartStatistic';
import AppTitle from 'components/Display/Title/AppTitle';
import { capitalizeFirst } from 'utils/utils';

const SectionsContainer = styled('div')`
  display: grid;
  grid-gap: 28px;
  position: relative;
  grid-template-columns: repeat(3, 390px);
  grid-auto-flow: row;
  justify-content: center;
  padding: 48px 4%;
  z-index: 0;

  @media(max-width: 1450px) {
    grid-gap: 24px;
    grid-template-columns: repeat(3, 384px);
  }

  @media (max-width: 1430px){
    grid-template-columns: 1fr 1fr 1fr;
  }

  @media(max-width: 1200px) {
    grid-template-columns: repeat(2, 390px);
  }

  @media(max-width: 991px) {
    padding: 20px;
    grid-template-columns: 384px 1fr;
  }

  @media(max-width: 940px) {
    grid-template-columns: 315px 1fr;
  }

  @media(max-width: 835px) {
    grid-template-columns: 295px 1fr;
  }

  @media(max-width: 770px) {
    grid-gap: initial;
    display: flex;
    flex-direction: column;
  }
```

```
  @media(max-width: 575px) {
    padding-bottom: 16px;
  }
`;

const NoWebsiteMessage = styled('div')`
  text-align: center;
  > h2:first-child {
    margin-bottom: 8px;
  }
`;

const ChartStatisticWrapper = styled('div')`
  position: relative;
  grid-column-start: 2;
  grid-column-end: 4;
  display: grid;
  grid-gap: 24px;
  grid-template-columns: repeat(2, minmax(390px, 21%));

  @media (max-width: 1430px){
    grid-column-start: 2;
    grid-column-end: 2;
    grid-template-columns: 1fr 1fr;
  }

  @media (max-width: 991px){
    grid-gap: initial;
    margin-right: -20px;
    grid-column-start: 1;
    grid-column-end: -1;
    display: flex;
    flex-wrap: nowrap;
    flex-direction: row;
    overflow-x: auto;
    order: -2;

    ::-webkit-scrollbar {
      display: none;
    }

    overflow: -moz-scrollbars-none;
  }

  @media(max-width: 770px) {
    margin-bottom: 24px;
```

```
  }
`;

const TittleWrapper = styled('div')`
  position: relative;
  order: -2;
  grid-column-start: 1;
  grid-column-end: -1;

  @media(max-width: 770px) {
    margin-bottom: 20px;
  }
`;

export default {
  name: 'Dashboard',

  components: {
    AppOverlay,
    AppSpinner,
    AppTitle,
    SectionUpdates,
    SectionComments,
    SectionThemesAndPlugins,
    ChartStatistic,
    Container,
    // DashboardFilters,
    SectionsContainer,
    NoWebsiteMessage,
    ChartStatisticWrapper,
    TittleWrapper,
    CreateWebsite
  },

  data() {
    return {
      showBlurringBlock: false
    };
  },

  computed: {
    ...mapGetters([
      'theme',
      'clientPublishedWebsites',
      'isLoading',
      'mediaQuery'
```

```
      ]),

      untilTabletPortrait() {
        return this.mediaQuery.untilTabletPortrait;
      },

      sectionCommentsPosition() {
        if (this.mediaQuery.onlyTabletLandscape) {
          return 'initial';
        }

        return '1 / 3';
      },

      statistics() {
        return [
          {
            storeAction: 'getUniqueVisitors',
            storeGetter: 'uniqueVisitors',
            title: this.$t('uniqueVisitors'),
            noDataTitle: capitalizeFirst(this.$t('noRecentUniqueVisitors')),
            noDataDescription: this.clientPublishedWebsites.length === 0 ?
this.$t('noWebsitesLackDataReason', {
              article: 'are',
              subject: this.$t('noRecentUniqueVisitors')
            }) : this.$t('noStatisticReason'),
            chartStroke: this.theme.uniqueVisitorsChartGradientStroke,
            chartOffsetFill: this.theme.uniqueVisitorsChartGradientFill,
            analyticKey: '',
            totalValueKey: 'total',
            measure: '',
            preventRequest: false
          },
          {
            storeAction: 'getTotalRequests',
            storeGetter: 'totalRequests',
            title: this.$t('totalRequests'),
            noDataTitle: capitalizeFirst(this.$t('noRecentRequests')),
            noDataDescription: this.clientPublishedWebsites.length === 0
              ? this.$t('noWebsitesLackDataReason', {
                article: 'are',
                subject: this.$t('noRecentRequests')
              })
              : this.$t('noStatisticReason'),
            chartStroke: this.theme.totalRequestsChartGradientStroke,
            chartOffsetFill: this.theme.totalRequestsChartGradientFill,
```

```
        analyticKey: 'pageViews',
        totalValueKey: 'total.pageViews',
        measure: '',
        preventRequest: false
      },
      {
        storeAction: 'getVisitsOverview',
        storeGetter: 'visitsOverview',
        title: this.$t('pageViews'),
        noDataTitle: capitalizeFirst(this.$t('noPageViews')),
        noDataDescription: this.clientPublishedWebsites.length === 0
          ? this.$t('noWebsitesLackDataReason', {
            article: 'are',
            subject: this.$t('noPageViews')
          })
          : this.$t('noStatisticReason'),
        chartStroke: this.theme.chartOrangeStroke,
        chartOffsetFill: this.theme.chartOrangeFill,
        analyticKey: 'pageViews',
        totalValueKey: 'total.pageViews',
        measure: '',
        preventRequest: false
      },
      {
        storeAction: 'getVisitsOverview',
        storeGetter: 'visitsOverview',
        title: this.$t('averageVisitsDuration'),
        noDataTitle: capitalizeFirst(this.$t('noAverageVisitDuration')),
        noDataDescription: this.clientPublishedWebsites.length === 0
          ? this.$t('noWebsitesLackDataReason', {
            article: 'is',
            subject: this.$t('noAverageVisitDuration')
          })
          : this.$t('noStatisticReason'),
        chartStroke: this.theme.averageVisitsDurationChartGradientStroke,
        chartOffsetFill: this.theme.averageVisitsDurationChartGradientFill,
        analyticKey: 'timeAverage',
        totalValueKey: 'total.timeAverage',
        measure: this.$t('sec'),
        preventRequest: true
      }
    ];
  }
},

methods: {
```

```
      handleBlurringBlock(isShow) {
        this.showBlurringBlock = isShow;
      },

      handleCreateWebsite() {
        this.$router.push({ name: 'option-add' });
      }
    }
  };
</script>
```

ДОДАТОК В Лістинг сторіки списку веб-сайтів

```
<template>
               <ContainerPageBody>
                 <!--    <DashboardFilters-->
                 <!--      @filter-open="handleBlurringBlock(true)"-->
                 <!--      @filter-destroy="handleBlurringBlock(false)"-->
                 <!--    />-->
                 <ContainerContent>
                   <AppOverlay v-show="showBlurringBlock" />
                   <AppSpinner v-if="isLoadingActive" :loading="true" class="center"
             :size="40" />
                   <template v-else>
                     <WebsitesHeaderInfo>
                       <AppTitle
                         tag="h1"
                         font-size="24px"
                         :line-height="1.33"
                         :margin-bottom="isUntilMobilePortrait ? '10px' : '0'"
                       >
                         {{ $t('manageWebsites') }}
                       </AppTitle>
                       <AppMultiSelect
                         v-if="itemList.length !== 0"
                         :value="filterBy"
                         :options="filterByOptions"
                         :max-height="500"
                         :searchable="false"
                         :show-labels="false"
                         :show-no-options="false"
                         :allow-empty="false"
                         track-by="name"
                         label="name"
                         width="240px"
                         min-width="auto"
                         @input="setFilterBy"
                       />
                     </WebsitesHeaderInfo>


                     <ListContainer>
                       <ListItem>
                         <ButtonTile
                           :label-text="$t('addNewWebsite')"
                           :height="isUntilMobilePortrait ? '128px' : '346px'"
```

```
              @click="handleCreateWebsite"
            />
          </ListItem>
          <ListItem v-for="(item, index) in itemList" :key="index">
            <WebsiteListItem
              v-if="isWebsite(item)"
              :key="`website_${index}`"
              :item="item"
              :is-until-mobile-portrait="isUntilMobilePortrait"
            />
            <TransferListItem
              v-else-if="isTransfer(item)"
              :key="`transfer_${index}`"
              :item="item"
              :height="isUntilMobilePortrait ? 'auto' : '346px'"
              :is-until-mobile-portrait="isUntilMobilePortrait"
            />
          </ListItem>
        </ListContainer>
      </template>
    </ContainerContent>
  </ContainerPageBody>
</template>


<script>
  import styled from 'vue-styled-components';
  import { mapGetters, mapMutations } from 'vuex';
  import lodashGet from 'lodash/get';
  import AppTitle from 'components/Display/Title/AppTitle';
  import ContainerPageBody from
'components/Layout/Contrainers/ContainerPageBody';
  import AppOverlay from 'components/Display/Transition/AppOverlay';
  import AppSpinner from 'components/Display/Loading/AppSpinner';
  import AppMultiSelect from 'components/Input/Select/AppMultiSelect';
  import ButtonTile from 'components/Input/Button/ButtonTile';
  // import FlexContainer from 'components/Layout/Flex/FlexContainer';
  // import DashboardFilters from '../Dashboard/DashboardFilters';
  import TransferListItem from 'views/WebsiteTransfer/List/TransferListItem';
  import WebsiteListItem from 'views/WebsiteList/WebsiteListItem';
  import { WebsiteStatusList } from 'constants/Website';

  const ContainerContent = styled('div')`
    position: relative;
    height: 100%;
    padding: 48px 5%;
```

```
  box-sizing: border-box;

  @media(max-width: 991px) {
    padding: 24px 20px 16px 20px;
  }

  @media(max-width: 576px) {
    min-height: calc(100vh - 56px);
  }
`;

const ListContainer = styled('div')`
  position: relative;
  display: grid;
  grid-gap: 24px;
  grid-template-columns: repeat(auto-fill, minmax(340px, 1fr));
  grid-auto-flow: row;

  @media(max-width: 767px) {
    grid-template-columns: repeat(auto-fill, minmax(370px, 1fr));
    grid-gap: 15px;
  }

  @media(max-width: 575px) {
    display: block;
  }
`;

const WebsitesHeaderInfo = styled('div')`
  display: flex;
  align-items: center;
  justify-content: space-between;
  margin-bottom: 48px;

  @media(max-width: 991px) {
    margin-bottom: 20px;
  }

  @media(max-width: 575px) {
    flex-direction: column;
    align-items: flex-start;
    justify-content: flex-start;
  }
`;

const ListItem = styled('div')`
```

```
  position: relative;

  @media(max-width: 575px) {
    margin: 0 -20px 8px -20px;

    :last-child {
      margin-bottom: 0;
    }
  }
`;

const SORT_OPTIONS = {
  ALL: 'All websites',
  ACTIVE: 'Active websites',
  PENDING: 'Pending websites',
  TRANSFERS: 'Transfers'
};

export default {
  name: 'WebsiteList',

  components: {
    AppTitle,
    ContainerPageBody,
    ContainerContent,
    AppOverlay,
    AppSpinner,
    ButtonTile,
    AppMultiSelect,
    // DashboardFilters,
    ListContainer,
    WebsiteListItem,
    TransferListItem,
    WebsitesHeaderInfo,
    ListItem
  },

  data() {
    return {
      showBlurringBlock: false,
      filterBy: {
        name: SORT_OPTIONS.ALL
      }
    };
  },
```

```
computed: {
  ...mapGetters([
    'clientWebsites',
    'transferList',
    'isLoading',
    'mediaQuery'
  ]),

  itemList() {
    return this.filterItems(this.filterBy.name);
  },

  filterByOptions() {
    return Object
      .values(SORT_OPTIONS)
      .map(option => ({
        name: option,
        $isDisabled: this.filterItems(option).length === 0
      }));
  },

  isLoadingActive() {
    return this.isLoading('getClientWebsites')
      && this.isLoading('getTransferList')
      && !this.isLoading('deleteWebsite');
  },

  isUntilMobilePortrait() {
    return this.mediaQuery.untilMobilePortrait;
  }
},

methods: {
  ...mapMutations({
    resetWizardState: 'RESET_WIZARD_STATE'
  }),

  handleBlurringBlock(isShow) {
    this.showBlurringBlock = isShow;
  },

  handleCreateWebsite() {
    this.resetWizardState('transferringOptions');
    this.$router.push({ name: 'option-add' });
  },
```

```
      setFilterBy(value) {
        this.filterBy = value;
      },

      filterItems(filter) {
        const allItems = [...this.clientWebsites, ...this.transferList];

        switch (filter) {
          case SORT_OPTIONS.ALL:
            return allItems;
          case SORT_OPTIONS.ACTIVE:
            return allItems
              .filter(item => lodashGet(item, 'status.status') ===
WebsiteStatusList.PUBLISHED);
          case SORT_OPTIONS.PENDING:
            return allItems
              .filter(item => lodashGet(item, 'status.status') ===
WebsiteStatusList.DEVELOPMENT);
          case SORT_OPTIONS.TRANSFERS:
            return allItems
              .filter(item => this.isTransfer(item));
          default:
            return allItems;
        }
      },

      isWebsite(item) {
        return Object.prototype.hasOwnProperty.call(item, 'wpData');
      },

      isTransfer(item) {
        return Object.prototype.hasOwnProperty.call(item, 'transferData');
      }
    }
  };
</script>
```

ДОДАТОК Г Лістинг сторіки аналітики

```
<template>
                <ContainerPageBody>
                  <AnalyticsHeader>
                    <TabsContainer>
                      <TabItem
                        :related-routes="['analytics']"
                        :padding="isUntilTabletPortrait ? '20px 0' : '29px 0 28px 0'"
                      >
                        {{ $t('webAnalytics') }}
                      </TabItem>
                      <TabItem
                        :related-routes="['analytics-resource']"
                        :padding="isUntilTabletPortrait ? '20px 0' : '29px 0 28px 0'"
                      >
                        {{ $t('resourceAnalytics') }}
                      </TabItem>
                    </TabsContainer>
                    <AnalyticsSelectColumn>
                      <AnalyticsSelectWrapper>
                        <AppMultiSelect
                          v-model="filters.domain"
                          :options="filterDomainOptions"
                          track-by="id"
                          label="domain"
                          :placeholder="$t('selectDomain')"
                          :searchable="false"
                          :show-labels="false"
                          :show-no-options="false"
                          :allow-empty="false"
                          :max-height="325"
                          min-width="100%"
                        />
                      </AnalyticsSelectWrapper>
                    </AnalyticsSelectColumn>
                  </AnalyticsHeader>


                  <keep-alive v-if="clientWebsites.length !== 0">
                    <router-view :site-id="filters.domain.id" />
                  </keep-alive>
                  <StateEmpty
                    v-else
                    :title="capitalizeFirst($t('noAnalyticAvailable'))"
```

```
      :sub-title="
        $t('noWebsitesLackDataReason', {
          article: 'is',
          subject: $t('noAnalyticAvailable')
        })
      "
      margin-top="80px"
    />
  </ContainerPageBody>
</template>


<script>
  import styled from 'vue-styled-components';
  import { mapGetters } from 'vuex';
  import AppMultiSelect from 'components/Input/Select/AppMultiSelect';
  import ContainerPageBody from
'components/Layout/Contrainers/ContainerPageBody';
  import TabItem from 'components/Display/Tab/AppTab';
  import StateEmpty from 'views/StatePages/StateEmpty';
  import { capitalizeFirst } from 'utils/utils';

  const TabsContainer = styled('div')`
    display: flex;
    box-sizing: border-box;
  `;

  const AnalyticsHeader = styled('div')`
    display: flex;
    justify-content: space-between;
    margin-bottom: 0!important;
    padding: 0 5%;
    background-color: ${props => props.theme.colorWhite};
    box-sizing: border-box;

    @media(max-width: 991px) {
      flex-direction: column;
      padding-left: 20px;
      padding-right: 20px;
    }
  `;

  const AnalyticsSelectColumn = styled('div')`
    display: flex;
    align-items: center;
    justify-content: flex-end;
```

```
  @media(max-width: 991px) {
    margin: 0 -20px;
    padding: 20px 20px 0 20px;
    background-color: ${props => props.theme.colorSelectActiveItem};
  }

  @media(max-width: 374px) {
    flex-direction: row;
    justify-content: center;
  }
`;

const AnalyticsSelectWrapper = styled('div')`
  min-width: 240px;
  width: 100%;

  @media(max-width: 991px) {
    min-width: initial;
    max-width: 240px;
  }

  @media(max-width: 479px) {
    max-width: 100%;
  }
`;

export default {
  name: 'Analytics',

  components: {
    AppMultiSelect,
    TabItem,
    ContainerPageBody,
    StateEmpty,
    TabsContainer,
    AnalyticsHeader,
    AnalyticsSelectColumn,
    AnalyticsSelectWrapper
  },

  data() {
    return {
      filters: {
        domain: this.getFilterDomainDefault()
      }
```

```
      };
    },

    computed: {
      ...mapGetters([
        'clientWebsites',
        'clientWebsitesDropdownOptions',
        'mediaQuery'
      ]),

      filterDomainOptions() {
        return [
          this.getFilterDomainDefault(),
          ...this.clientWebsitesDropdownOptions
        ];
      },

      isUntilTabletPortrait() {
        return this.mediaQuery.untilTabletPortrait;
      }
    },

    methods: {
      getFilterDomainDefault() {
        return {
          id: '',
          domain: this.$t('all')
        };
      },

      capitalizeFirst
    }
  };
</script>
```

ДОДАТОК Д Лістинг сторіки списку веб-сайтів

```
<template>
               <TabManageWebsiteContainer>
                 <AppHeadLine
                   :title="$t('contentAssistantTitle', { name: currentUser.firstName })"
                   :sub-title="$t('contentAssistantSubTitle')"
                 />
                 <AppLine :margin="mediaQuery.untilTabletPortrait ? '25px 0 0 0' : '51px 0 0
            0'" />
                 <TabManageNavigation>
                   <TabManageNavigationSearch>
                     <AppInput
                       v-model="form.search"
                       padding="10px 16px 10px 40px"
                       :background="`url(
                       ${require('@svg/search-secondary.svg')}) 16px 50% / 16px 16px no-
            repeat`"
                       :placeholder="searchPlaceHolder"
                     />
                   </TabManageNavigationSearch>
                   <TabManageNavigationSelect>
                     <AppMultiSelect
                       :value="valueOfClientWebsitesDropdown"
                       :options="clientWebsitesDropdownOptions"
                       :max-height="500"
                       :placeholder="$t('selectDomain')"
                       :searchable="false"
                       :show-labels="false"
                       :show-no-options="false"
                       :allow-empty="false"
                       :input-label="mediaQuery.untilMobilePortrait ? '' :
            $t('inputSelectLabelSelectWebsite')"
                       display="flex"
                       width="100%"
                       min-width="auto"
                       track-by="id"
                       label="domain"
                       label-width="120px"
                       label-padding="0 16px 0 0"
                       label-font-weight="500"
                       label-margin-bottom="0"
                       label-vertical-position="center"
                       label-display="flex"
                       @input="handleSetWebsiteUnderEditing"
```

```
    />
  </TabManageNavigationSelect>
</TabManageNavigation>


<SpinnerWrapper v-if="isLoadingList">
  <AppSpinner
    :loading="true"
    :size="40"
    class="center"
  />
</SpinnerWrapper>
<template v-else>
  <TabManageTableWrapper v-if="tableItemListFiltered.length">
    <TabManageTable>
      <TabManageTableHeaderRow>
        <TabManageTableHeaderColumn>
          {{ $t('title') }}
        </TabManageTableHeaderColumn>
        <TabManageTableHeaderColumn>
          <TabManageTableSort @click="handleSortClick">
            {{ $t('publishedOn') }}
            <IconBaseSvg
              :name="sortType === 'date-up' ? 'arrow-down' : 'arrow-up'"
              width="16px"
              height="16px"
              margin="-1px 0 0 4px"
              vertical-align="text-top"
            />
          </TabManageTableSort>
        </TabManageTableHeaderColumn>
        <TabManageTableHeaderColumn>
          {{ $t('blog') }}
        </TabManageTableHeaderColumn>
        <TabManageTableHeaderColumn>
          {{ $t('author') }}
        </TabManageTableHeaderColumn>
        <TabManageTableHeaderColumn>
          {{ $t('type') }}
        </TabManageTableHeaderColumn>
        <TabManageTableHeaderColumn>
          {{ $t('actions') }}
        </TabManageTableHeaderColumn>
      </TabManageTableHeaderRow>
      <TabManageTableRow
        v-for="(item, index) in tableItemListBySort"
```

```
            :key="index"
            :item="item"
          />
        </TabManageTable>
      </TabManageTableWrapper>
      <StateEmpty
        v-else
        margin-top="80px"
        :title="$t('noItemsMatchSearchCriteria')"
      />
    </template>
  </TabManageWebsiteContainer>
</template>


<script>
  import styled from 'vue-styled-components';
  import { mapGetters, mapMutations, mapActions } from 'vuex';
  import AppMultiSelect from 'components/Input/Select/AppMultiSelect';
  import AppLine from 'components/Display/Shape/AppLine';
  import AppInput from 'components/Input/Input/AppInput';
  import AppSpinner from 'components/Display/Loading/AppSpinner';
  import IconBaseSvg from 'components/Display/Icons/IconBaseSvg';
  import AppHeadLine from 'components/Display/HeadLine/AppHeadLine';
  import TabManageTableRow from
'views/Content/ContentManageWebsite/TabManageTableRow';
  import StateEmpty from 'views/StatePages/StateEmpty';

  const SortType = {
    DATE_UP: 'date-up',
    DATE_DOWN: 'date-down'
  };

  const TabManageTableHeaderColumn = styled('div')`
    ${props => `
      color: ${props.theme.manageWebsiteTableHeaderTextColor};

      :last-child {
        text-align: center;
      }
    `}
  `;

  const TabManageWebsiteContainer = styled('div')`
    padding: 0 5% 109px 5%;
    * {
```

```
    box-sizing: border-box;
  }

  @media(max-width: 991px) {
    padding: 20px;
  }
`;

const TabManageTableSort = styled('button')`
  display: inline-flex;
  align-items: center;
  padding: 0;
  font-size: 14px;
  line-height: 21px;
  font-family: inherit;
  color: inherit;
  background-color: transparent;
  border: none;
  cursor: pointer;

  &:hover {
    opacity: 0.6;
  }

  &:focus {
    outline: none;
  }
`;

const TabManageNavigation = styled('div')`
  margin-top: 47px;
  display: flex;
  justify-content: space-between;

  @media(max-width: 991px) {
     margin-top: 25px;
  }

  @media(max-width: 730px) {
    display: block;
  }
`;

const TabManageNavigationSearch = styled('div')`
  width: 100%;
  max-width: 584px;
```

```
    flex-basis: 584px;

    @media(max-width: 1200px) {
      max-width: 358px;
      flex-basis: 358px;
    }

    @media(max-width: 991px) {
      max-width: 280px;
      flex-basis: 280px;
    }

    @media(max-width: 730px) {
      max-width: 100%;
      width: 100%
    }
  `;

  const TabManageNavigationSelect = styled('div')`
    width: 100%;
    max-width: 380px;
    flex-basis: 380px;

    @media(max-width: 730px) {
      margin-top: 10px;
      max-width: 100%;
      width: 100%
    }

    @media(max-width: 576px) {
      .multiselect {
        width: 100%;
      }
    }
  `;

  const TabManageTable = styled('div')`
    ${props => `
      margin-top: 46px;
      padding: 37px 0 0;
      background-color: ${props.theme.colorWhite};
      box-shadow: 0 1px 2px 0 ${props.theme.tabletManageWebsiteBoxShadow};

      @media(max-width: 991px) {
        width: 1000px;
      }
```

```
    @media(max-width: 576px) {
      margin-top: 25px;
    }
  `}
`;

const TabManageTableWrapper = styled('div')`
  overflow-x: auto;
`;

const TabManageTableHeaderRow = styled('div')`
  display: grid;
  grid-gap: 10px;
  grid-template-columns: 22% 13% 27% 15% 12% 1fr;
  padding: 0 24px 12px 24px;

   @media(max-width: 1300px) {
    grid-template-columns: 20% 13% 27% 14% 10% 1fr;
  }

  @media(max-width: 1100px) {
    grid-template-columns: 20% 14% 23% 14% 10% 1fr;
  }
`;

const SpinnerWrapper = styled('div')`
   position: relative;
   margin-top: 46px;
   min-height: 250px;
`;

export default {
  name: 'TabContentManageWebsite',

  components: {
    AppLine,
    AppInput,
    AppMultiSelect,
    AppHeadLine,
    IconBaseSvg,
    AppSpinner,
    StateEmpty,
    TabManageWebsiteContainer,
    TabManageNavigation,
    TabManageNavigationSearch,
```

```
    TabManageNavigationSelect,
    TabManageTable,
    TabManageTableRow,
    TabManageTableHeaderRow,
    TabManageTableSort,
    SpinnerWrapper,
    TabManageTableHeaderColumn,
    TabManageTableWrapper
  },

  data() {
    return {
      form: {
        search: ''
      },
      sortType: SortType.DATE_UP
    };
  },

  computed: {
    ...mapGetters([
      'theme',
      'currentUser',
      'clientPublishedWebsites',
      'clientWebsitesDropdownOptions',
      'websiteUnderEditing',
      'isLoading',
      'websitePageListCurrent',
      'websiteBlogPostListCurrent',
      'mediaQuery'
    ]),

    valueOfClientWebsitesDropdown() {
      return this.clientWebsitesDropdownOptions.find((website) => {
        return website.id === this.websiteUnderEditing.id;
      });
    },

    isLoadingList() {
      return this.isLoading('getWebsitePageList');
    },

    tableItemList() {
      return [
        ...this.websitePageListCurrent,
        ...this.websiteBlogPostListCurrent
```

```
    ];
  },

  tableItemListFiltered() {
    const filter = this.form.search.toUpperCase();
    return this.tableItemList.filter((option) => {
      const matchText = (option.title && option.title.rendered) || '';
      return matchText.toUpperCase().indexOf(filter) > -1;
    });
  },

  tableItemListBySort() {
    const itemsList = this.tableItemListFiltered.slice();

    switch (this.sortType) {
      case SortType.DATE_UP:
        return itemsList.sort((a, b) => {
          return Number(new Date(b.date)) - Number(new Date(a.date));
        });
      case SortType.DATE_DOWN:
        return itemsList.sort((a, b) => {
          return Number(new Date(a.date)) - Number(new Date(b.date));
        });
      default:
        return [];
    }
  },

  searchPlaceHolder() {
    if (this.mediaQuery.untilTabletPortrait) {
      return this.$t('tabContentManageWebsitePlaceholderMobile');
    }

    return this.$t('tabContentManageWebsitePlaceholder');
  }
},

created() {
  this.fetchMainData();
},

methods: {
  ...mapActions([
    'getWebsiteBlogPostList',
    'getUserList'
  ]),
```

```
...mapMutations({
  setWebsiteUnderEditing: 'SET_WEBSITE_UNDER_EDITING'
}),

handleSortClick() {
  if (this.sortType === SortType.DATE_UP) {
    this.sortType = SortType.DATE_DOWN;
  } else {
    this.sortType = SortType.DATE_UP;
  }
},

handleSetWebsiteUnderEditing({ id }) {
  const website = this.getWebsiteById(id);
  if (!website) {
    return;
  }

  this.setWebsiteUnderEditing(website);
  this.updateUrl();
  this.fetchMainData();

  this.$emit('replace-website', id);
},

fetchWebsiteBlogPostList() {
  this.getWebsiteBlogPostList({
    parameters: {
      siteId: this.websiteUnderEditing.id,
      per_page: 100,
      context: 'embed'
    },
    cachingEnabled: true
  }).catch(() => {
    this.$toast.error(this.$t('apiErrorMessage'));
  });
},

fetchUserList() {
  return this.getUserList({
    parameters: {
      siteId: this.websiteUnderEditing.id,
      context: 'edit',
      per_page: 100
    },
```

```
            cachingEnabled: true
          }).catch(() => {
            this.$toast.error(this.$t('apiErrorMessage'));
          });
        },

        fetchMainData() {
          this.fetchWebsiteBlogPostList();
          this.fetchUserList();
        },

        updateUrl() {
          this.$router.replace({
            name: this.$route.name,
            params: { ...this.$route.params, siteId: this.websiteUnderEditing.id }
          });
        },

        getWebsiteById(id) {
          return this.clientPublishedWebsites.find((website) => {
            return website.id === id;
          });
        }
      }
    };
</script>
```

Додаток Е – Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доц. кафедри ПЗ _____ В.В. Войтко

Рецензент, д.т.н., проф. кафедри КН _____ О.М. Василевський

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____В.В. Войтко

Виконавець, студентка групи 1ПІ–19м _____ В. В. Донченко

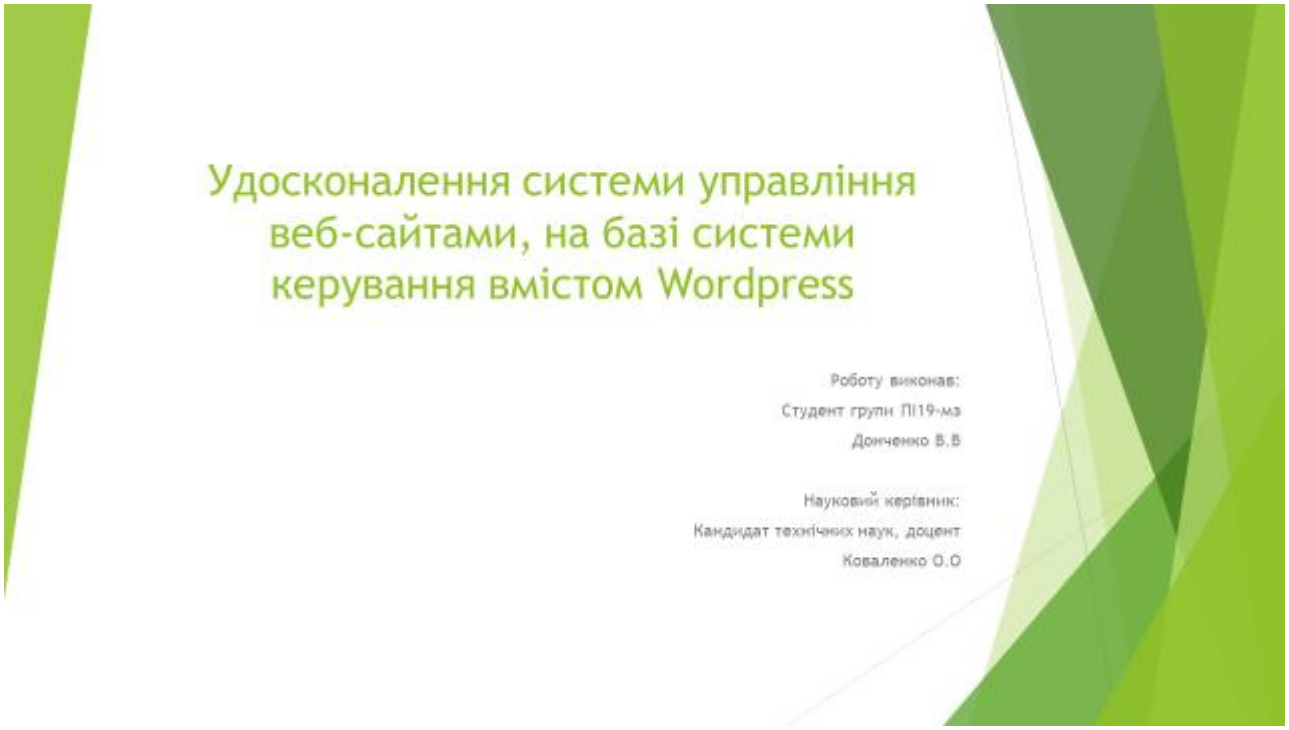

Рисунок Е.1 - Титульна сторінка

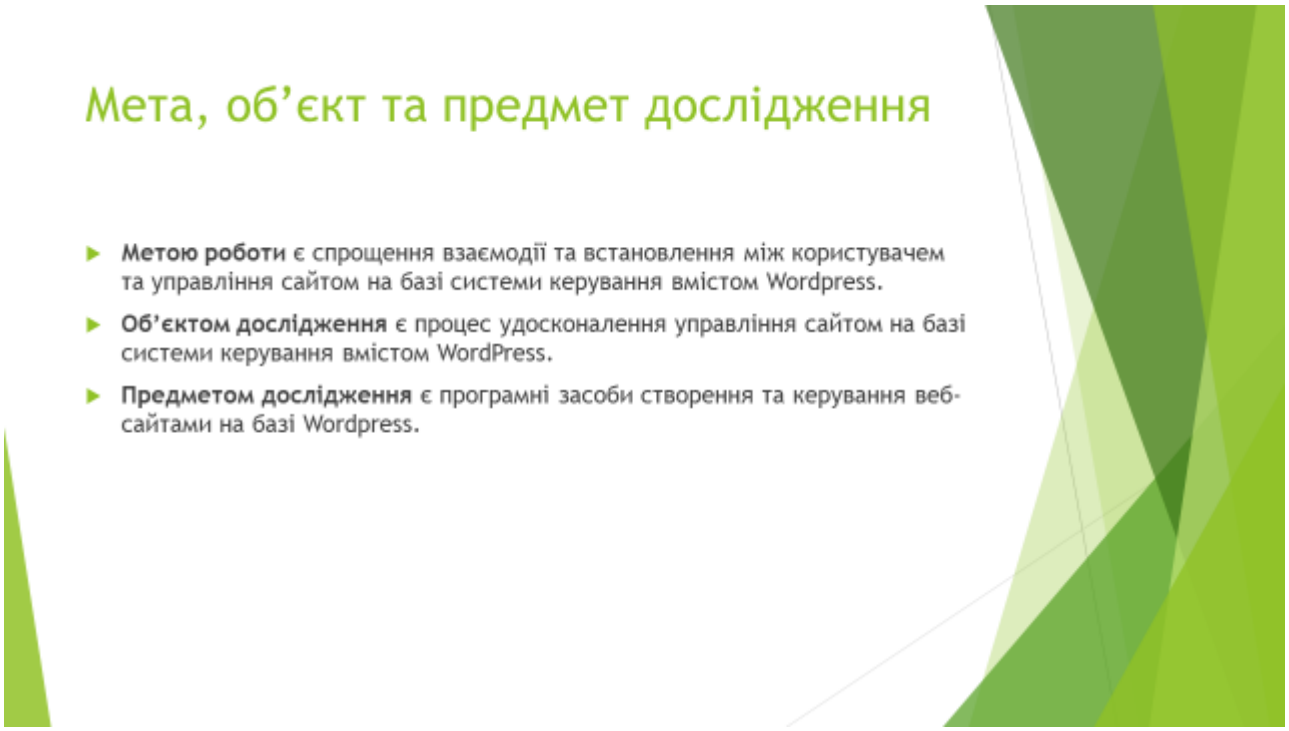

Рисунок Е.2 - Мета, об’єкт та предмет дослідження

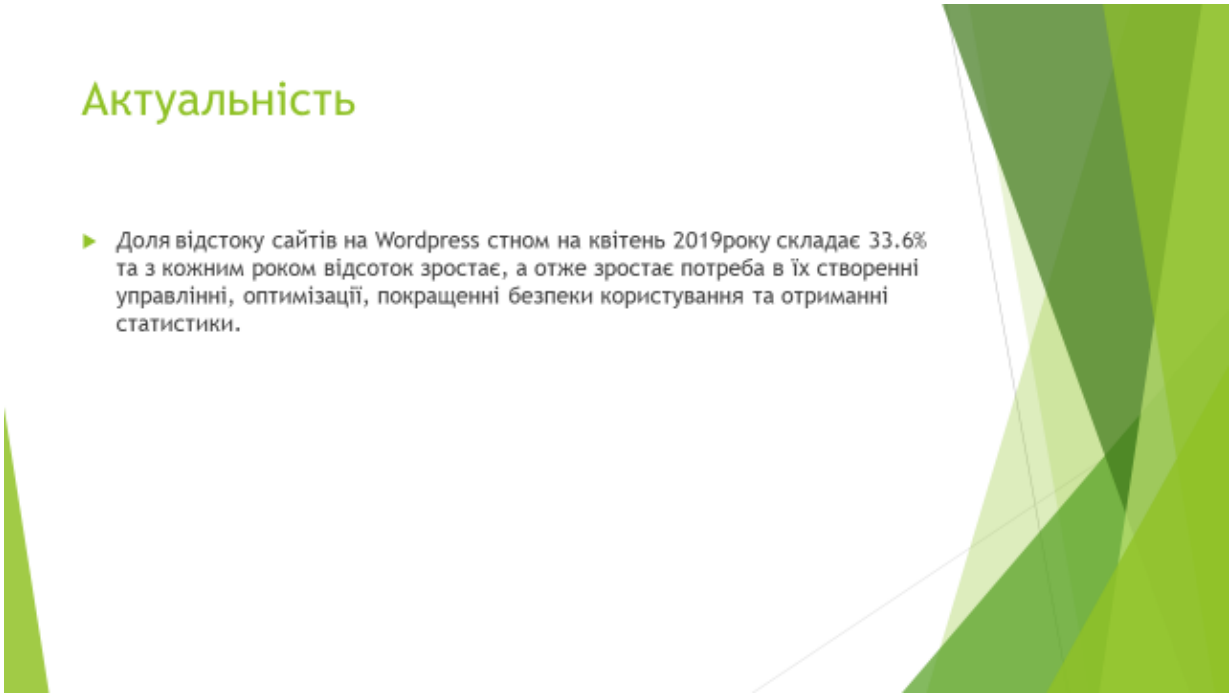

Рисунок Е.3 - Актуальність розробки

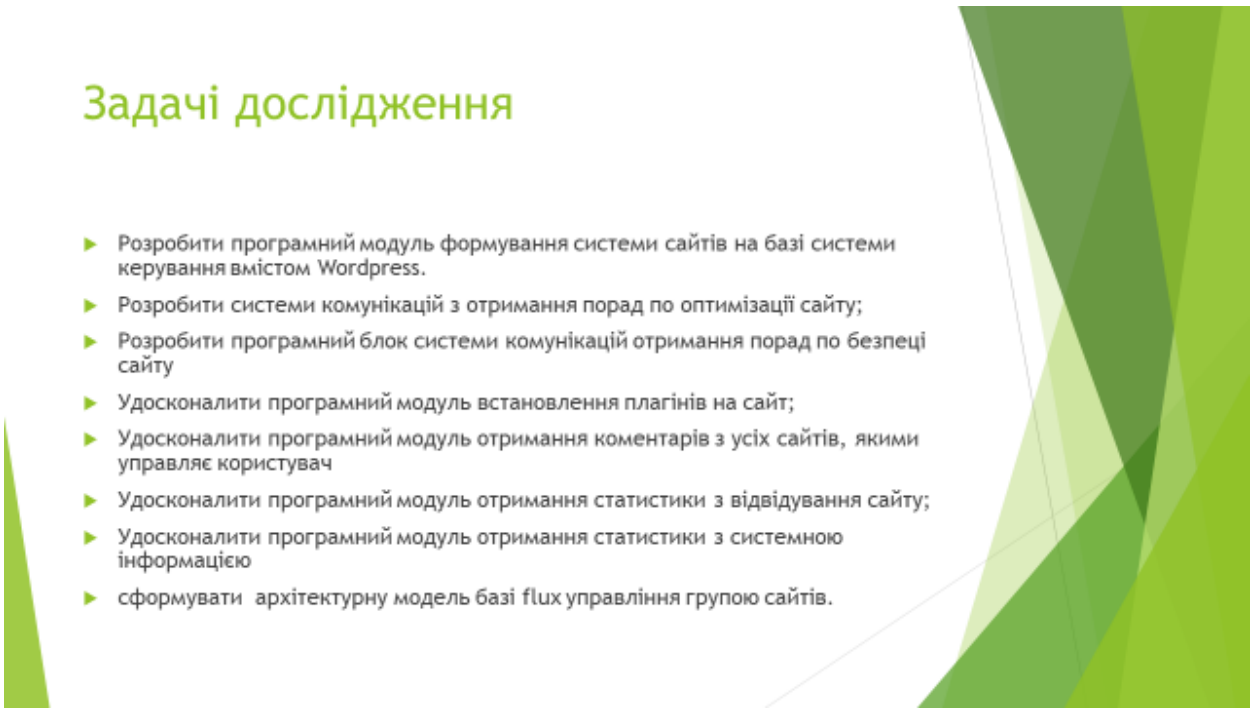

Рисунок Е.4 - Задачі дослідження

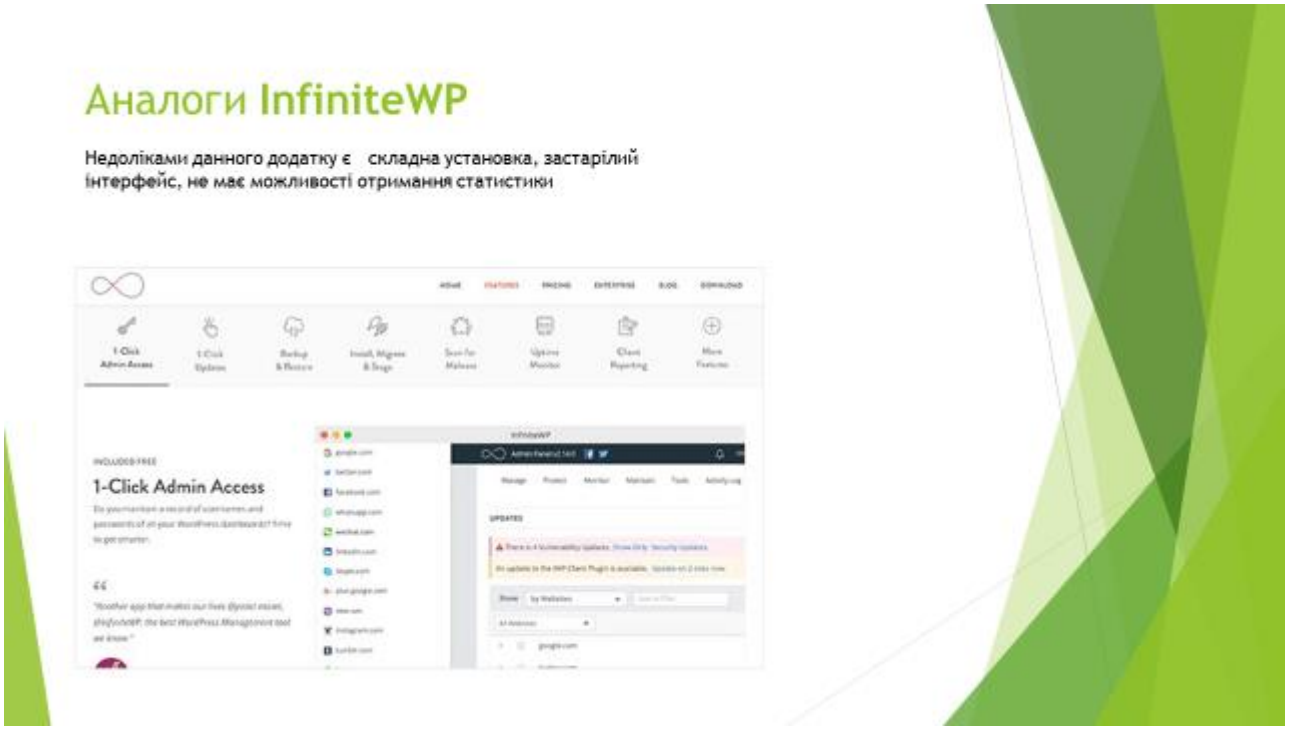

Рисунок Е.5 - Аналоги (InfiniteWP)

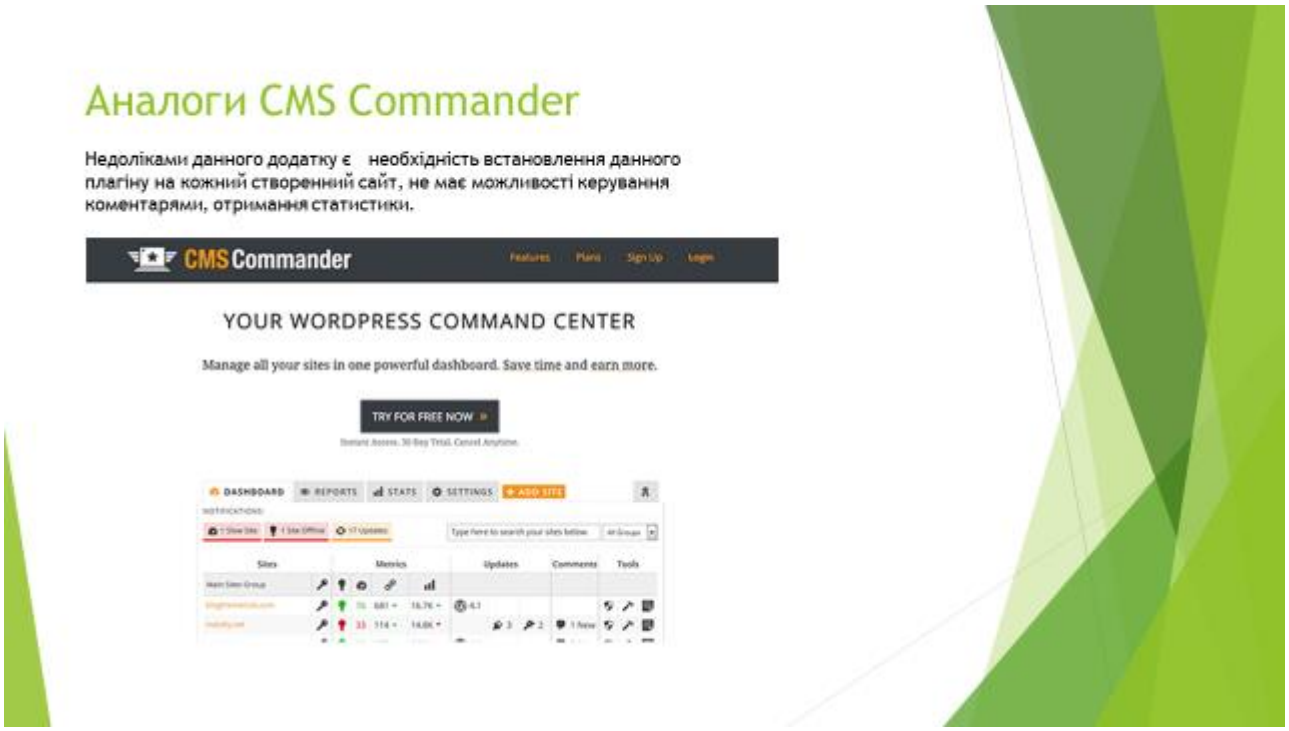

Рисунок Е.6 - Аналоги (CMS Commander)

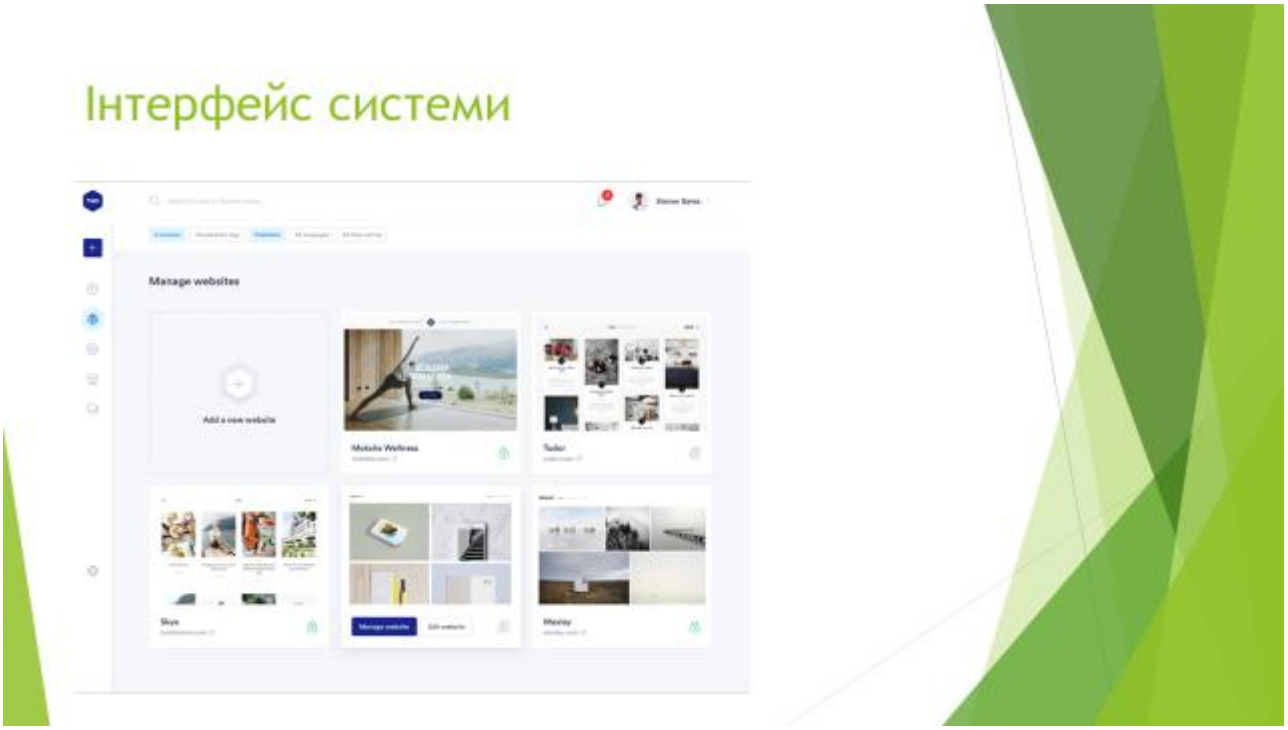

Рисунок Е.7 - Інтерфейс списку веб-сайтів

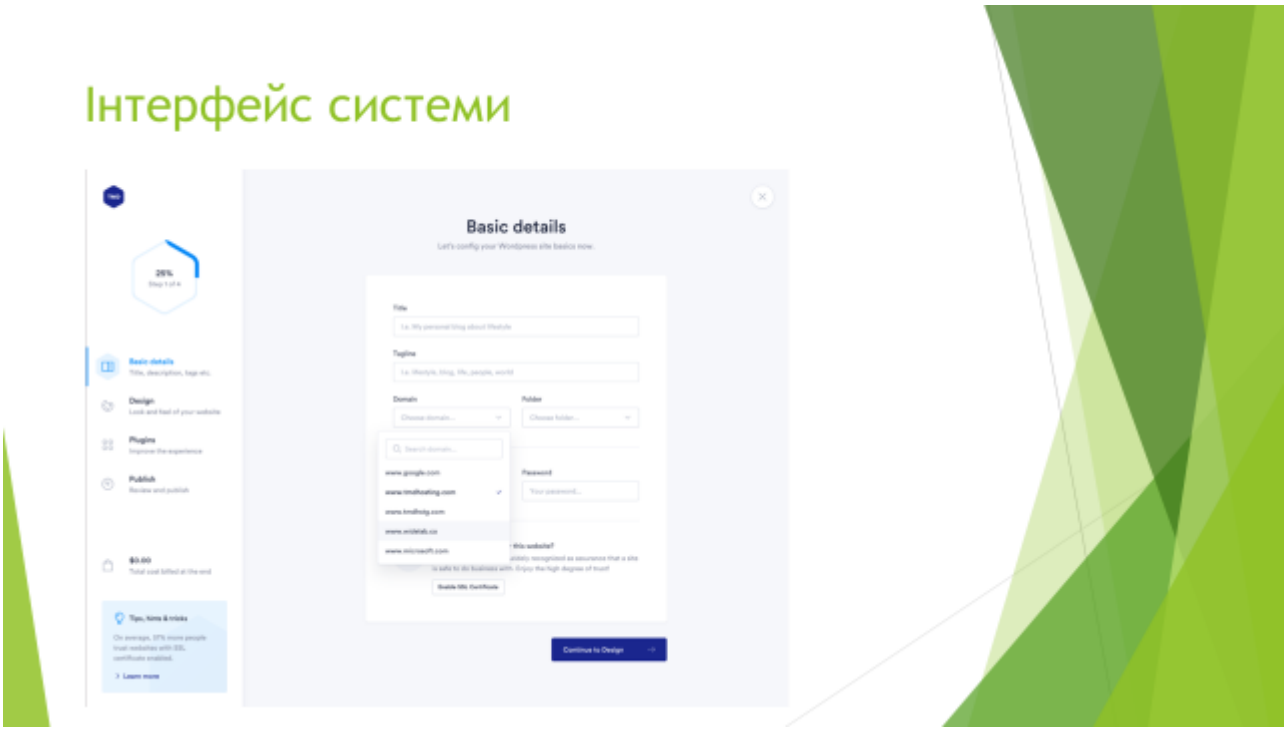

Рисунок Е.8 - Інтерфейс створення сайту

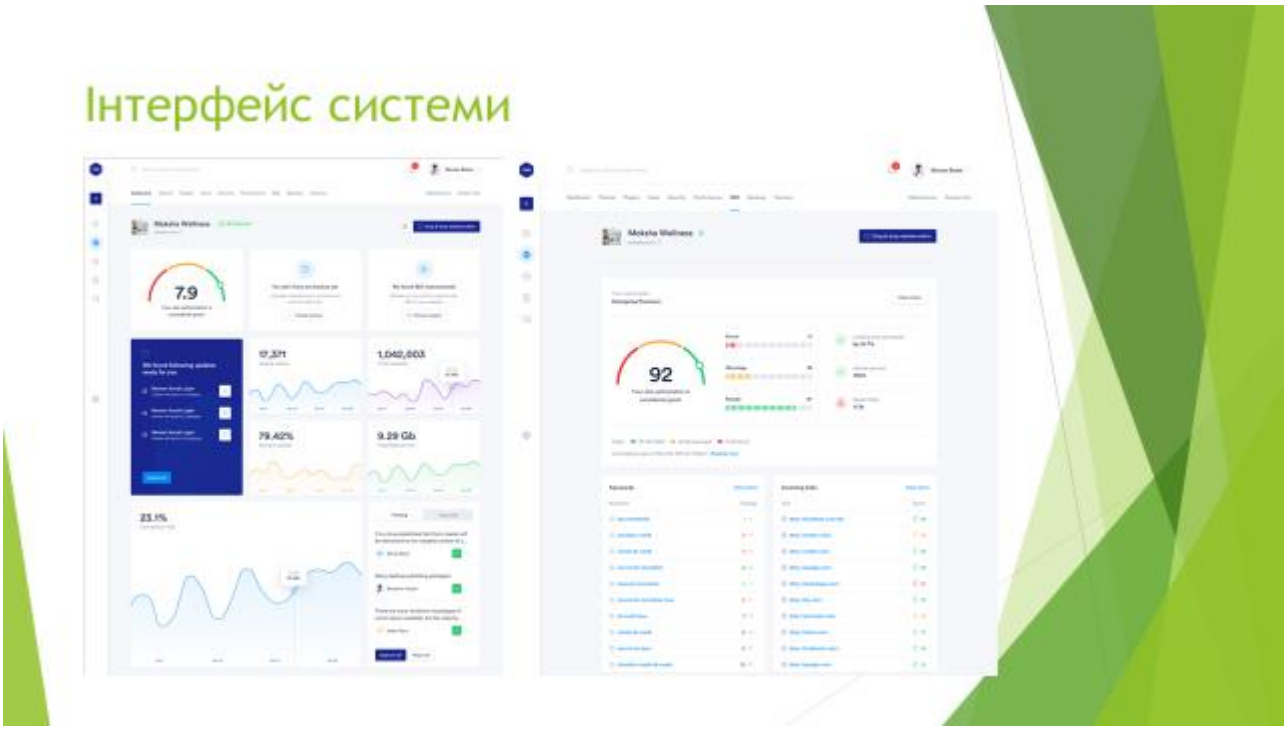

Рисунок Е.9 - Інтерфейс вкладки приладів та оптимізації

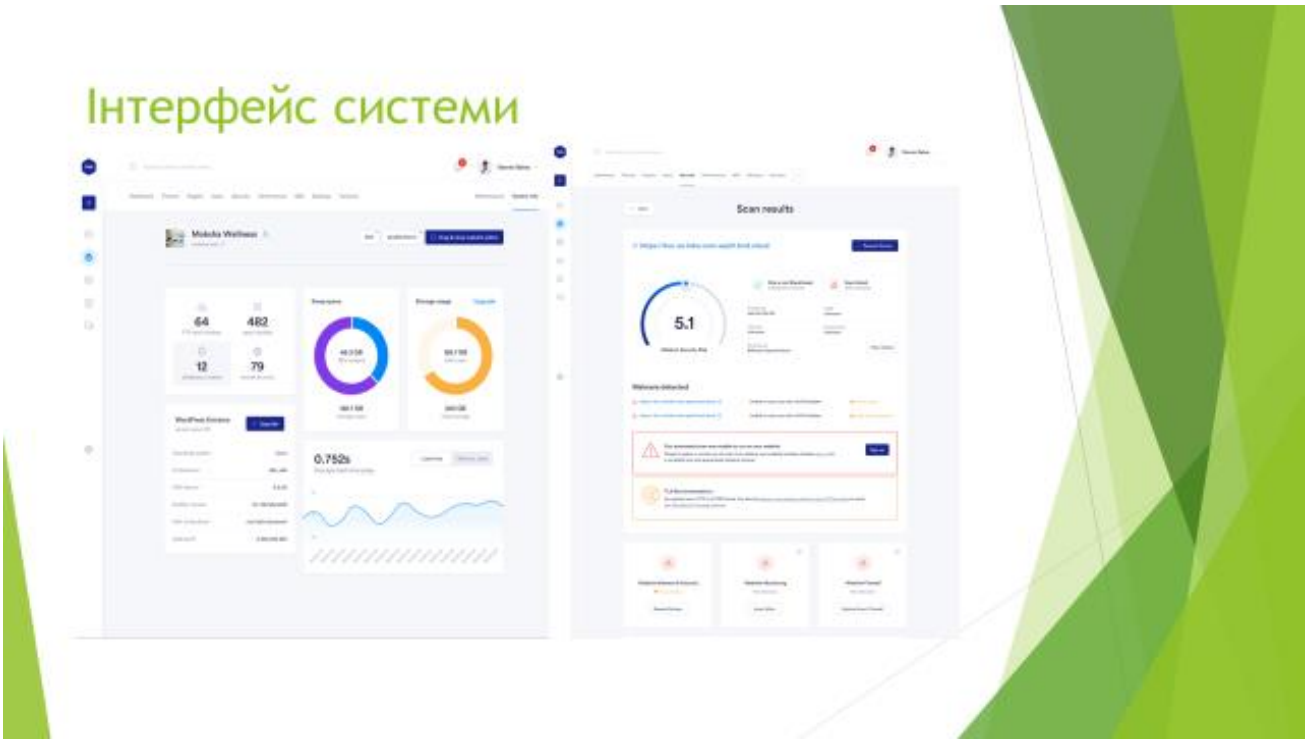

Рисунок Е.10 - Інтерфейс вкладки безпеки та статистики

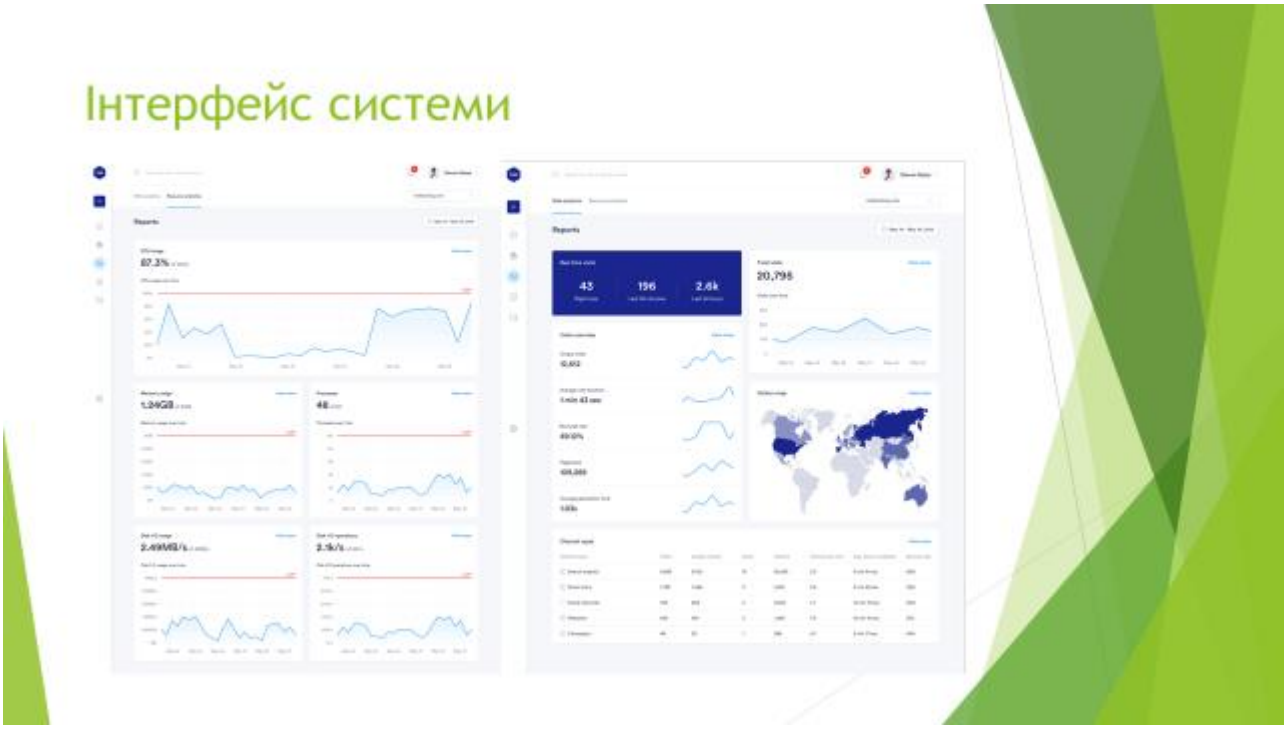

Рисунок Е.11 - Інтерфейс сторінки загальної аналітики

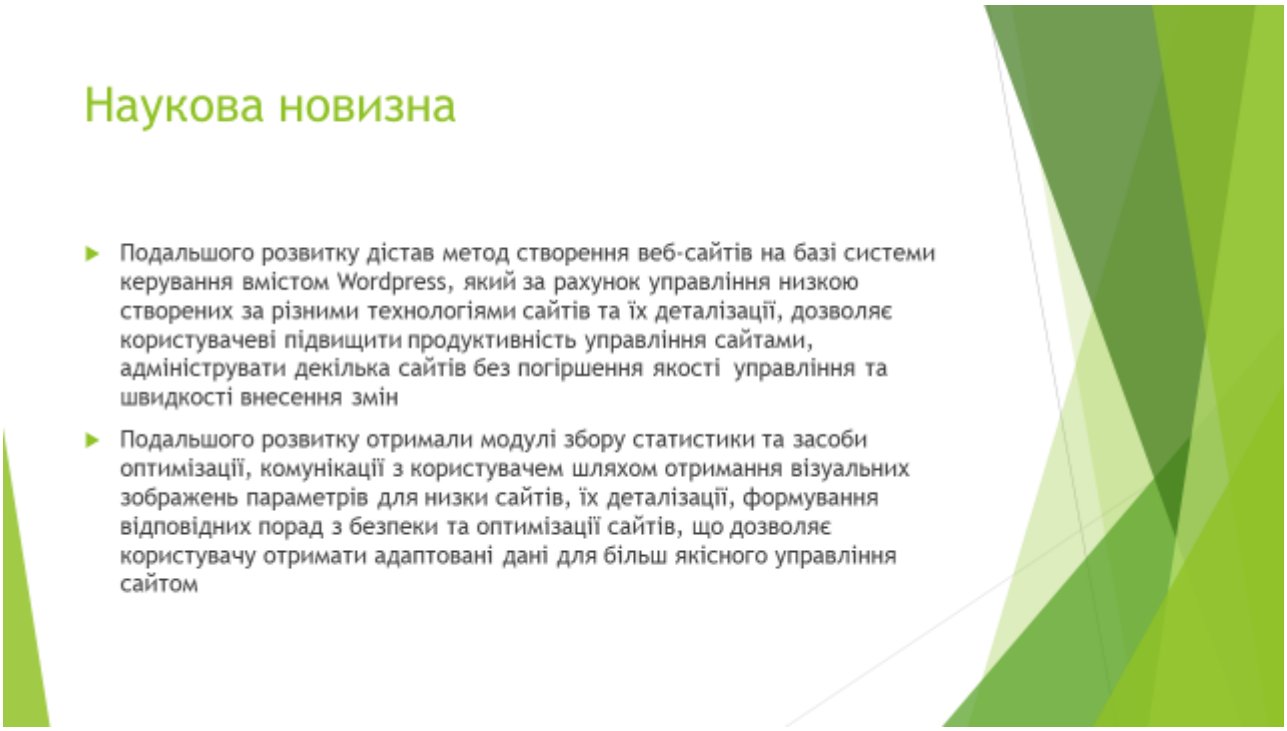

Рисунок Е.12 - Наукова новизна

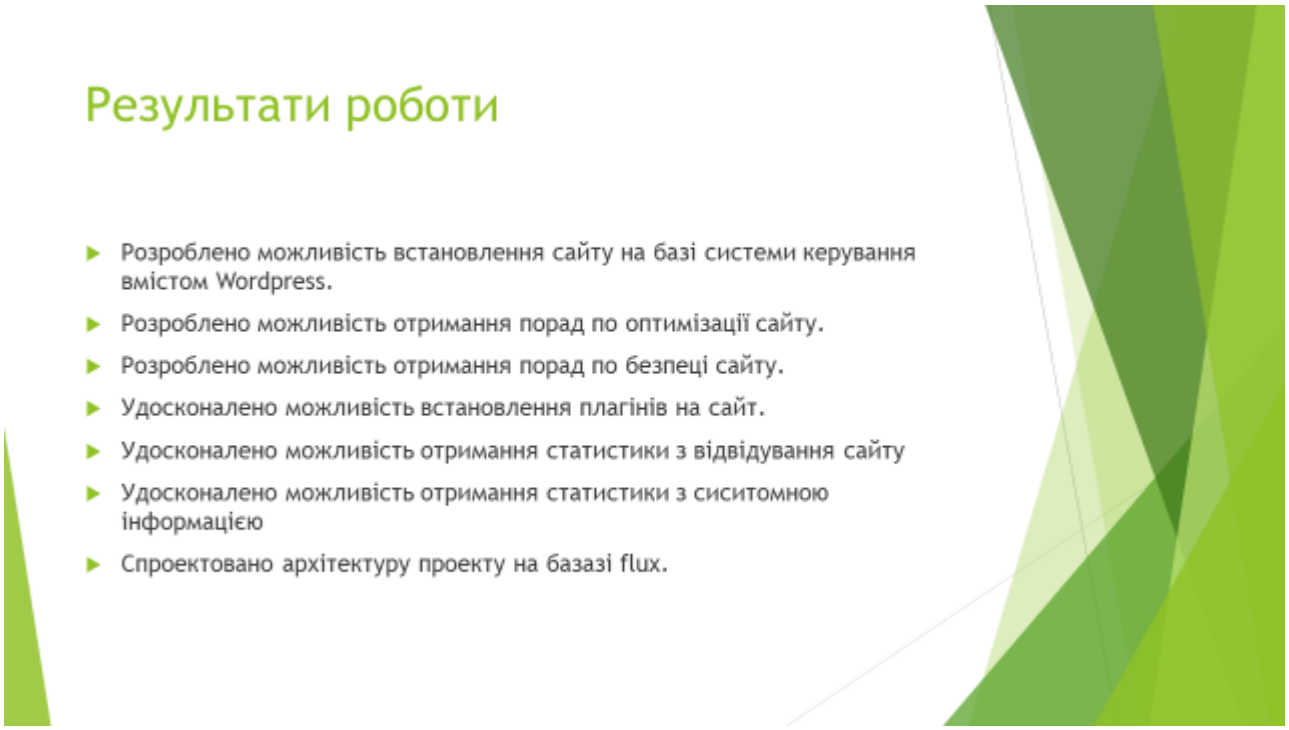

Рисунок Е.13 - Результати роботи

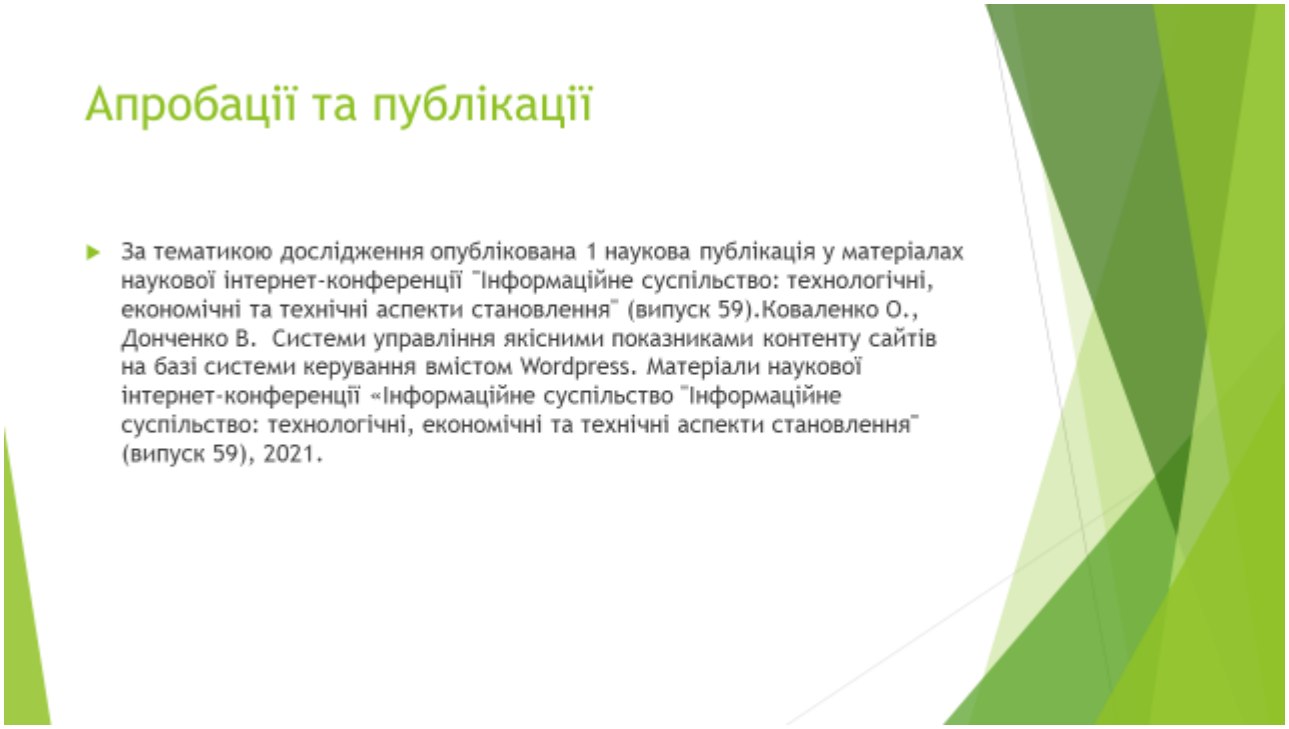

Рисунок Е.14 - Апробації та публікаці