МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МАРУЩАК ДЕНИС РУСЛАНОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т. В. Нескородєва
«____» _____ 2022р.

ДОСЛІДЖЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ОРЕНДИ СЕРВЕРІВ ТА ДОМЕНІВ НА ОСНОВІ КОНТЕНТНОЇ ФІЛЬТРАЦІЇ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Федоров Є. Є., професор кафедри
інформаційних технологій
д.т.н., професор
_____
(підпис)

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

Вінниця – 2022

АНОТАЦІЯ

Магістерська кваліфікаційна робота спрямована на удосконалення та дослідження рекомендаційної системи для оренди серверів та доменів на основі контентної фільтрації.

У магістерській кваліфікаційній роботі удосконалено та досліджено рекомендаційну систему з контентною фільтрацією для оренди серверів та доменів шляхом розрозробки інтерфейсу та удосконалення API на базі WHMCS. Розроблена система призначення для створення, редагування вмісту, встановлення SSL сертифікатів, отримання статистики, порад по оптимізації та безпеці роботи серверів та доменів, перестановка додатків для серверів, а також підбір потрібних серверів та доменів під вимоги користувача за допомогою контентної фільтрації.

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 based on a follow-up recommendation system for ordering servers and domains based on content filtering.

At the master's qualification robot, the recommendation system was improved for ordering servers and domains in the way of expanding the interface and improving the API based on WHMCS. The system has been broken down for creating, editing in place, installing SSL certificates, removing statistics, optimizing and securing websites, rearranging themes and plugins, as well as picking up servers and domains for additional content filtering.

Creations of a software product written in the mobile programming JavaScript, with the Vue.js framework. The web addendum is a SPA, that is a software product, quick and easy to peel and secure to the crust close to the crust of the flooring.

Зміст

ВСТУП

З появленням інтернету кількість інформації, з якою люди щодня стикаються, значно зросла.

Це означає, що люди повинні орієнтуватися серед надзвичайно великої кількості доступних альтернатив, коли хочуть щось знайти. Але з іншого боку виступають власники інтернет-магазинів і сервісів: вони зацікавлені в персональній рекламі і рекомендаціях кожному конкретному користувачеві, тому що такий підхід може значно збільшити прибуток компанії. Як результат, в останні роки інтерес до розробки і поліпшення існуючих рекомендаційних систем значно виріс.

Рекомендаційні системи – це програми, головна мета яких полягає у формуванні рекомендацій різних продуктів або сервісів для користувачів на основі їх переваг.

Сервіси збирають інформацію про переваги користувачів і намагаються запропонувати їм корисні товари. На даний момент існує множина методів для формування рекомендацій, але всі вони мають свої переваги і недоліки. Саме тому дослідження в даній області актуальні. Проблема особливо актуальна для нових, щойно створених систем. Проблема релевантності оцінок часто виникає в разі холодного старту, оскільки нові об'єкти або користувачі ускладнюють створення релевантних рекомендацій.

Обґрунтування вибору теми дослідження. Кожного дня попит на оренду доменів та серверів зростає, разом з цим зростає необхідність у швидкому створенні, адмініструванні, та оптимізації користувацького інтерфейсу а також систем налаштування або фільтрації для того чи іншого серверу або домену. На даний момент існує безліч АПІ, які дають можливість реалізувати це, проте не має сервісу, який міг би зібрати це все у одному місці і надати саме те що потрібно користувачу за допомогою контентної фільтрації та рекомендаційної системи.

Серед головних задач користувача додатком можна виділити такі як можливість швидкої фільтрації серверів та доменів, зміни акцентів на інформації відносно її актуальності, динамічного запровадження програмних засобів для

оптимізації домену або серверу та його безпеки, та рекомендації щодо вибору того чи іншого сервісу. Серед відомих програмних додатків, що дозволяють орендувати, відфільтровувати, та налаштувати сервер або домен, 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.

Платформа TMD Hosting є достатньо популярною серед власників та адміністраторів серверів, але процеси встановлення серверу та домену, формування рекомендацій щодо того який сервіс обрати, фільтрації доменів та серверів, отримання статистики з відвідуванням серверу та інших аналітичних даних потребує свого удосконалення.

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

Мета та завдання дослідження. Метою роботи є спрощення взаємодії між користувачем, встановленням та управління сервером за допомогою контентної фільтрації на базі системи керування вмістом TMD Hosting.

Системи електронної комерції пропонують персональний перелік товарів або послуг для споживача, що і забезпечило їх швидкий розвиток в останнє десятиріччя. Для побудови рекомендованого списку об'єктів в системах ecommerce використовується рекомендаційна підсистема. Дана підсистема формує рекомендації в офлайн- та онлайн-режимах. В першому режимі при побудові персонального переліку товарів або послуг використовується вся наявна інформація про минулі покупки споживача та про характеристики 8 цікавих для нього об'єктів. Такі рекомендації готуються заздалегідь, до входу користувача в систему електронної комерції. В онлайн-режимі виконується уточнення офлайн-рекомендацій. Онлайн-режим ставить підвищені вимоги до швидкості побудови рекомендацій з урахуванням особливостей поточної поведінки споживача. Проведений аналіз методів побудови рекомендацій показав, що існуючі підходи розділяються на дві групи:

– на основі схожості товарів або послуг; предмети до рекомендації підбираються на основі подібності їх властивостей, які становлять цінність для споживача;

– на основі схожості користувачів; подібність вимог споживачів розраховується виходячи із історії покупок та рейтингів їх та схожих користувачів рекомендаційної системи.

При формуванні рекомендацій в онлайн-режимі необхідно комбінувати ці методи, в залежності від характеристик поточного споживача. Наприклад, для нового користувача необхідно буде показувати ті товари, які користуються популярністю. Для водимого споживача необхідно враховувати останні дані про перегляд сторінок сайту електронної комерції і, на цій основі, уточнити попередні рекомендації.

Для того щоб вирішити проблеми, що були наведені вище, необхідно використовувати рекомендаційні системи. Рекомендаційні системи - це інформаційні системи, що надають пропозиції щодо предметів, які будуть корисними для користувача. Пропозиції стосуються різних процесів прийняття рішень, наприклад, які товари купувати, яку музику слухати чи які новини в Інтернеті читати. Прикладом цього є система рекомендування книг, яка допомагає користувачам вибрати книгу для читання.

Оскільки рекомендації зазвичай персоналізовані, різні користувачі або групи користувачів отримують різноманітні пропозиції. Крім того, є також неперсоналізовані рекомендації. Їх набагато простіше створити, і вони зазвичай розміщуються в журналах чи газетах. Типовими прикладами є десятка найкращих книжок, компакт-дисків тощо. Хоча вони можуть бути корисними 9 та ефективними у певних ситуаціях, ці типи неперсоналізованих рекомендацій зазвичай не враховуються дослідженнями РС.

У цій роботі ми зосередимось на двох основних стратегіях методах формування онлайн-рекомендацій:

– контентна фільтрація;

– колаборативна фільтрація.

Контентний підхід створює профіль для кожного користувача чи продукту для характеристики його природи. Як приклад, профіль фільму може містити атрибути, що стосуються його жанру, акторів-учасників, популярності касових зборів тощо. Профілі користувачів можуть містити демографічну інформацію або відповіді на відповідну анкету. Отримані профілі дозволяють програмам асоціювати користувачів із відповідними продуктами. Однак стратегії, що базуються на вмісті, вимагають збору зовнішньої інформації, яка може бути недоступною або простою для збору.

Колаборативна фільтрація (КФ) - термін, придуманий розробниками першої системи рекомендацій. КФ аналізує взаємозв'язки між користувачами та взаємозалежності між продуктами, щоб виявити нові асоціації предметів та користувача. Наприклад, деякі системи КФ ідентифікують пари предметів, які, як правило, оцінюються однаково або однодумці користувачів зі схожою історією рейтингу чи покупок для виведення невідомих взаємозв’язків між користувачами та предметами. Єдиною необхідною інформацією є минула поведінка користувачів, яка може бути їх попередніми транзакціями або способом оцінки товарів. Незважаючи на те, що КФ, як правило, є більш точним, ніж методи, що базуються на вмісті, КФ страждає від проблеми холодного старту через його нездатність звертатися до нових продуктів системи, для яких підходи на основі вмісту були б достатніми. Рекомендовані системи покладаються на різні типи вхідних даних.

Однак ці методи формування рекомендацій зазвичай враховують окремо дані споживача або характеристики товарів, історію активності тощо. Наприклад, колаборативна фільтрація страждає від проблеми «холодного 10 старту» для користувачів, та у випадку з неявним зворотнім зв’язком, не використовує негативний відгук. Тоді як контентна фільтрація не бере до уваги вподобання користувачів.

Для вирішеня цієї проблеми необхідно удосконалити метод колаборативної фільтрації аби брати до уваги негативний зворотній зв’язок, наприклад повернення товарів назад до магазину, а також комбінувати

колаборативну фільтрацію з контентною, щоб мати можливість рекомендувати користувачу товари у випадках, коли його вподобання невідомі.

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

- виконати аналіз необхідних вдосконалень в системі управління сервером та доменом;
- розробити програмний модуль формування системи серверів та доменів на базі системи керування вмістом TMD Hosting;
- розробити системи комунікацій з отримання порад по оптимізації та безпеки серверву;
- удосконалити програмний модуль встановлення плагінів на сервер;
- удосконалити програмний модуль отримання коментарів з усіх серверів, якими управляє користувач;
- удосконалити програмний модуль отримання статистики з відвідування серверву;
- удосконалити програмний модуль отримання статистики з системною інформацією;
- сформувати архітектурну модель базі flux управління групою серверів.

Виконання перелічених завдань у повному обсязі зможе продемонструвати можливості контентної фільтрації на базі рекомендаційної системи для оренди серверів та доменів TMD Hosting.

Розроблена система може бути впроваджена в організаціях, де вже використовуються сервери на базі системи керування вмістом TMD Hosting, або для адмінів, які обслуговують групу серверів та доменів.

Об’єктом дослідження є процес удосконалення рекомендаційної системи управління серверами та доменами на основі контентної фільтрації.

Предметом дослідження є методи контентної фільтрації для удосконалення рекомендаційної системи управління серверами та доменами.

Методи дослідження:

- аналізу для порівняння технологій та обгрунтованого вибору засобів створення та удосконалення програмних модулів;

- аналіз алгоритмів фільтрації у додатку;
- методи для здійснення комунікації між користувачем там додатком;
- методи веб-програмування для розробки сучасних односторінкових веб-додатків;
- методи розробки сучасних додатків на базі flux-архітектури для створення архітектурної моделі програмного додатку управління серверами;
- методи тестування для перевірки розроблених програмних модулів.

Наукова новизна одержаних результатів:

1. Подальшого розвитку дістав метод фільтрації та налаштування серверів або доменів на базі системи керування вмістом TMD Hosting, який за рахунок управління низкою створених за різними технологіями серверів та їх деталізації, дозволяє користувачеві підвищити продуктивність управління серверами, адмініструвати декілька серверів або доменів 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 серверів, їх деталізації, формування відповідних порад з безпеки та оптимізації серверу, що дозволяє користувачу отримати адаптовані дані для більш якісного управління серверами та доменами.

Практична цінність отриманих результатів. Практична цінність результатів полягає у програмному продукті, що дозволяє спростити процеси користування та управління серверами за допомогою контентної фільтрації на базі системи управління керування TMD Hosting, удосконалити рівень обслуговування групи серверів. Отриманим результатом роботи є удосконалений метод, який підвищує точність надання рекомендацій, частково вирішує проблему «холодного старту» для користувачів, використовуючи контентну фільтрацію.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У роботах, опублікованих у співавторстві, здобувачу належить удосконалення рекомендаційної системи для оренди серверів та доменів на основі контентної

фільтрації за допомогою системи керування TMD Hosting, алгоритми роботи системи та приклади, що ілюструють її роботу по створенню та управлінню сайтами.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Дослідження характеристик рекомендаційних систем та аналіз стану питання

Перш за все, необхідно дати визначення рекомендаційної системи. Рекомендаційні системи – це програми, головна мета яких полягає у формуванні рекомендацій різних продуктів або сервісів для користувачів на основі їх переваг.

З появленням інтернету кількість інформації, з якою люди щодня стикаються, значно зросла. Це означає, що люди повинні орієнтуватися серед надзвичайно великої кількості доступних альтернатив, коли хочуть щось знайти. Але з іншого боку виступають власники інтернет-магазинів і сервісів: вони зацікавлені в персональній рекламі і рекомендаціях кожному конкретному користувачеві, тому що такий підхід може значно збільшити прибуток компанії. Як результат, в останні роки інтерес до розробки і поліпшення існуючих рекомендаційних систем значно виріс.

В наші дні рекомендаційні системи вже досить поширені і мають велику кількість застосувань. В першу чергу, рекомендаційні системи використовуються в інтернет-комерції для того, щоб допомогти користувачам вибрати відповідні товари. Такі сервіси збирають інформацію про переваги користувачів і намагаються запропонувати їм корисні товари. На даний момент існує множина методів для формування рекомендацій, але всі вони мають свої переваги і недоліки. Саме тому дослідження в даній області актуальні.

Якість рекомендаційної системи залежить від багатьох властивостей вироблюваних нею рекомендацій, які зазвичай оцінюються з допомогою показників продуктивності, що відображають вузьке уявлення про поведінку системи. Найчастіше розробники оцінюють точність рекомендацій за допомогою метрик, які вимірюють здатність рекомендаційної системи точно представляти набір відомих переваг. Точність вимірюється шляхом 11 резервування частини набору даних рейтингів у вигляді набору відомих уподобань і використання інших рейтингів для навчання рекомендаційної системи і вироблення

рекомендацій. Відомі переваги можуть використовуватися деякими метриками, такими як точність або відгук, для обчислення оцінки, що представляє точність алгоритму рекомендації для цього набору даних.

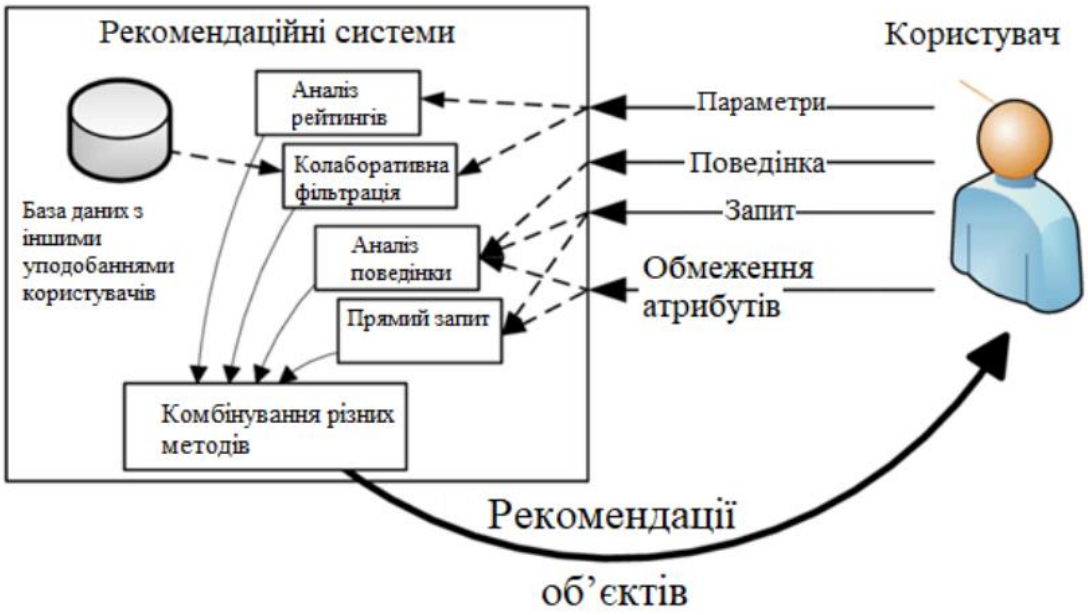

Рис.1.1 – Структура рекомендаційної системи

Однак метрики точності дають тільки грубе, одномірне уявлення про продуктивність рекомендаційної системи. Зокрема, точність і відгук обмежені декількома способами. По-перше, вони дають мало інформації про поведінку предметів з невідомими уподобаннями. По-друге, коли набір відомих вподобаних елементів малий, значення зазвичай низькі через малу ймовірність включення рекомендованих елементів в набір відомих переваг. Оскільки точність і відгук не передають ніякої інтуїції про поведінку предметів, які не входять в набір відомих уподобань, висновки про відносну перевагу однієї системи над іншою, можуть вводити в оману.

Рекомендації складаються індивідуально для кожного користувача на основі попередніх дій користувача або минулої активності на тому чи іншому веб-ресурсі. Крім того, важлива й поведінка попередніх учасників процесу.

Це важлива функція для інтернет-магазинів та один із небагатьох способів ефективної роботи великих каталогів, таких як Amazon. Рекомендаційний се в

даному випадку забезпечує користувачеві зручність навігації по веб-ресурсам, а не додаткові звичні можливості. З більш ніж 20 000 найменувань продуктів у вашому електронному каталозі, орієнтація вже здається непомірно складною. Що робити, якщо у вас мільйони продуктів?

Наскільки стомлива взаємодія з такими майданчиками для потенційних покупців відповідь зрозуміла. Також на допомогу приходять віджети пошуку товарів, які візуально схожі на товар, відносяться до тієї ж товарної групи або є безкоштовними (наприклад, якщо вам пропонують вибрати сумочку до взуття). Такі рішення не лише збільшують кількість переглядів, а й позитивно впливають на конверсії.

Такі прийоми, як свідчить практика, використовують не тільки інтернет-магазини. Подібні прийоми можна легко знайти практично скрізь: різні соціальні платформи, портали присвячені літературі, туристичні портали та ресурси новин, інтернет-магазини. Ця техніка справді популярна. Рекомендаційні сервіси збирають різну інформацію про користувачів, використовуючи кілька методів, загальних всім систем.

Отже, перший тип – це явний збір даних. Як можна здогадатися з назви, необхідні для роботи матеріали користувач готує сам. Наприклад, коли Google та інші системи рекомендаціних пошукових систем просять вас оцінити різні фактори, створити список обраного у певній області або відповісти на кілька запитань. Якщо особа відмовляється надати інформацію самостійно, актуальним буде спосіб далі:

Другий тип – неявний збір даних. Це шпигунська місія, де дії учасників процесу записуються програмно для подальшої обробки та застосування. Програма розпізнає покупки, оцінки на сайті, збирає інформацію про перегляди та коментарі. Звісно, вибір такого методу викликає етичні проблеми. Це пов'язано з тим, що захист персональних даних є однією з основних вимог користувачів, які здійснюють пошук на нашому сайті. Однак факт залишається фактом: можливе спостереження, і відвідувачі сайту не можуть бачити, чи справді така діяльність має місце.

Існують також види рекомендаційних систем, що визначаються за підходами, які вони застосовують.

Сьогодні сервер або домен потрібен кожній компанії, 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 сервери зростає щороку, але разом і з цим виникає проблема в їх пошуку та створенні, оскільки при створенні постає питання в використанні системи керування чи створення на базі сучасних фрейморків.

Створюючи сервер на TMD Hosting ми маємо такі переваги як:

- Швидкість створення. раніше створення серверу було надзвичайно тривалим і трудомістким процесом, повним проб і помилок. Сьогодні TMD Hosting позбавили користувачів і розробників від маси непотрібних операцій. Максимальний рівень автоматизації, максимальна швидкість створення нових серверів є головною перевагою таких систем.

- Поширеність TMD Hosting. Системи управління серверами міцно лідирують в серверо-будівництві - переважна більшість веб-серверів переходить на хостингові додатки, і тенденція ця зберігається. За останні роки близько 59% сайтів використовували хостингові майданчики.

- Простота розробки та підтримки. Завдяки ретельно продуманим функціоналом хостингового додатку, сьогодні не потрібно бути гуру програмування, щоб запустити найпростіший сервер. Зручний інтерфейс TMD Hosting і Google легко виведуть тямущого студента на правильну дорогу.

- Швидкий запуск серверу. Залежно від дизайну і рівня настройки, сучасні хостингові додатки 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 хостингових додатків йде значно менше часу.

- SEO-оптимізація. Хостингові додатки 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, сервери на базі хостингових додатків мають певні недоліки такі як:

- Низька безпека серверу. Поширеність хостингових додатків у всьому світі - це одночасно успіх і вразливість. Внутрішній устрій найбільш популярних систем вивчено хакерами уздовж і поперек, що робить ваш сервер потенційною жертвою чергового зловмисника, але це все може вирішити SSL сертифікат.

- Однотипність серверів. Зробити сервер у хостинговому додатку 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. TMD Hosting 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 на якому створюються сервери є TMD Hosting, адже за допомогою такої системи створюються різноманітні сервери та домени. Але така система має ряд недоліків, серед яких є незручність використання системи при обслуговуванні групи серверів, має повільне завантаження та неефективну систему комунікацій допомоги користувачу. Саме тому, необхідно створити додаткові програмні модулі для формування удосконаленого додатку, а саме фільтрації, зв’язку з користувачем, та безпеки серверів.

1.2 Аналіз процесу роботи рекомендаційних систем

Дані про придбані сервери та домени збираються з різних причин, не тільки для машинного навчання. Кожна навчальна система має специфічні вимоги до того, як дані повинні бути представлені для аналізу. Вибір конкретних даних для вивчення може сильно вплинути на модель навчання, з цих причин підготовка даних є важливою частиною для будь-яких цілей машинного навчання. Підготовка даних часто є найбільш трудомісткою частиною будь-якого проекту машинного навчання.

Вибір того чи іншого виду фільтрації чи комбінації методів безпосередньо залежить від двох факторів: складності проекту та обсягу його фінансування. Наприклад, створення алгоритму для системи тематичних блогів - відносно просте і помірно затратне завдання. Великий різнорідний проект, такий як інтернет-магазин, може бути витратним, особливо, якщо стоїть завдання збільшити конверсію на дуже значущу величину. У принципі, такі проекти не можуть обмежуватися одним типом рекомендаційного алгоритму і повинні використовувати гібридну фільтрацію. Результатом є збільшення вартості та складності розробки на порядок. Так чи інакше, при розробці проекту, що надає користувачеві можливість вибору конкретного об'єкта із загального набору, необхідно враховувати стрімкий розвиток юзабіліті абсолютно у всіх сферах життя людини – до 14 щоденних продуктів, заснованих на поточних потребах користувача, від оптимізації сну за допомогою пристрою що аналізує всі процеси що відбуваються під час сну і видає рекомендації щодо поліпшення

автоматичного підбору.

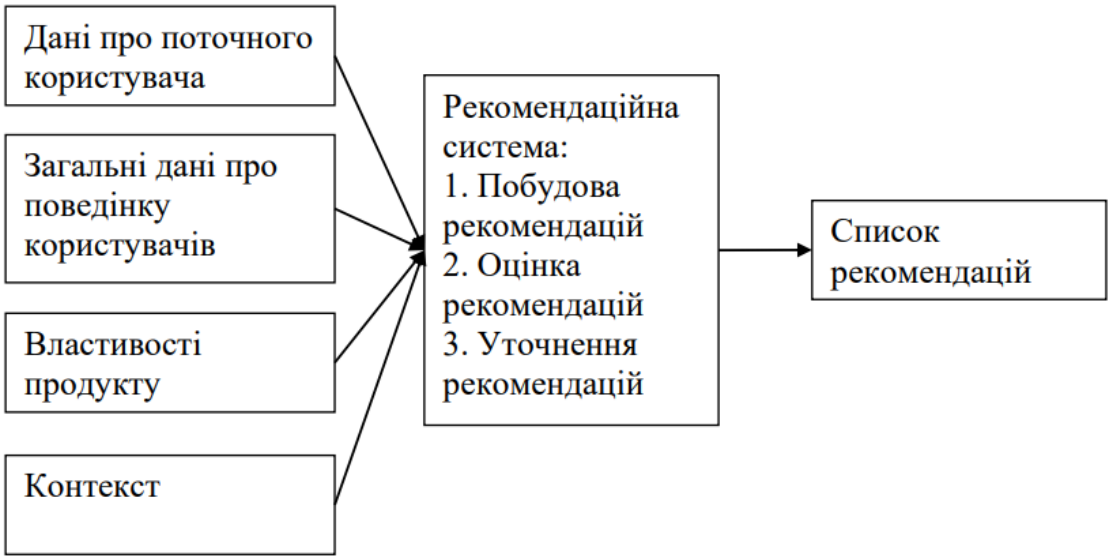

Рис 1.2 – Процес роботи рекомендаційних систем

На сьогоднішній день для створення рекомендаційних систем використовуються дві основні стратегії: контентна фільтрація та колаборативна фільтрація. Зазначимо, що на практиці зазвичай використовується гібридна методика, що поєднує в собі переваги наступних підходів:

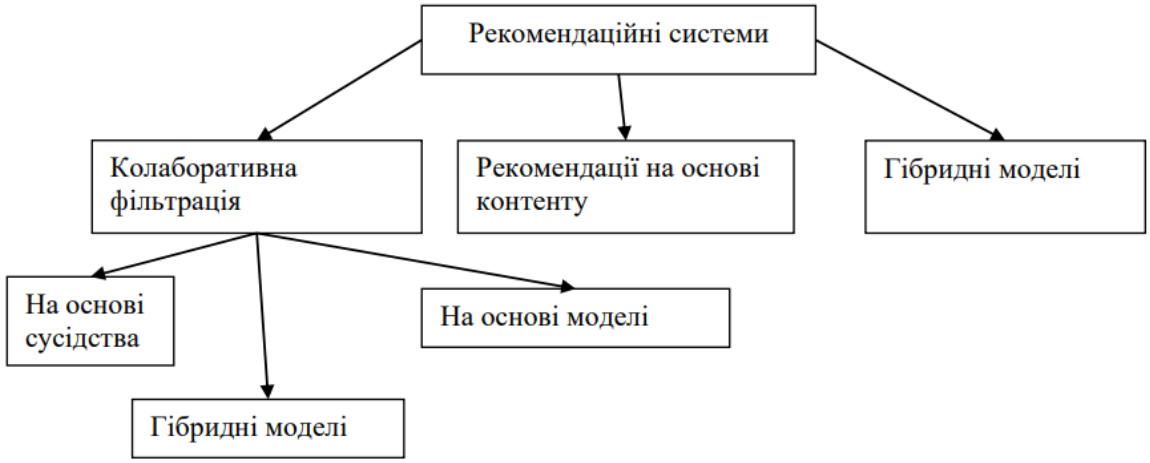

Рис.1.3 – Класифікація рекомендаційних систем

Контентна фільтрація (Рис. 1.4) ґрунтується на створенні профілю користувача і профілю об'єкта. Необхідно враховувати параметри об'єктів і їх відповідність перевагам користувача. Для цього в рекомендаційних системах

використовуються теги (ключові слова), щоб описати об'єкти, а профіль користувача відображав оцінку певних тегів або їх сукупність.

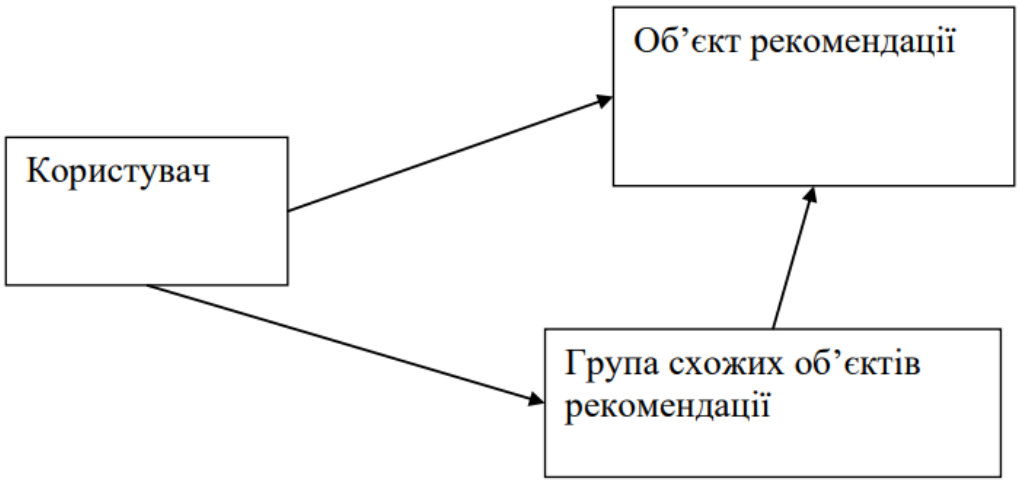

Рис. 1.4 – Контента фільтрація

У рекомендаційних системах з фільтрацією контенту функція задоволеності h(u, s) користувача u деяким об'єктом s визначається, ґрунтуючись на відомостях про задоволеність користувача об'єктами si ∈ S, які мають схожість з s. Тобто, якщо рекомендаційну систему використовувати в веб-додатку по оренді доменів та серверів, то для того щоб рекомендувати користувачеві u домени система повинна визначити, що пов'язує різні домени, яким користувач раніше дав високу оцінку. Потім система буде рекомендувати користувачеві домени з максимальною відповідністю тим, які були високо оцінені користувачем раніше. Призначений для користувача профіль в системі формується із споживаного користувачем контенту у вигляді множини параметрів, які визначають об'єкт s. Такими параметрами зазвичай є ключові слова і відповідні їм ваги для кожного об'єкта. Отже, необхідно визначити ваги цих параметрів. В інформаційному пошуку Користувач Об’єкт рекомендації Група схожих об’єктів рекомендації 16 одним з найбільш відомих способів визначення ваги ключових слів є TF-IDF міра. Припустимо, що N – підсумкова кількість об'єктів, які мають ймовірність бути рекомендованими користувачеві, а також, що ключове слово kj зустрічається в ni об'єктах, а fi, j - кількість

входжень цього слова в об'єкт dj. Тоді TFi, j (term frequency) – це відношення числа входжень тега до загальної кількості тегів об'єкта, тобто:

$$TF_{i.j} = \frac{f_{i,j}}{max_z f_{z,j}}.$$

Але якщо враховувати тільки частоту входження тега, то в більшості об'єктів максимальна вага буде у найпоширеніших тегів, що, найімовірніше, призведе до неправильного оцінювання переваг користувача. Для уникнення подібного застосовується IDFi (inverse document frequency) – величина, зворотна частоті входження тега в об'єкт колекції. Визначаємо її як:

$$IDF_i = log \frac{N}{n_i}.$$

Таким чином, вага wi, j у ключового слова ki в об'єкті dj позначається як добуток частоти входження тега на зворотну частоту об'єкта:

$$w_{i.j} = TF_{i,j} \times IDF_i,$$

В такому випадку контент об'єкту dj можна визначити так:

$$Content(d_{j,i}) = (w_{1,j}, \dots, w_{k,j}).$$

Як зазначалося вище, рекомендаційні системи з контентною фільтрацією пропонують об'єкти з урахуванням тих, які користувач високо оцінив раніше. Різні об'єкти порівнюються з тими, які сподобалися користувачеві і з них рекомендуються ті, які мають максимальну схожість. Сукупність об'єктів, які користувач оцінив раніше, утворює призначений для користувача профіль ContentBasedProfile(u) або, інакше кажучи, вектор ваг (wu,1, ..., wu,k), де кожна вага wu,i визначає важливість тега ki для користувача u. Отже, ContentBasedProfile (u) і Content (s) можна уявити як TF-IDF вектори $\vec{w}\,u$ і $\vec{w}\,s$, при цьому функція задоволеності користувача h(u, s) може бути представлена 17 як косінусний коефіцієнт векторів $\vec{w}\,u$ і $\vec{w}\,s$, де K – спільна кількість тегів в системі:

$$h(u,s) = \cos(\vec{w}_u, \vec{w}_s) = \frac{\vec{w}_u \vec{w}_s}{\|\vec{w}_u\| \|\vec{w}_s\|} = \frac{\sum_{i=1}^{K} w_{i,u} w_{i,s}}{\sqrt{\sum_{i=1}^{K} w^2_{i,u}} \sqrt{\sum_{i=1}^{K} w^2_{i,s}}},$$

На додаток до евристиків, які в основному ґрунтуються на методах пошуку інформації, часто використовуються інші методи рекомендацій, які використовують фільтрацію контенту. Наприклад, різні методи машинного навчання, такі як прості баєсівські класифікатори, нейронні мережі, дерева рішень та кластеризація. Однак, на відміну від методів пошуку інформації, ці методи намагаються передбачити задоволеність користувачів, використовуючи як основу евристику косинусного коефіцієнта. Наприклад, у додатку по оренді доменів та серверів можна, використовуючи інформацію про домени які сподобались та ні, через наївний байесовский класифікатор виділити домени, які користувач ще не бачив і оцінити ймовірність належності домена pj до певного класу Ci (вподобані та ні).

Контента фільтрація (Рис. 1.5) використовує відомі переваги групи користувачів, до якої входить той користувач, для якого потрібно спрогнозувати рекомендації. Головна підстава такого підходу в тому, що користувачі однаково оцінили деякі об'єкти в минулому, найімовірніше, однаково оцінять інші об'єкти в майбутньому.

Рис 1.5 – Колаборативна фільтрація

На відміну від контентної фільтрації, методи колаборативної фільтрації намагаються передбачити чи буде відповідати перевагам користувача той чи інший об'єкт, грунтуючись на оцінках інших користувачів однієї групи, тобто

отримуємо оцінку задоволеності h (u, s) деяким об'єктом s для користувача u, яка обчислюється залежно від задоволеності h (uj, s) тим же об'єктом s користувачів uj є U, які мають схожі ознаки з користувачем u. Варто відзначити важливу особливість даного методу: ознаки схожості одного користувача на інших не завжди є частиною даної системи. Тобто, в системі оренди доменів та серверів призначені для користувача групи можуть визначатися не тільки на підставі уподобань користувачів, але і грунтуючись на більш загальних ознаках, таких як географічне положення, профілі в соціальних мережах та інших параметрах. Подібного роду інформація може служити для первинної оцінки схожості нового користувача на вже зареєстрованих в системі і може застосовуватися для вирішення поширеною для контентної фільтрації проблеми «холодного старту».

Існує досить велика кількість user-based рекомендаційних систем, розроблених за участю як вчених, так і комерційних структур різних сфер діяльності. Системою такого роду є «Grundy system» - програм бібліотекар задає користувачам питання на різну тематику, складаючи таким чином призначений для користувача профіль і рекомендуючи книги, які могли б припасти до смаку користувачеві. Якщо користувач відповідав, що запропонована книга йому не цікава, то система починала з'ясовувати причину, поглиблюючи тим самим інформацію про профілі. Таким чином, «Grundy system» встановила для кожного користувача персональну модель, завдяки якій могла вельми точно радити цікаву літературу. В той самий час інша система, звана «Tapestry», замість визначення схожості через питання, пропонувала користувачеві самому перерахувати користувачів з відповідним вибором переваг.

Також, існує думка, що методи колаборативної фільтрації вмісту діляться на дві основні категорії: засновані на пам'яті і засновані на моделях. 19 Перша категорія алгоритмів намагається передбачити рейтинг об'єкта, використовуючи знання про всі об'єкти, які користувач встиг оцінити раніше. Величина рейтингу ru,s для користувача u і об'єкта s обчислюється, як агреговане значення оцінок інших найбільш схожих N користувачів для цього самого об'єкта s:

$$r_{u,s} = \text{aggr}_{u \epsilon \hat{U}} r_{u',s},$$

де U позначає множина N користувачів, які оцінювали об’єкт s і найбільш схожі на користувача u. В якості найпростішого прикладу функції агрегування можна привести обчислення середнього:

$$r_{u,s} = \frac{1}{N} \sum_{u \epsilon \hat{U}} r_{u',s}.$$

Найпоширенішим випадком агрегування є:

$$r_{u,s} = k \sum_{u \epsilon \hat{U}} \text{sim}(u, u') \times r_{u',s},$$

$$r_{u,s} = \bar{r}_u + k \sum_{u \epsilon \hat{U}} \text{sim}(u, u') \times (r_{u',s} - \bar{r}_{u'}),$$

де k застосовується для нормалізації і, як правило, a sim (u, u') – міра «схожості», яка є зворотною відстані і в більшості випадків береться в якості ваги. Таким чином, чим більше користувачі u і u' схожі, тим більший буде врахована вага при обчисленні ru,s. Ми можемо бачити, що в різних рекомендаційних системах, цілком можливо, можуть бути застосовані різні заходи схожості. Давайте розберемо дві з них, які зустрічаються найбільш часто. У випадку із застосуванням зважених сум є кілька складнощів, наприклад, нездатність враховувати, що різні користувачі не завжди оцінюють об'єкти з однаковим рівнем строгості.

Існує величезна кількість модифікацій, які вдосконалюють ефективність роботи даних методів. Наприклад, зумовлені рейтинги (default voting) представляють одну з таких модифікацій згаданого вище методу, який заснований на пам'яті.

В той же час, коли розібрані вище підходи застосовувалися для обчислення схожості користувачів, B. Sarwar застосували ці ж самі підходи для 20 визначення кореляції між об'єктами, щоб обчислювати оцінки для них. Пізніше ця думка була вдосконалена до top-N алгоритму. Також, існує емпіричне підтвердження того факту, що алгоритми з фільтрацією вмісту здатні працювати

з більшою продуктивністю щодо витрачених обчислювальних ресурсів, при цьому забезпечуючи не менше точні рекомендації, ніж більшість алгоритмів з колаборативною фільтрацією.

Деякі рекомендаційні системи використовують комбінацію підходів фільтрації контенту та колаборативної фільтрації, які називаються гібридними методами. Вони дозволяють дещо обійти недоліки обох підходів. Є кілька основних варіантів комбінування різних методів у гібридній рекомендаційній системі., вони указані на рисунку 1.6.

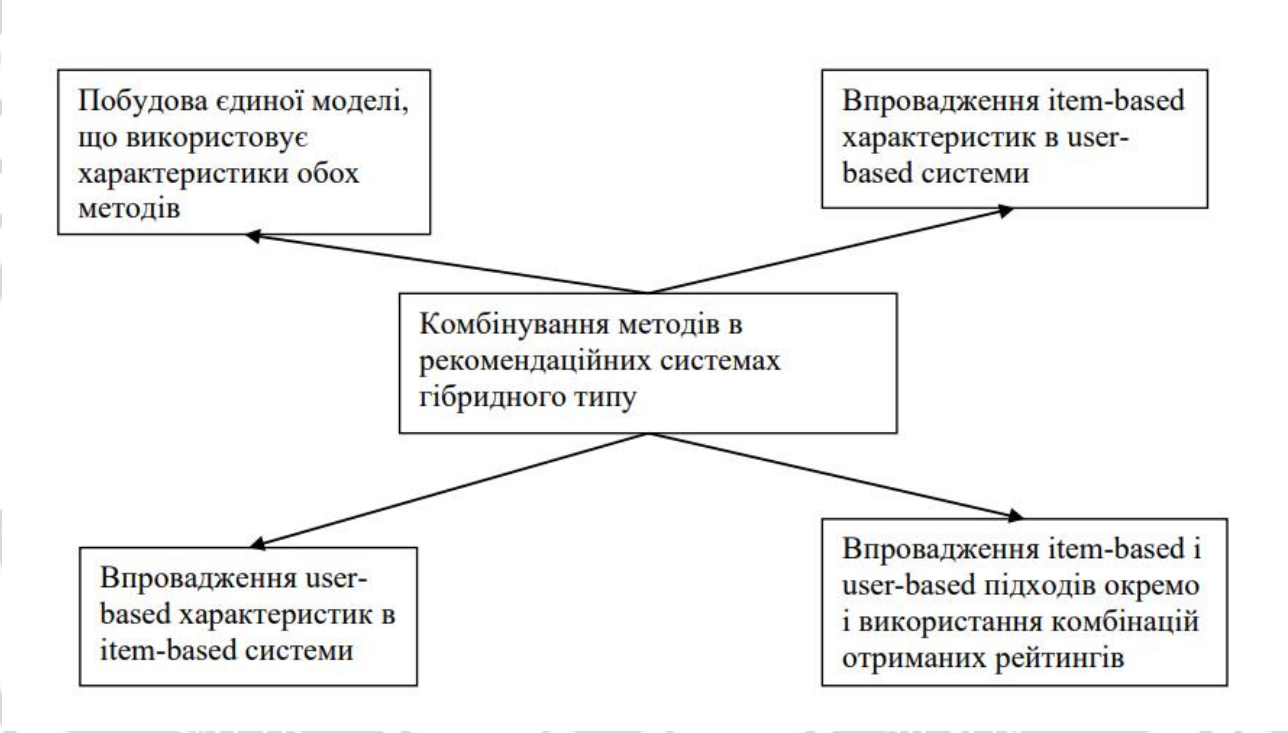

Рис. 1.6 – Комбінування методів в рекомендаційних системах гібридного типу

Останнім часом цей варіант став дуже популярним серед багатьох дослідників. Основний принцип цього методу полягає у використанні властивостей контентної фільтрації та колаборативної фільтрації (наприклад, жанр музичного твору та рід занять користувача) в одній рекомендаційній системі. Об'єднання методів у гібридній рекомендаційній системі для побудови єдиної моделі, що використовує межі обох методів. Впровадження item-based характеристик в userbased системи Впровадження userbased характеристик в item-based системи. Впровадження item-based і user-based підходів окремо і використання комбінацій отриманих рейтингів 21 До таких підходів належить уніфікований імовірнісний метод, який застосовує латентно-семантичний аналіз.

Такі гібридні рекомендаційні системи побудовані на колаборативній складовій, але також включають в призначений для користувача профіль деякі дані фільтрації вмісту. Потім дані є основою розрахунку подібності користувачів, а не суми оцінених об'єктів. Цей підхід дозволяє уникнути проблеми нестачі даних, викликаної невеликою кількістю пар користувачів при досить великій кількості об'єктів, що оцінюються. Цей метод також дозволяє нам рекомендувати користувачам не тільки об'єкти, що мають позитивні відгуки інших користувачів, але й об'єкти, які можуть сподобатися їм, виходячи з їх особистих смаків.

Одним із найвідоміших методів гібридних рекомендаційних систем цієї групи є зменшення розмірності профілю за допомогою фільтрації контенту. Один із таких методів використовує латентний-семантичний аналіз (LSA) для створення спільного контенту для профілів користувачів, представлених набором векторів. Такий підхід значно покращує продуктивність у порівнянні з системами, що використовують тільки фільтрацію вмісту.

Однією з комбінацій гібридних рекомендаційних систем є роздільна реалізація підходів на основі елементів та на основі користувачів. Крім того, є два способи розгорнути події. У першому випадку ми можемо поєднати рейтинги з двох систем, використовуючи лінійну комбінацію рейтингів. Другий спосіб пропонує використовувати систему, що підходить для конкретного випадку. У нас є система DailyLearner, в якій застосовується система рекомендацій, що дає рекомендації з мінімальним рівнем помилок. Гібридні рекомендаційні системи іноді доповнюються методами, які використовують бази знань для підвищення якості рекомендацій та вирішення основної проблеми більшості рекомендаційних систем (наприклад, холодного запуску). Існують роботи різних авторів, що демонструють перевагу гібридних підходів над рекомендаційними системами, заснованими виключно на предметах та користувачах, порівняно з продуктивністю.

1.3 Аналіз критеріїв та методів оцінювання рекомендаційних систем

Існує множина різних критеріїв за якими можна оцінювати рекомендаційну систему – такі як точність, новизна, можливість дивувати, стійкість до атак, залежність від холодного старту, переконливість і так далі, але одним з найбільш важливих все-таки є точність. Вона показує на скільки наші передбачення близькі до еталонного результату. Для вимірювання точності є множина методів, рекомендую підійти до вибору акуратно, так як від цього багато залежить. Один з найпопулярніших методів – розрахунок середньоквадратичної помилки (RMSE). Після того як на основі тестових даних алгоритм виробляє передбачення, помилку можна обчислити за такою формулою:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (p_{ui} - r_{ui})^2}$$

U – користувач;

i – предмет;

r – оцінка;

p – прогнозована оцінка;

T – загальна кількість тестових оцінок, чим менше тим краще;

Дослідники вивчили властивості рекомендаційних систем, які можуть бути корисні розробникам за межами точності рекомендацій, а також метрики для їх захоплення. Кілька таких властивостей, часто з відповідними метриками, були ідентифіковані в літературі рекомендаційних систем, таких як охоплення, різноманітність, новизна і адаптивність. Наприклад, різноманітні рекомендації, в яких елементи відрізняються один від одного, можуть забезпечити більшу цінність для користувача. Якщо відомо, що користувачеві подобаються науково-фантастичні фільми, рекомендація всіх фільмів Star Wars може бути точною, але вона не різноманітна і може бути незадовільною для користувача.

Друге завдання, на якому ми зосереджені – це оцінка стабільності. Стабільність рекомендаційної системи вимірює узгодженість рекомендацій

після внесення змін в набір даних. Розглянемо рекомендаційну систему, яка 23 пропонує фільм Interstellar в якості найбільш рейтингового елемента для 1% користувачів. Якщо якийсь користувач додає новий рейтинг для Interstellar і в результаті він більше не рекомендується нікому, то рекомендаційна система може вважатися нестабільною і негативно впливати на довіру користувачів.

Ми визначаємо загальну модель рекомендаційних систем, яку потім використовуємо для формального визначення та систематичного дослідження властивостей простору рекомендаційних систем.

Для поліпшення продуктивності веб-сайтів може бути використано кешування. Воно дозволяє не генерувати часто запитувані сторінки кожен раз заново, а використовувати збережені версії. Це дає можливість значно скоротити час завантаження сторінки і навантаження на сервер. Наприклад, система управління базою даних може зберігати результат часто виконуваних запитів, окремі фрагменти інформації можуть завантажуватися в оперативну пам'ять сервера, HTML-код сторінок або їх невеликих блоків може бути витягнутий з файлової системи веб-сервера, сторінки цілком можуть бути закешовану інтернет-провайдером, і т. д.

Для того щоб кешування було ефективним, інформація повинна використовуватися багаторазово і часто. Але персоналізовані сторінки можуть містити дані, що відносяться до конкретного користувача. Веб-сайт ідентифікує відвідувача по cookie-файлах і генерує сторінки спеціально для нього. Такі сторінки не можуть бути закешовані цілком. Навіть якщо сторінка буде закешована, ймовірність того, що вона буде повторно завантажена з кешу, як правило, досить низька, тому в результаті значно виросте обсяг використовуваної пам'яті на жорсткому диску сервера, в той час як навантаження на процесор знизиться незначно.

Розв’язанням даної проблеми може бути виділення всіх персоналізованих блоків в окремі URL-адреси і їх динамічне підвантаження за допомогою технології AJAX.

Крім проблеми, пов'язаної з неможливістю кешування сторінок, існують труднощі, викликані специфікою вихідних даних. Кожному користувачеві

відповідають пари «документ-оцінка», де документом може бути сторінка сайту 24 або певна тема (тег), а оцінкою – деяка міра інтересу користувача до неї, наприклад, число переглядів.

Можна уявити собі ці дані як матрицю, кожен рядок якої відповідає користувачеві, а стовпець – документу. В такому випадку завдання зводиться до обчислення невідомих елементів матриці. Матриця рейтингів, як правило, дуже розріджена - і користувачів, і сторінок багато, а рейтингів на практиці набагато менше, ніж їх твір, адже середній користувач відвідує небагато сторінок. Інші елементи матриці невідомі, і їх значення треба передбачити, тобто кількість оцінок, які необхідно передбачити, набагато перевищує кількість відомих оцінок. Важливо, щоб система вміла ефективно передбачити оцінки, виходячи з невеликої кількості прикладів. Також необхідна наявність критичної кількості користувачів. Подолати проблему розрідженості оцінок можна, якщо при пошуку схожих користувачів використовувати інформацію про користувача, що міститься в його профілі.

Колаборативна фільтрація по користувачах є одним з найбільш ефективних рекомендаційних алгоритмів, але має низку недоліків. Серед них, в тому числі, неможливість генерувати рекомендації для нових користувачів і погана масштабованість.

Перша проблема може бути вирішена за допомогою використання колаборативної фільтрації по схожості сторінок. Для використання колаборативної фільтрації на високонагруженних сайтах з великою кількістю сторінок доцільно проводити розрахунки схожості користувачів паралельно в кілька потоків. Швидкість складання рекомендацій може бути підвищена шляхом зниження розмірності вихідних даних. Значно поліпшити продуктивність дозволить попереднє виділення груп користувачів зі схожою поведінкою. При цьому дистанцію між двома користувачами, які входять в різні групи, можна буде вважати нескінченною. Для цього може бути використано попереднє виділення груп схожих користувачів і виконання алгоритму колаборативної фільтрації всередині груп. Це дозволить вирішити проблему з розрідженістю вихідних даних і поліпшити масштабованість.

Для скорочення числа операцій по складанню рекомендацій можна навчати алгоритм не на всіх даних про користувачів, а на деякій релевантній вибірці. Виділення підмножин з вихідних даних може бути також використано для створення навчальної та перевірочної вибірок. В цьому випадку параметри алгоритму налаштовуються на основі даних з першої вибірки, а точність рекомендацій оцінюється на основі решти даних. Головним завданням при формуванні вибірки є пошук релевантної підмножини у вихідних даних. Найбільш простим способом є використання випадкової вибірки, де у всіх елементів вихідних даних рівна ймовірність попадання у вибірку. У деяких випадках виправдане використання стратометричгого відбору, коли генеральна сукупність ділиться на групи, що володіють певними характеристиками (стать, вік, політичні уподобання, освіта, рівень доходів тощо.), і відбираються елементи з відповідними характеристиками. Як правило, використовуються вибірки без повторень - один і той же елемент не може бути обраний двічі. Широко поширений підхід, при якому застосовується випадкова вибірка без повторів з виділенням тренувального і тестового набору даних в пропорції 80/20. Процес навчання алгоритму може повторюватися кілька разів, щоб уникнути помилок, пов'язаних з використанням невеликої підмножини замість всіх вихідних даних. Після створення навчальної та тестової вибірок налаштовується модель рекомендаційної системи, а потім оцінюється її точність. Далі створюється нова пара вибірок, і процес навчання і тестування повторюються k разів. В результаті використовується усереднене значення k моделей. Такий підхід називається крос-валідація. Вона може здійснюватися кількома методами. У випадку з повторюваними випадковими вибірками підмножини для навчання вибираються k раз поспіль випадковим чином. Іншим способом може бути виділення n підмножин вихідних даних. Одне з них використовується як тестове, а решта – як навчальні. Процес навчання і тестування повторюються n разів так, що кожна підмножина виявляється тестом один раз.

Відвідувачів і сторінки сайту можна класифікувати так, що вихідними даними буде інформація веб-перегляду, а результатом – клас відвідувачів, до 26 якого належить конкретний користувач. Існує множина способів класифікації,

серед яких можна виділити два типи – з навчанням і без нього. В алгоритмах класифікації з навчанням заздалегідь відомі класи і є дані про належність частини елементів до класів. Ґрунтуючись на цих даних, можна навчити алгоритм і класифікувати інші елементи. Якщо класи заздалегідь не відомі, то потрібно розділити всі елементи на групи так, щоб елементи, що входять в одну групу, були схожі між собою, але не схожі на елементи з інших груп.

Алгоритм k найближчих сусідів дозволяє розділити вихідні дані на непересічні класи, використовуючи інформацію про схожість окремих елементів. Користувач, клас якого ще ніхто не знає, буде віднесений до того класу, до якого відносяться k найближчих до нього інших користувачів. Одна з основних переваг цього методу полягає в тому, що при додаванні нових користувачів не потрібно перенавчати алгоритм на всіх даних – досить визначити найбільш відповідні класи тільки для нових відвідувачів. Але при класифікації доданих користувачів потрібно попарно порівняти їх вектори з даними про всіх вже наявних відвідувачів, що може зажадати великого обсягу пам'яті для обчислень і займе багато часу. Для високонавантажених веб-сайтів це може бути неприпустимо повільно.

Для виділення груп схожих користувачів може бути використаний Байєсівський класифікатор. Даний метод заснований на принципі максимізації апостеріорної ймовірності. Для користувача обчислюються функції правдоподібності кожного з класів і по ним обчислюються апостеріорні ймовірності класів. В результаті користувач відноситься до того класу, для якого апостеріорна ймовірність максимальна. Визначення класу може бути спрощено, якщо вважати, що всі параметри, що характеризують поведінку користувача і утворюють його вектор, незалежні один від одного.

Завдання функціонування рекомендаційної системи можна сформулювати як пошук N найцікавіших користувачеві посилань, а значить оцінити ефективність рекомендаційної системи, заснованої на класифікації, можна перевіривши, наскільки точно вона класифікує сторінки як цікаві для конкретного користувача. Таким чином, для подібних систем можна 27 застосовувати такі поняття, як точність і повнота. Для рекомендаційної системи

точність – це частка цікавих користувачеві сторінок серед запропонованих в рекомендаціях. Повнота – частка відомих системою як цікаві користувачеві сторінок серед всіх існуючих.

1.4 Постановка задачі дослідження.

Задача дослідження – аналіз критеріїв та властивостей оцінювання рекомендаційних систем на основі контентної та колаборативної фільтрації, дослідження рекомендаційної системи для оренди серверів та доменів на основі контентної фільтрації та колаборативної фільтрації**,** опис принципу роботи рекомендаційних систем і їх порівняльний аналіз, аналіз принципу оцінювання рекомендаційних систем, дослідження методу колаборативної та контентної фільтрації на основі подібності елементів, створення діючої системи оцінювання рекомендаційних систем, сортування негативних результатів в матриці вихідних даних, удосконалення контентної фільтрації у додатку по оренді доменів на серверів.

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 додатку весь необхідний код (HTML, JavaScript, CSS) завантажується разом із сторінкою або завантажується динамічно за необхідності (зазвичай у відповідь дії користувача). Сторінка не оновлюється та не перенаправляє користувача на іншу сторінку під час роботи. Взаємодія з односторінковими програмами часто включає динамічний зв'язок з веб-сервером. Термін «односторінковий додаток» був запроваджений Стівом Еном у 2005 році, але ця концепція обговорювалася у 2003 році. Крім того, Стюарт (stunix) Моріс описав "автономний веб-сайт" з тією ж метою та функціональністю з 2002 року. , Lucas Burdeau, Kevin Hackman, Michael Peachey та Evan Ye описані в патенті США 8,136,109. Javascript можна використовувати у веб-браузерах для відображення інтерфейсів користувача, виконання логіки додатків та взаємодії з веб-серверами. Розробникам Javascript потрібно писати менше коду, оскільки вони можуть створювати односторінкові програми з використанням власних бібліотек з відкритим вихідним кодом.

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. 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 у Google Creative Labs, він мав проблему йому потрібно було швидко створювати прототипи складних інтерфейсів і потрібен був інструмент, щоб уникнути написання HTML, що повторюється. React був у зародковому стані, а AngularJS та Backbone були надто громіздкими для прототипування, тому Еван створив свій власний фреймворк. З того часу Vue.js розвинувся, щоб дозволити створювати складні веб-програми, а також прототипи.

Вперше Vue було випущено у грудні 2014 року. Інформація про проект була розміщена у Hacker News, Echo JS та у підкатегорії javascript у день його початкового випуску. Одного разу проект з'явився на першій сторінці цих сайтів: Vue використовує синтаксис шаблону на основі HTML, який дозволяє вам декларативно пов'язати ваші DOM-рендеринги з основним екземпляром даних Vue. Усі шаблони Vue є дійсними HTML та можуть бути проаналізовані браузерами та парсерами HTML. Усередині Vue компілює шаблони у віртуальні функції рендерингу DOM. У поєднанні з реактивною структурою Vue може інтелектуально підраховувати кількість компонентів та повторно відображати їх, застосовуючи мінімальні маніпуляції з DOM у разі зміни стану програми.

Vue дозволяє використовувати синтаксис шаблону або писати свої функції рендерингу безпосередньо за допомогою JSX. Для цього просто замініть шаблон на функцію рендерингу. Можливості рендерингу відкривають можливості потужних шаблонів на основі компонентів. Наприклад, нова система трафіку буде повністю компонентною і використовуватиме функції рендерингу під капотом.

Однією з особливостей Vue є його ненав'язлива реактивна структура. Моделі – це просто плоскі об'єкти JavaScript. Це робить керування станом дуже простим та інтуїтивно зрозумілим. Vue забезпечує оптимізований повторний рендеринг без будь-яких спеціальних дій. Кожен компонент відстежує реактивні залежності під час рендерингу, тому система точно знає, коли його потрібно повторно рендерувати та які компоненти потрібно повторно рендерувати.

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

Сам Vue не включає маршрутизацію, але є пакет vue-router, який вирішує цю проблему. Він підтримує прив'язку вкладених шляхів до вкладених компонентів, що дозволяє точно контролювати переходи. Vue дозволяє створювати програми з використанням компонентів. Якщо ви хочете додати до цього vue-router, просто зв'яжіть свої компоненти з вашими маршрутами та дозвольте vue-router вирішити, де їх відображати.

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. 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:

- 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. 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 целомі. 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.

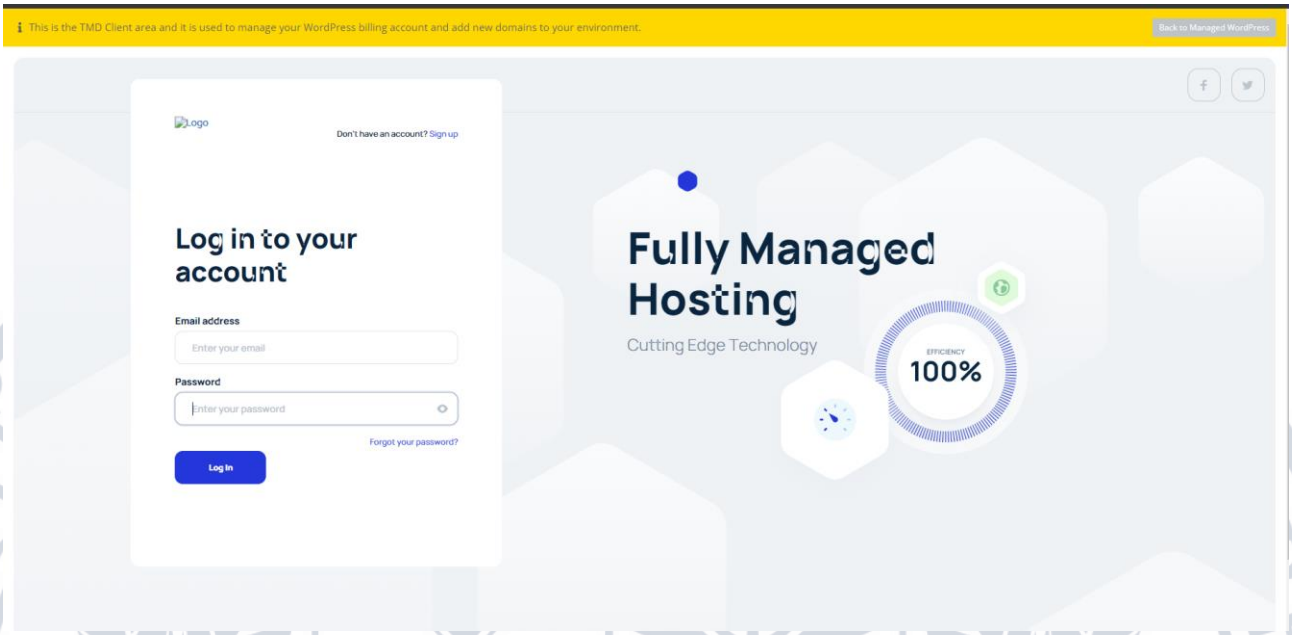

Рисунок 2.1 – Сторінка входу користувача

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 серверів, 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 серверву, тоді у на головній реалізується інтерфейс за допомогою якого він може це легко зробити.

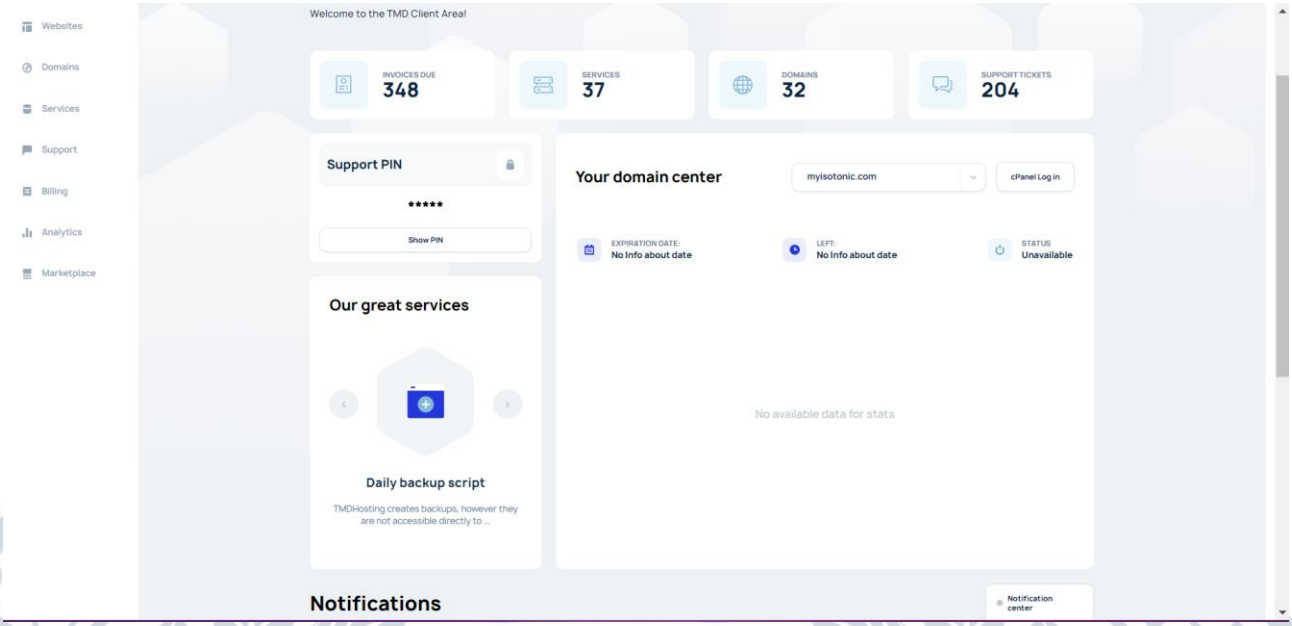

Рисунок 2.2 – Панель управління

Сторінка для управління системою фільтрації (Рисунок 2.2) – це сторінка на якій здійснюється алгоритм контентної фільтрації, користувач може вибрати необхідні фільтри після чого натиснути кнопку “Search” і йому будуть видані вільні домени для оренди (Рисунок 2.3).

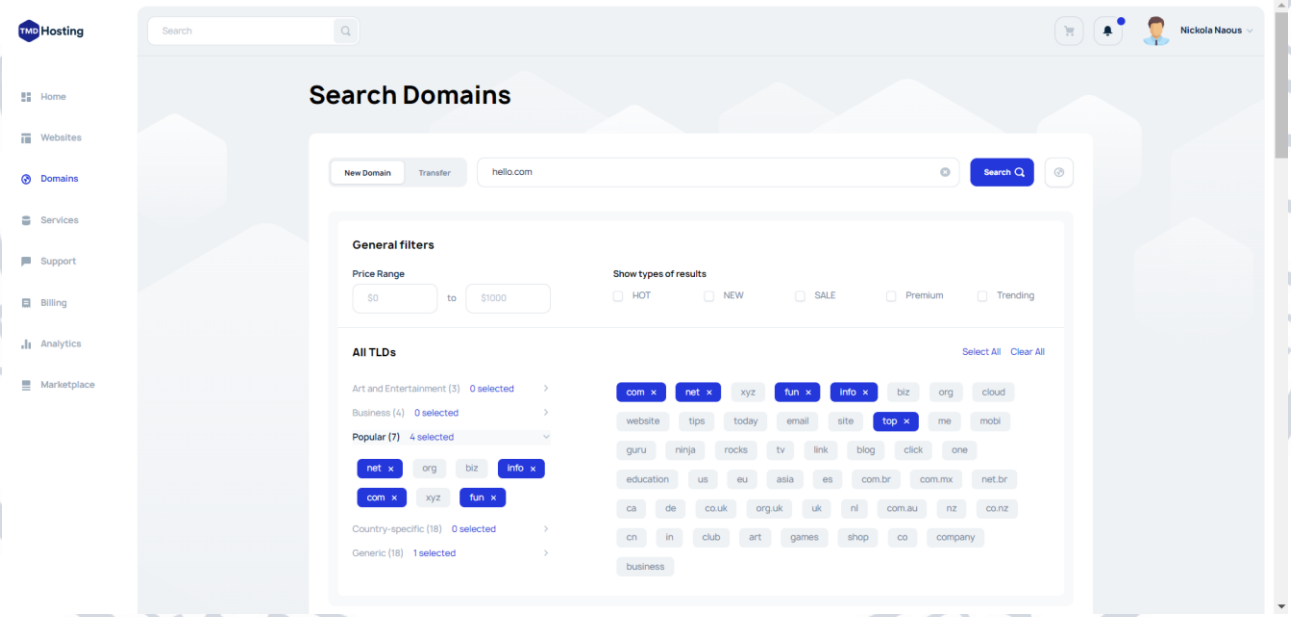

Рисунок 2.2 – Управління системою фільтрації

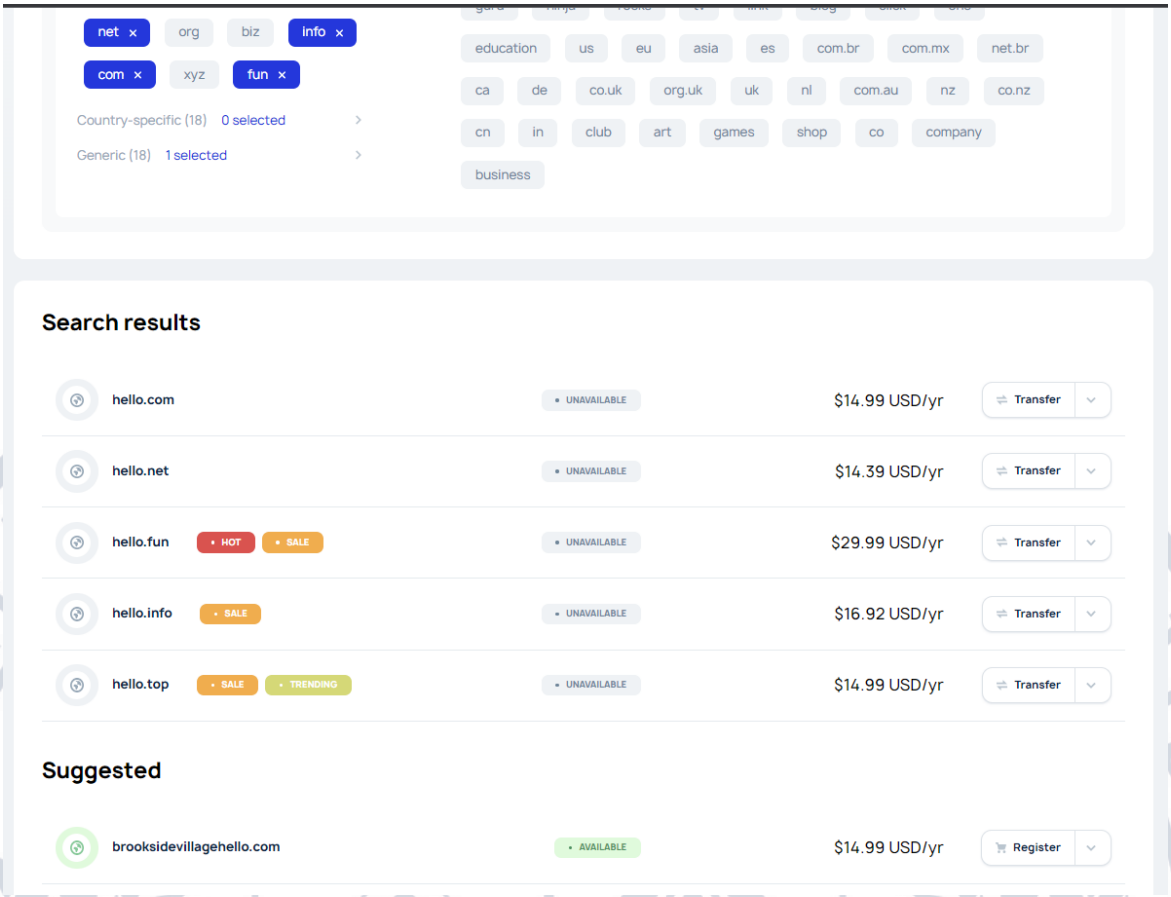

Рисунок 2.3 – Знайдені вільні домени за допомогою фільтрації

Таблиця із орендними доменами (Рисунок 2.4) допомогає користувачу зрозуміти які домени він орендує, а також при натисканні на кнопку Manage користувач буде направлений на сторінку із можливостями налаштування обраного домену, на цій сторінці він зможе налаштувати безпеку домену, його оплату, а також підв’язати під нього свої дані.

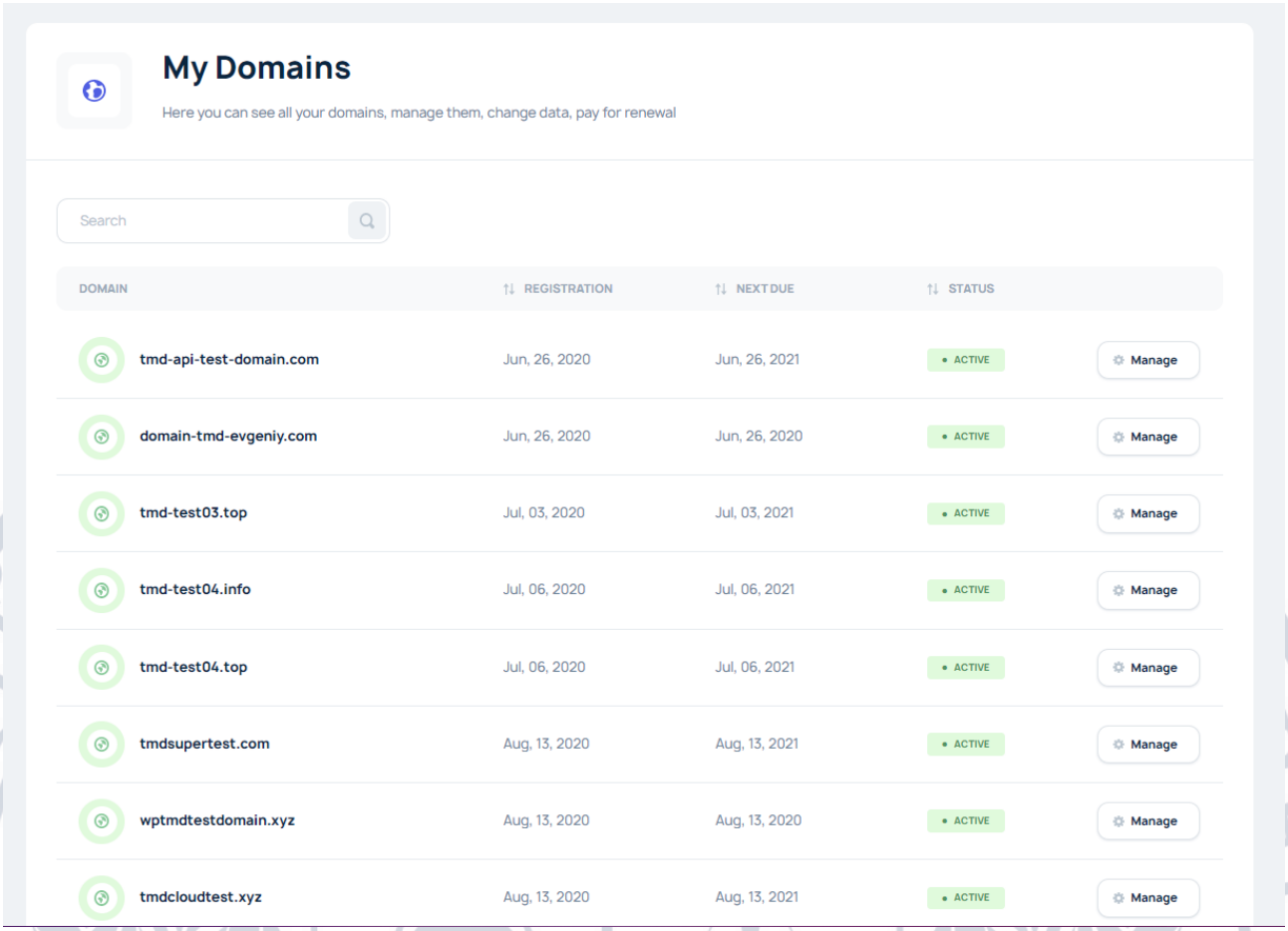

Рисунок 2.4 – Орендні домени

Сторінка Управління веб-сайтами створених чи перенесених із інших хостингів (Рисунок 2.4) передбачає використання окремих блоків, які містять інформацію про кожен веб-сайт і також при натисканні на копку Login, користувач переходить на сторінку управління визначеним сайтом, крім цього є можливість сортування сайтів.

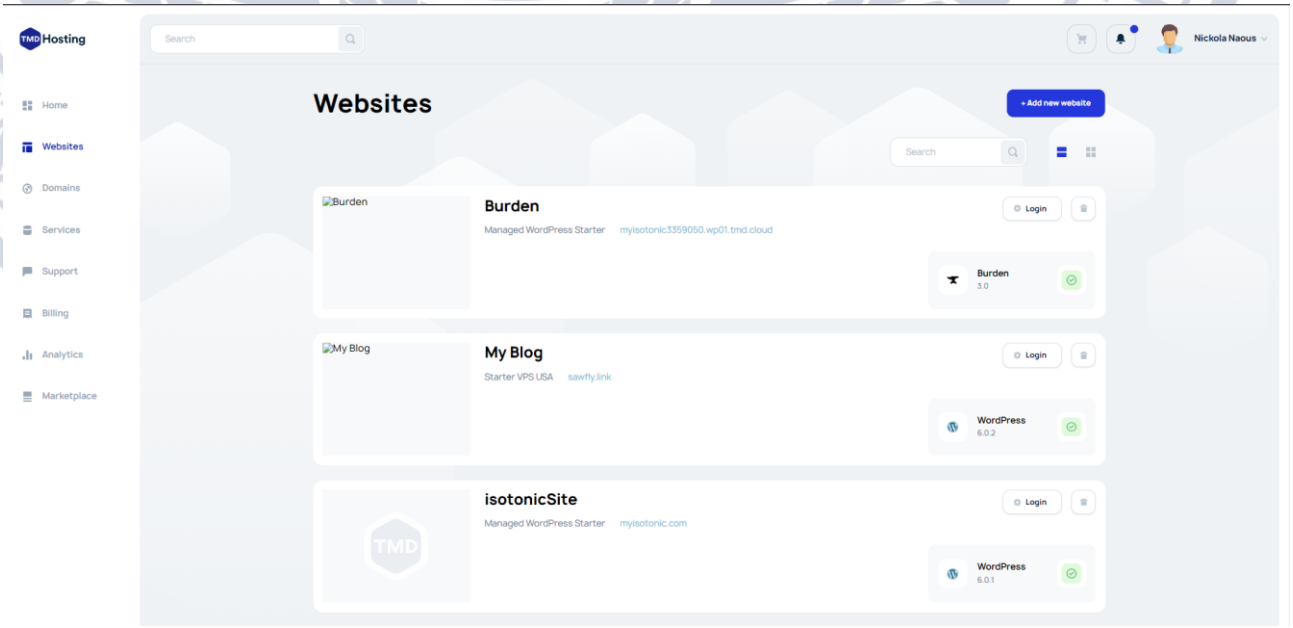

Рисунок 2.4 – Управління веб сайтами

Сторінка для обрання сторонніх додатків які користувач хоче встановити на свій сервер, тут працює пошук, користувач може обрати різноманітні додатки та за допомогою 5-и кроків їх встановити.

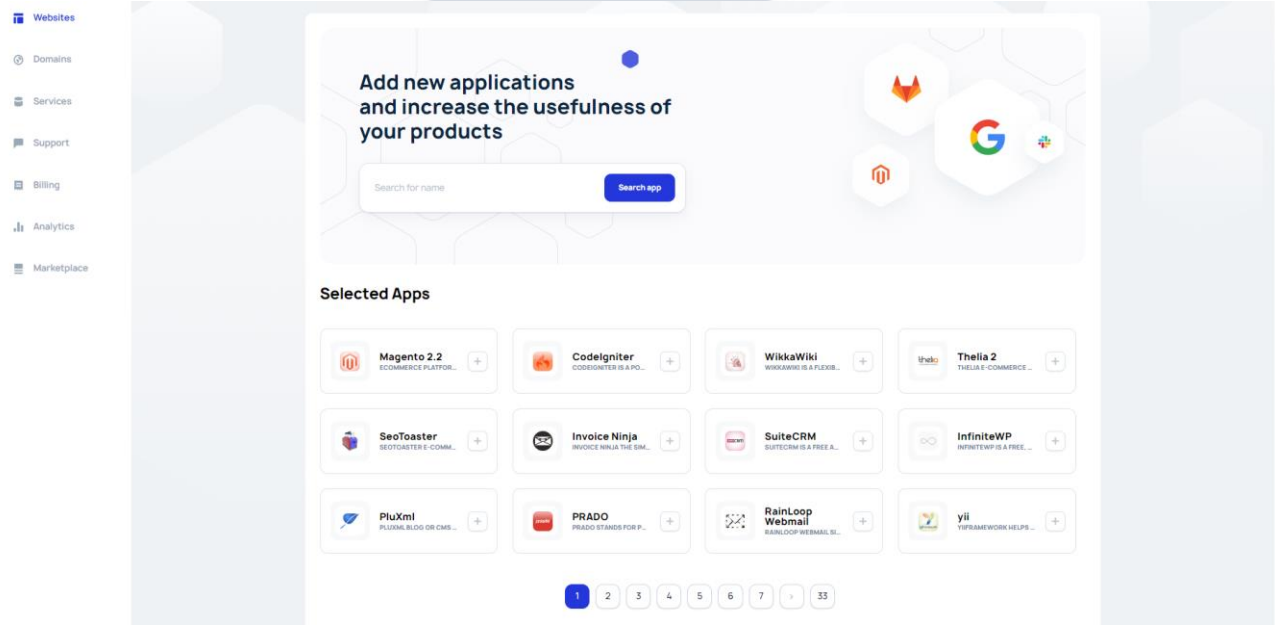

Рисунок 2.5 – Додатки для встановлення на сервер

Сторінка з чатом для підтримки користувачів тут користувач може задати питання на яке він отримає відповідь від служби підтримки. У цьому чаті є інформація про департамент у який звертається користувач, час звернення та час відповіді. Також є суппорт пін для того щоб верифікувати користувача. (рисунок 2.6)

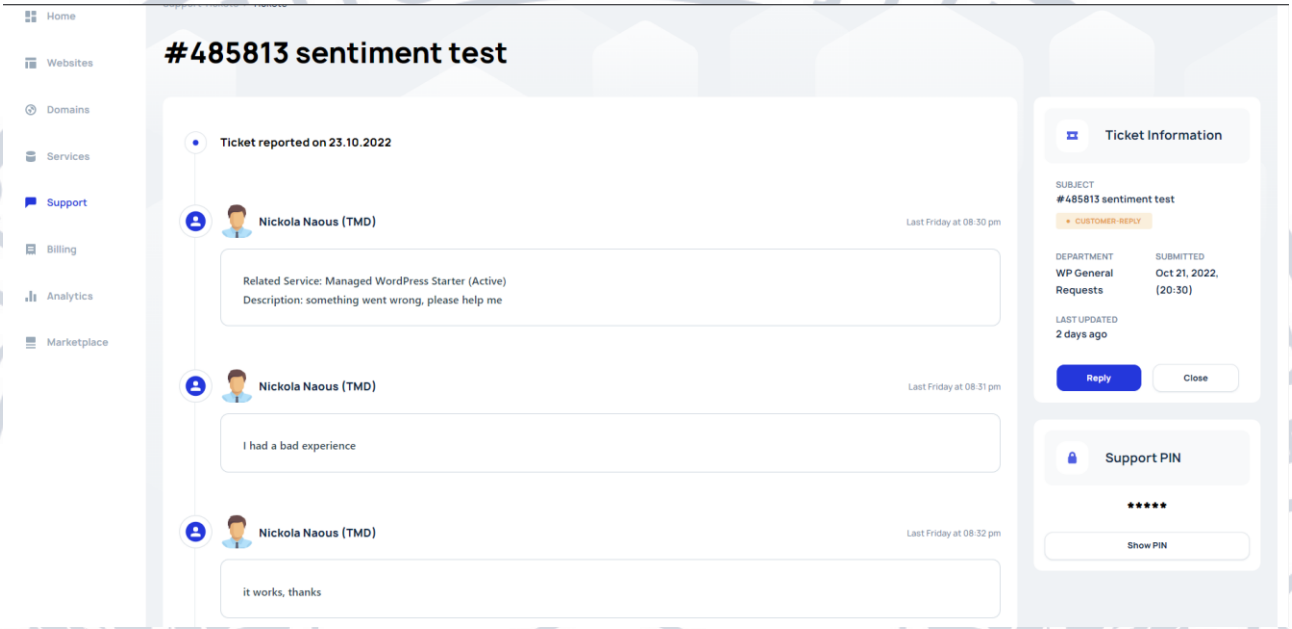

Рисунок 2.6 – чат для підтримки користувачів

Сторінка для перевірки даних (Рисунок 2.7, Рисунок 2.8) які приходять з серверу. Тут зображено 5-ть інформаційних блоків які показують навантаження на сервер. Також знизу є графік на якому можна відслідковувати по часу коли та яке навантаження було.

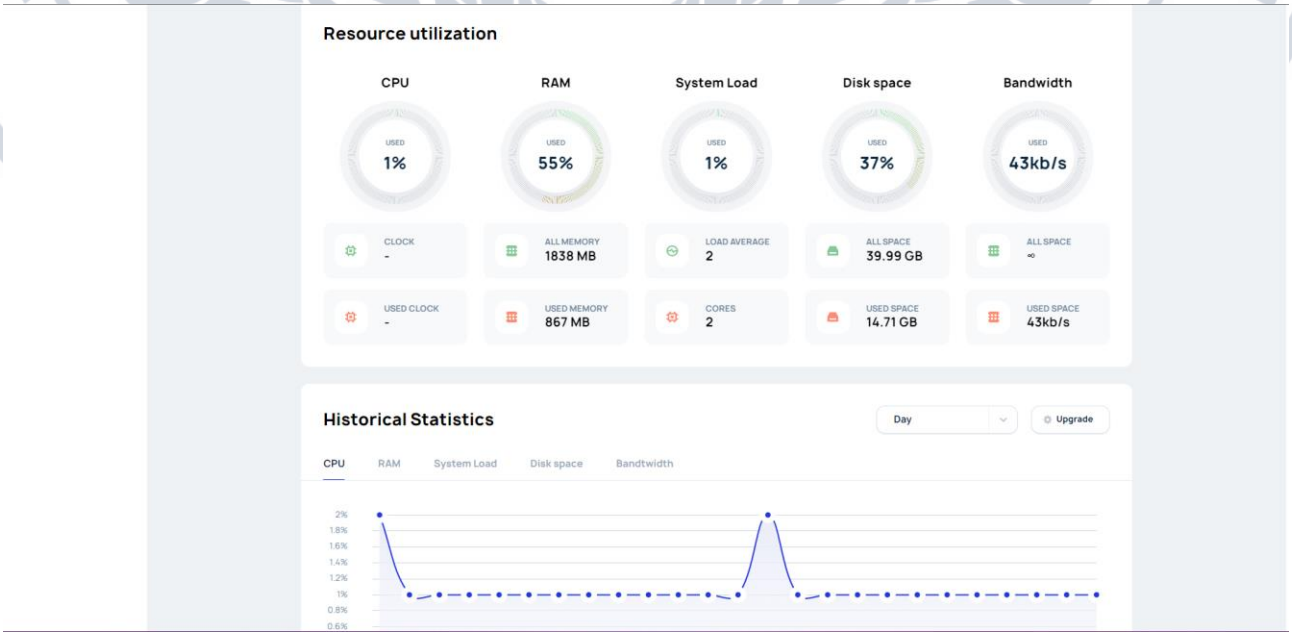

Рисунок 2.7 – Дані з сервера

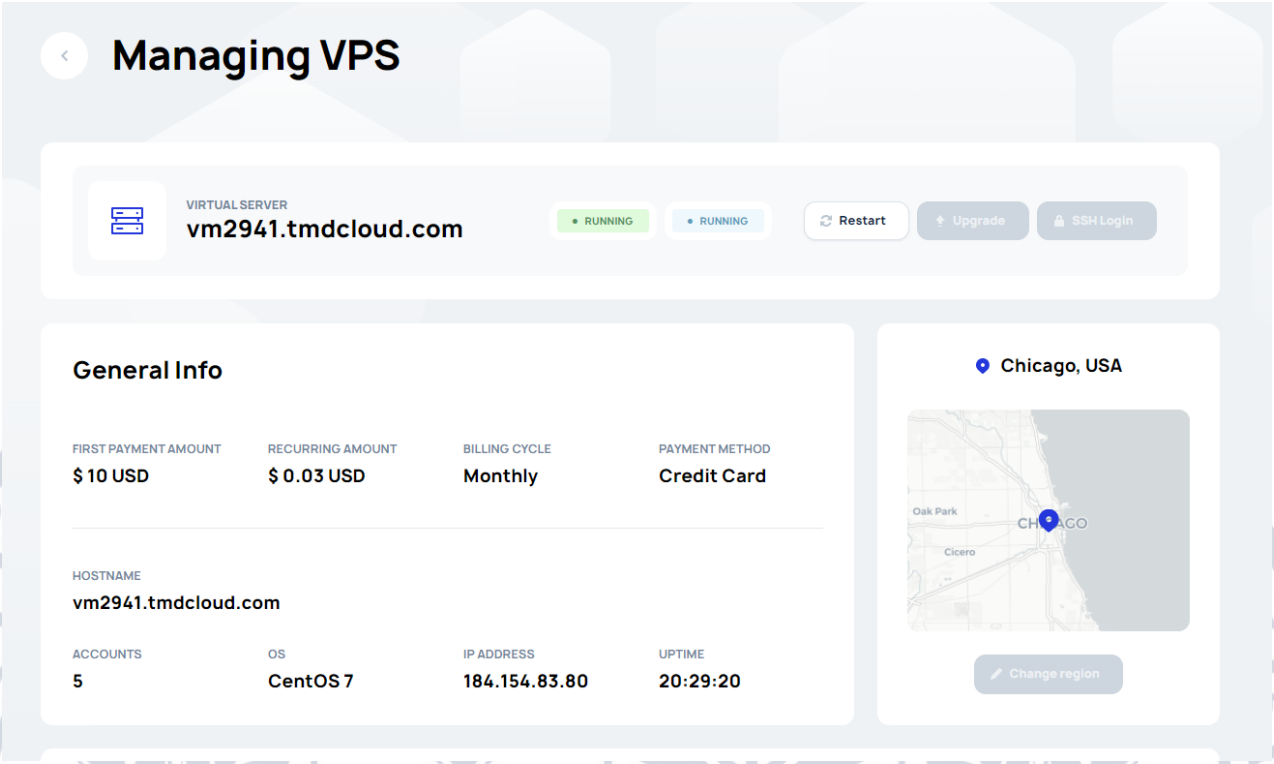

Рисунок 2.8 – Дані з сервера

Також користувач у додатку TMD Hosting може купити різноманітні послуги для свого серверу, це може бути або захист від DDOS атак, або купити свою базу даних для сайту чи серверу, тут реалізована контентна фільтрація при пошуку потрібної послуги. (Рисунок 2.8)

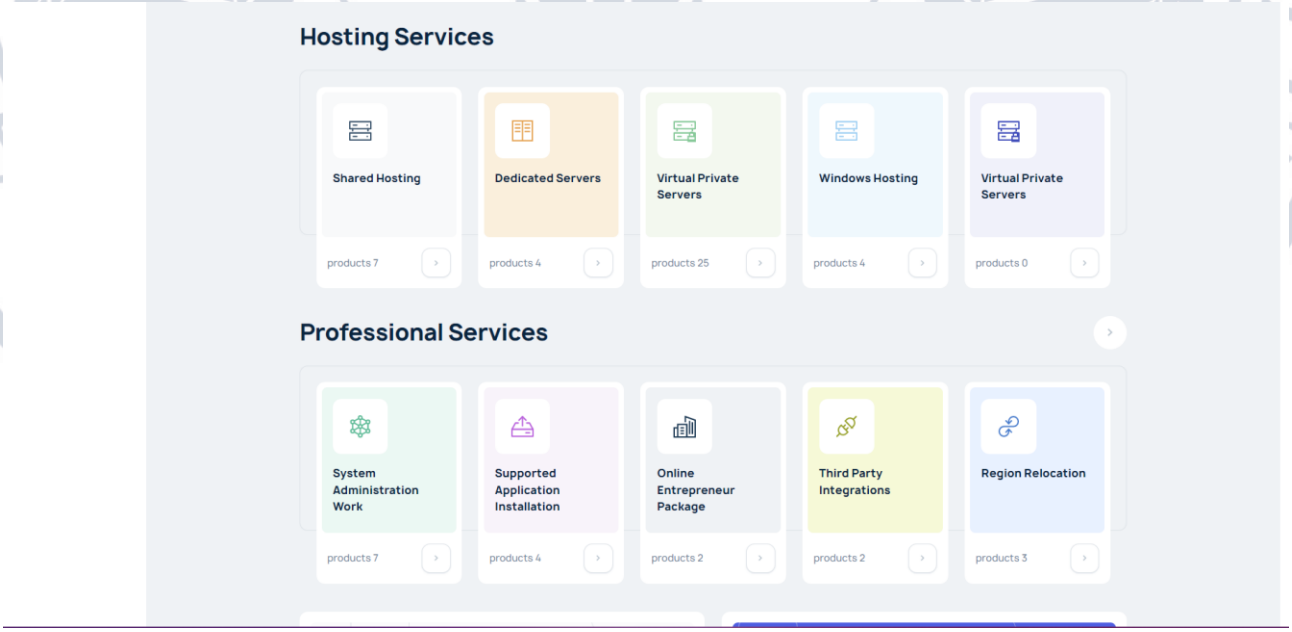

Рисунок 2.8 – послуги для користувача

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.

Ключовою особливістю Flux є односпрямованість передачі між компонентами архітектури Flux. Ця архітектура накладає обмеження на потік даних, особливо усунення можливості незалежного оновлення стану компонента. Такий підхід робить потік даних передбачуваним та полегшує відстеження джерел можливих помилок у 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, доки попередня дія не буде повністю оброблена компонентом. Хоча дії від роботи джерел дій, таких як користувачі, надходять асинхронно, їхнє відправлення є синхронним процесом. На додаток до імені, дія може мати корисне навантаження, що містить дані, пов'язані з дією.

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 – це централізоване місце для стану програми. Згідно з Flux, інші компоненти не мають значного стану (з архітектурної точки зору). Зміни стану сховища суворо ґрунтуються на даних дій та старому стані сховища.

Подання (англ. View) - це компонент, що зазвичай відповідає за подання інформації користувачеві. В архітектурі Flux, яка технічно може взагалі не торкатися внутрішнього пристрою уявлень, це кінцева точка потоку даних. Інформаційна архітектура дбає тільки про дані, що надходять до системи (тобто повертаються до сховища).

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 2.3.1).

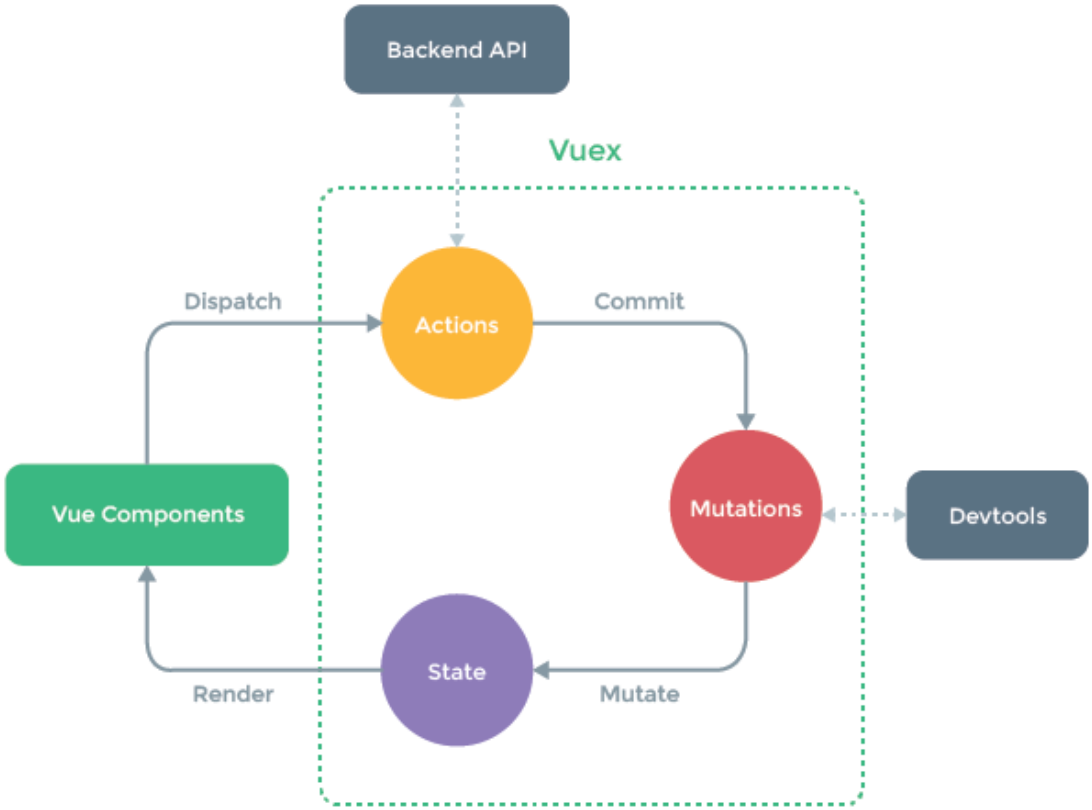

Рисунок 2.3.1 – Модель взаємодії компонентів програмного продукту

Розробка методу веб-додатку, можна поділити на два етапи, перший – це можливість користувача аторизуватись в веб додатку (рис 2.3.2).

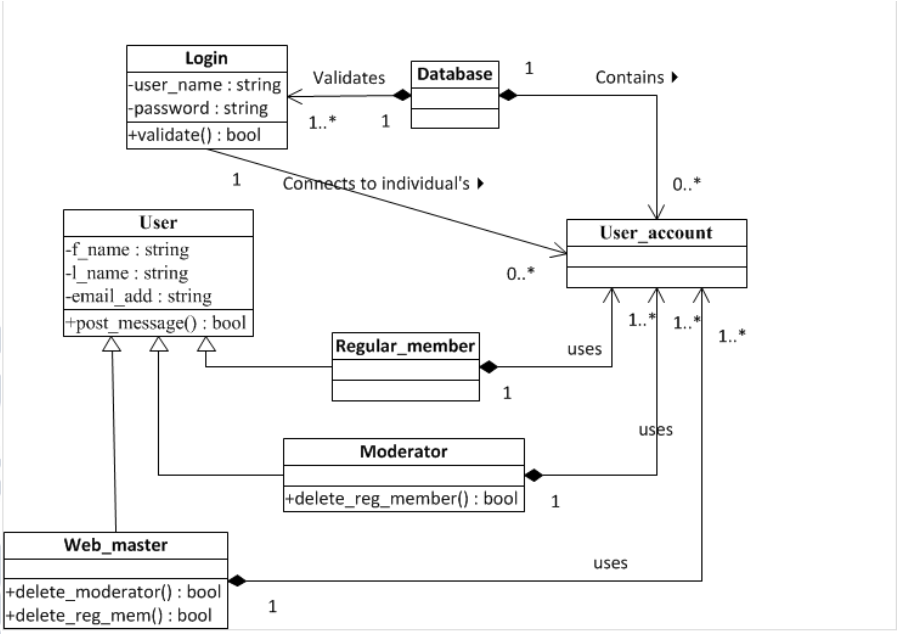

Рисунок 2.3.2 – Модель авторизації користувача

Модельна Архітектура реалізує вимоги до функціональності та продуктивності системи, а також такі вимоги, як надійність, масштабованість, портативність та доступність (рис 2.3.3).

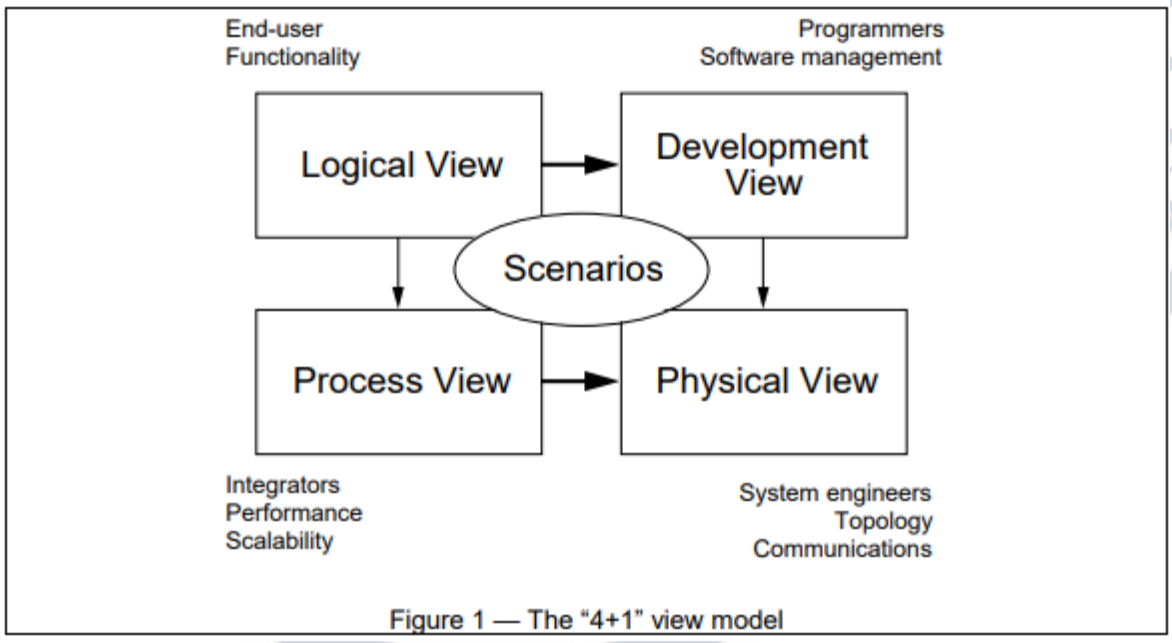

Рисунок 2.3.3 – Базова модель архітектури

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 2.3.4).

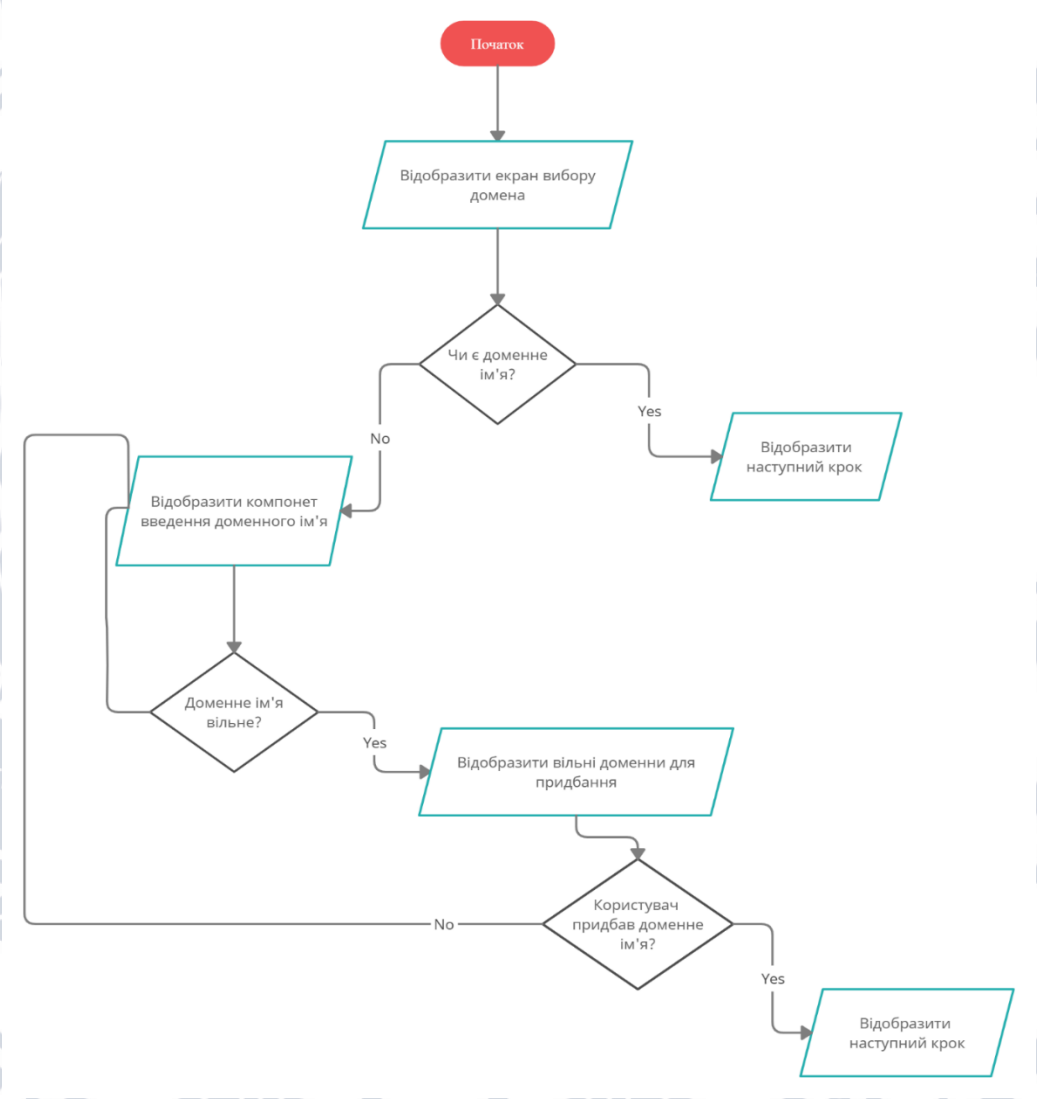

Рисунок 2.3.4 – Алгоритм Створення веб-сайту (етап 1)

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 бачить 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 оффрмлень 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.

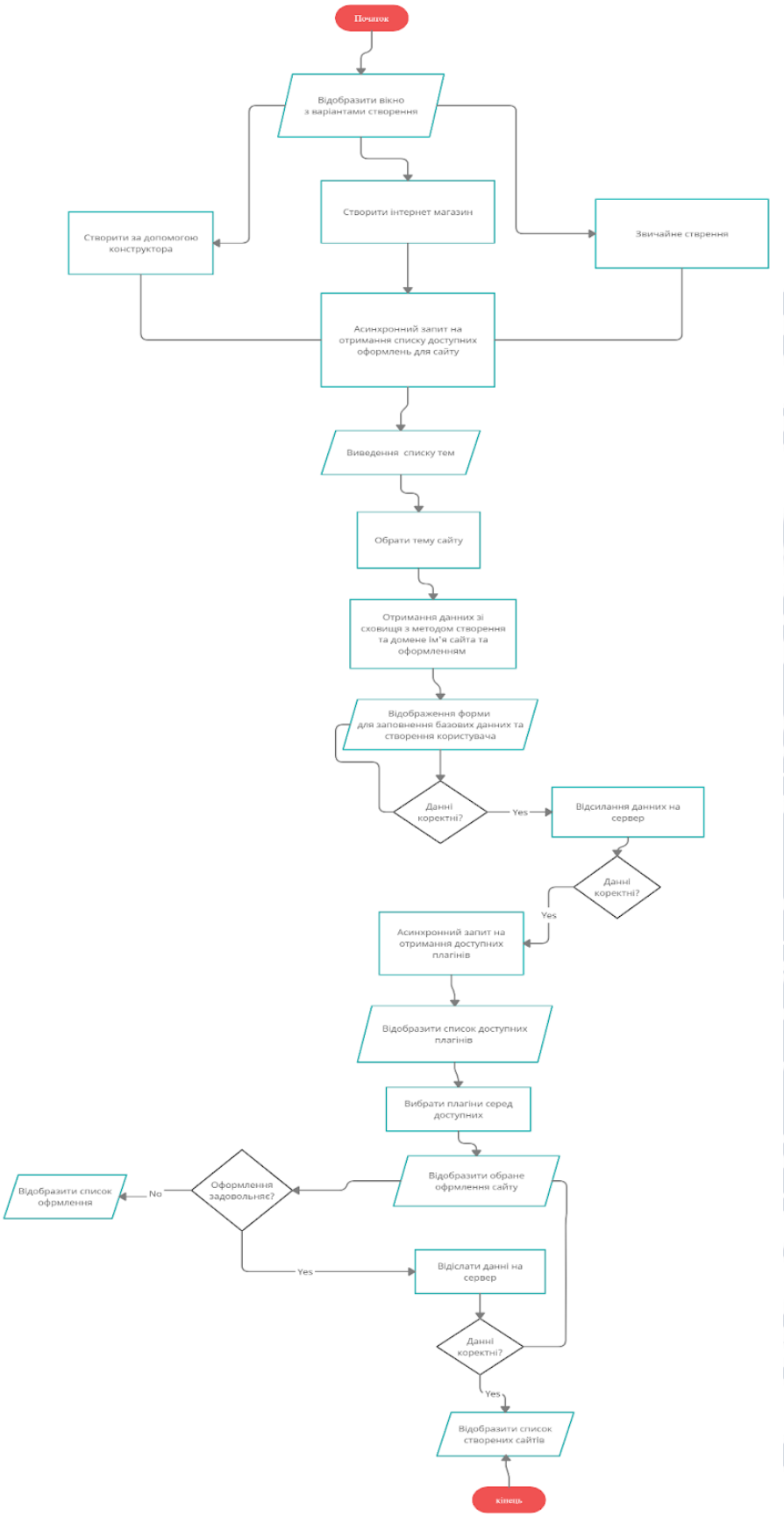

Рисунок – Алгоритм Створення веб-сайту (етап 2)

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 рекомендаційну систему для оренди серверів та доменів на основі контентної фільтрації, 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 фільтрації, 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 .

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%. 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. 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. 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 клєнтської 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. 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 це програмна платформа, що визначає структуру програмної системи. Програмне забезпечення, яке полегшує розробку та інтеграцію різних компонентів великого програмного проекту.

Vue використовує синтаксис шаблонів на основі HTML, що дозволяє вам декларативно прив'язувати рендеринг DOM до базових екземплярів даних Vue. Усі шаблони Vue є дійсними HTML та можуть бути проаналізовані браузерами та парсерами HTML. Усередині Vue компілює шаблони у віртуальні функції рендерингу DOM. У поєднанні з реактивною структурою Vue може інтелектуально підраховувати кількість компонентів та повторно відображати їх, застосовуючи мінімальні маніпуляції з DOM у разі зміни стану програми.

Vue.js (вимовляється як «подання» від англійського «view») — це середовище JavaScript, яке використовує шаблон MVVM для створення інтерфейсів користувачів на основі моделей даних за допомогою реактивної прив'язки даних.

Vue дозволяє використовувати синтаксис шаблону або писати свої функції рендерингу безпосередньо за допомогою JSX. Для цього просто замініть шаблон на функцію рендерингу. Можливості рендерингу відкривають можливості

потужних шаблонів на основі компонентів. Наприклад, нова система трафіку буде повністю компонентною і використовуватиме функції рендерингу під капотом.

React (раніше відомий як React.js, ReactJS) — це інструмент створення інтерфейсу користувача з відкритим вихідним кодом, призначений для вирішення проблеми часткового оновлення вмісту веб-сторінки, що виникає при розробці односторінкових додатків. Бібліотека JavaScript. Працює на Facebook, Instagram та спільнотах окремих розробників.

React дозволяє розробникам створювати великомасштабні веб-застосунки, які використовують дані, які згодом змінюються, без перезавантаження сторінки. Його мета — бути швидким, простим і масштабованим. React обробляє тільки інтерфейс користувача вашої програми. Він відповідає уявленням у шаблоні Model-View-Controller (MVC) і може використовуватися у поєднанні з іншими бібліотеками JavaScript або у великих середовищах MVC, таких як AngularJS. Його також можна використовувати з плагінами на основі React для обробки частини створення веб-додатку без інтерфейсу користувача. Як бібліотека інтерфейсу користувача React найчастіше використовується в поєднанні з іншими бібліотеками, такими як Redux.

Angular (звичайно званий фреймворком Angular 2 або Angular 2+ або пізнішою версією) - це інтерфейсний фреймворк з відкритим вихідним кодом, написаний на TypeScript, розроблений під керівництвом команди Angular у Google, а також доступний у приватній спільноті. мають. Розробники та компанії. Angular - це AngularJS, переосмислений та повністю переписаний тією ж командою розробників.

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 – це інтегроване середовище розробки для JavaScript, HTML та CSS від JetBrains, побудоване на платформі IntelliJ IDEA. WebStorm - це спеціалізована версія PhpStorm, що пропонує частину його функцій. WebStorm поставляється з встановленими модулями JavaScript (такими як Node.js), які ви можете використовувати з PhpStorm безкоштовно.

WebStorm підтримує JavaScript, CoffeeScript, TypeScript та Dart. WebStorm пропонує автодоповнення, аналіз коду «на льоту», навігацію за кодом, рефакторинг, старіння та інтеграцію із системами контролю версій. Важливою перевагою інтегрованого середовища розробки WebStorm є робота з проектами (рефакторинг коду JavaScript, що міститься в різних файлах та папках проекту, вбудовування його в HTML тощо). Підтримується множинна вкладеність (де сценарій JavaScript вкладено в HTML документ, а інший код HTML вкладено з вкладеним у нього JavaScript) - для таких конструкцій 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.

3.4 Архітектура програмного додатку

Архітектура програмного додатку містить наступні розділи:

- Adapters – використовуються для перетворення даних у структури даних які потрібні в конкретній задачі.
- Assets – розділ використовується для зберігання іконок, картинок, та деяких CSS стилів.
- Components – в розділі зберігаються вже готові компоненти які використовуються для написання та розширення функціоналу додатку.
- Composable – розділ який містить файли з винесеною бізнес-логікою додатку, для збереження принципу DRY, в додатку були створені composable які виносять однакову бізнес-логіку в окремий файл до якого можна звернутися з різних компонентів додатку.
- Configs – папка де прописуються налаштування підключених бібліотек.
- Const – розділ в якому винесені всі постійні значення додатку. Ці значення можуть зустрічатися в різних частинах додатку.
- i18n – розділ бібліотеки яка використовується для реалізації багатомовності додатку, містить файли з обраною мовою та словником.
- Layout – розділ в якому написані Styled Components для стилізації написаних компонентів.
- Router – розділ використовується для прописування URL потрібної сторінки. Визначає статичні та динамічні маршрути за допомогою інтуїтивно зрозумілого та потужного синтаксису. Перехоплює будь-яку навігацію та точно контролюйте її результат. Зіставляє кожен маршрут із компонентом, який має відображатися.

- Store – розділ використовується для управління глобальним сховищем додатку. У центрі кожної програми Vuex знаходиться store. «Store» — це, по суті, контейнер, який зберігає стан вашої програми.
- Theme – розділ в якому задаються всі стани кнопок та різних елементів інтерфейсу.
- Utils – розділ де прописані різноманітні рішення які допомагають проекту функціонувати.
- Views – розділ в якому містяться всі елементи які будуть відображені у додатку.

3.5 Опис основного коду програми

Основний код програми відповідає за відображення всієї архітектури веб-додатку (Рис 3.5.1) цей файл створює DOM елемент від якого наслідуються всі компоненти та стилі прописані у проекті, також від цього файлу починає працювати Flux концепція які відповідає за передачу даних зверху вниз по дереву компонентів, за допомогою цього принципу та реактивності працює SPA, та не перезавантажує додаток кожен раз при зміні даних.

```
      <template>
  <ThemeProvider
    class="provider"
    :theme="theme.primary"
  >
    <Spinner
      v-if="!isAuthorized"
      position="absolute"
      top="50%"
      left="50%"
      style="transform: translate(-50%, -50%)"
    />
    <template
      v-else
    >
      <GridBox
        min-height="100vh"
        :padding="{
          _: '0',
          lg: '16px'
        }"
        :grid-template-columns="{
          _: '1fr',
          lg: '160px 1fr',
          xl: '176px 1fr',
```

```
          xll: '192px 1fr',
          xxl: '208px 1fr'
        }"
      >
        <NavMain
          :is-opened-menu="isOpenedMenu"
          @on-menu-item-click="handleMenuItemClick"
        />
        <FlexBox
          min-width="0"
          flex-grow="1"
          position="relative"
          background-size="100%"
          flex-direction="column"
          background-color="grey.g50"
          background-position="0 102px"
          background-repeat="no-repeat"
          :background-image="`url(${clientAreaBackground})`"
          :border-radius="{
            _: '0',
            lg: '16px'
          }"
        >
          <Header
            :is-primary="!isMarketplacePage"
            @on-burger-button-click="handleBurgerButtonClick"
          />
          <FlexBox
            as="main"
            flex-grow="1"
            flex-direction="column"
            :padding="{
              _: isMarketplacePage ? '64px 0 40px 0' : '96px 0 40px 0',
              lg: isMarketplacePage ? '0 0 63px 0' : '121px 0 63px 0'
            }"
          >
            <RouterView/>
          </FlexBox>
        </FlexBox>
      </GridBox>
      <Overlay
        :display="{
          lg: 'none'
        }"
        z-index="11"
        :is-opened="isOpenedMenu"
        @click="handleBurgerButtonClick"
      />
    </template>
    <ContainerToaster
      id="toast-container"
    />
  </ThemeProvider>
</template>
<script>
  import {
    ref, watch, onMounted, computed,
  } from 'vue';
  import { useStore } from 'vuex';
  import { injectGlobal, ThemeProvider } from 'vue3-styled-components';
  import theme from '@/theme/index';
  import FlexBox from '@/layout/Box/FlexBox';
  import GridBox from '@/layout/Box/GridBox';
  import Header from '@/views/Header/Header';
  import NavMain from '@/views/NavMain/NavMain';
  import Overlay from '@/layout/Overlay/Overlay';
```

```
  import Spinner from '@/components/Spinner/Spinner';
  import ContainerToaster from '@/layout/Container/ContainerToaster';
  import useMarketPlaceView from '@/composable/useMarketPlaceView';
  import clientAreaBackground from '@/assets/img/client-area-background.svg';
  import ManropeRegularWoff2 from '@/assets/fonts/Manrope-Regular.woff2';
  import ManropeRegularWoff from '@/assets/fonts/Manrope-Regular.woff';
  import ManropeMediumWoff2 from '@/assets/fonts/Manrope-Medium.woff2';
  import ManropeMediumWoff from '@/assets/fonts/Manrope-Medium.woff';
  import ManropeBoldWoff2 from '@/assets/fonts/Manrope-Bold.woff2';
  import ManropeBoldWoff from '@/assets/fonts/Manrope-Bold.woff';
  import ManropeExtraBoldWoff2 from '@/assets/fonts/Manrope-ExtraBold.woff2';
  import ManropeExtraBoldWoff from '@/assets/fonts/Manrope-ExtraBold.woff';
  import { createFormData } from '@/utils';

  injectGlobal([`
  @font-face {
    font-family: "Manrope";
    font-display: swap;
    src: url(${ManropeRegularWoff2}) format("woff2"), url(${ManropeRegularWoff})
format("woff");
    font-weight: 400;
    font-style: normal;
  }

  @font-face {
    font-family: "Manrope";
    font-display: swap;
    src: url(${ManropeMediumWoff2}) format("woff2"), url(${ManropeMediumWoff})
format("woff");
    font-weight: 500;
    font-style: normal;
  }

  @font-face {
    font-family: "Manrope";
    font-display: swap;
    src: url(${ManropeBoldWoff2}) format("woff2"), url(${ManropeBoldWoff})
format("woff");
    font-weight: 700;
    font-style: normal;
  }

  @font-face {
    font-family: "Manrope";
    font-display: swap;
    src: url(${ManropeExtraBoldWoff2}) format("woff2"),
url(${ManropeExtraBoldWoff}) format("woff");
    font-weight: 800;
    font-style: normal;
  }


  html {
    min-height: 100vh;
    box-sizing: border-box;
  }

  * {
    box-sizing: inherit;
  }

  *::after {
    box-sizing: inherit;
  }

  *::before {
```

```
  box-sizing: inherit;
}

body {
  min-height: 100vh;
  min-width: 375px;
  margin: 0;
  font-family: "Manrope", arial, sans-serif;
  font-weight: 500;
  background-color: #FFFFFF;
}

.visibility-hidden {
  position: absolute;
  width: 1px;
  height: 1px;
  margin: -1px;
  border: 0;
  padding: 0;
  white-space: nowrap;
  clip-path: inset(100%);
  clip: rect(0 0 0 0);
  overflow: hidden;
}

img {
  max-width: 100%;
  height: auto;
}

button {
  cursor: pointer;
}

#app {
  min-height: 100vh;
}

.provider {
  min-height: 100vh;
}

.swiper-slide {
  display: grid;
  grid-template-columns: 1fr;
}

body .swiper-button-lock {
  display: none;
}

.checkbox-group-text {
  border-radius: 0;
}

.checkbox-group-item:first-child .checkbox-group-text {
  border-radius: 12px 0 0 12px;
}

.checkbox-group-item:last-child .checkbox-group-text {
  border-radius: 0 12px 12px 0;
}

@media screen and (max-width: 1280px) {
  .show-nav {
    position: fixed;
```

```
      width: 100%;
    }

    .show-main-nav {
      overflow: hidden;
    }
  }
`]);

  export default {
    name: 'App',

    components: {
      Header,
      NavMain,
      FlexBox,
      GridBox,
      Spinner,
      Overlay,
      ThemeProvider,
      ContainerToaster,
    },

    setup() {
      const store = useStore();
      const {
        isMarketplacePage,
      } = useMarketPlaceView();

      const isOpenedMenu = ref(false);
      const isLoadingCurrentUser = ref(false);

      const isAuthorized = computed(() =>
store.getters['user/getUserAuthorizedStatus']);

      const fetchCurrentUser = () => {
        isLoadingCurrentUser.value = true;

        return store.dispatch('user/loadCurrentUser')
          .finally(() => {
            isLoadingCurrentUser.value = false;
          });
      };

      const loadUserAuthorizedStatus = () =>
store.dispatch('user/loadUserAuthorizedStatus');

      const openMainMenu = () => {
        document.body.classList.add('show-nav');
        document.documentElement.classList.add('show-main-nav');
      };

      const closeMenu = () => {
        document.body.classList.remove('show-nav');
        document.documentElement.classList.remove('show-main-nav');
      };

      const handleMenuItemClick = () => {
        isOpenedMenu.value = false;
      };

      const updateAccountDetails = () => {
        if (localStorage.getItem('sid')) {
          store.dispatch('account/updateDefaultClientArea', createFormData({
            newCaDefault: true,
          }));
```

```
          localStorage.removeItem('sid');
        }
      };

      const handleBurgerButtonClick = () => {
        isOpenedMenu.value = !isOpenedMenu.value;
      };

      watch(isOpenedMenu, (isOpened) => (isOpened ? openMainMenu() :
closeMenu()));

      onMounted(() => {
        fetchCurrentUser();
        updateAccountDetails();
        loadUserAuthorizedStatus();
      });

      return {
        theme,
        isAuthorized,
        isOpenedMenu,
        isLoadingCurrentUser,
        clientAreaBackground,
        handleMenuItemClick,
        handleBurgerButtonClick,
        isMarketplacePage,
      };
    },
  };
</script>

<style>
@import "assets/css/normalize.css";
@import "~swiper/swiper-bundle.min.css";
</style>
```

Рисунок 3.5.1

Файл з конфігурацією проекту (Рис 3.5.2) тут за допомогою бібліотеки Webpack прописуються всі властивості проекту які потрібні для його швидкої розробки.

```
// optional config file
// see https://cli.vuejs.org/config/#global-cli-config
const path = require('path');

module.exports = {
  lintOnSave: false,

  devServer: {
    port: 9000,

  },

  configureWebpack: {
    resolve: {
      extensions: ['.js', '.vue', '.json'],
      alias: {
        '@': path.resolve(__dirname, 'src'),
      },
    },
  },

  pages: {
```

```
    index: {
      entry: 'src/main.js',
      // template title tag needs to be <title><%=
htmlWebpackPlugin.options.title %></title>
      title: 'TMD Client Area',
    },
  },

  chainWebpack: (config) => {
    config.module
      .rule('svg-sprite')
      .use('svgo-loader')
      .loader('svgo-loader');
  },
};
```

Рисунок 3.5.2

Файл в якому прописуються всі бібліотеки та залежності (Рис 3.5.3) які використовувалися для проекту, також в цьому файлі фіксуються всі версії бібліотек які використовуються, та можливо подивитись що саме та на якій версії встановлено.

```
{
  "name": "frontend-client-area",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "serve": "vue-cli-service serve",
    "build": "vue-cli-service build",
    "lint": "vue-cli-service lint",
    "dev": "vue-cli-service serve",
    "serve:prod": "vue-cli-service serve --mode=production ",
    "test": "vue-cli-service test:unit"
  },
  "dependencies": {
    "@kangc/v-md-editor": "^2.3.10",
    "@popperjs/core": "2.9.2",
    "@styled-system/theme-get": "5.1.2",
    "@vueform/multiselect": "1.4.0",
    "@vuelidate/core": "^2.0.0-alpha.21",
    "@vuelidate/validators": "^2.0.0-alpha.18",
    "axios": "0.21.1",
    "chart.js": "2.9.4",
    "core-js": "3.8.1",
    "he": "1.2.0",
    "js-cookie": "2.2.1",
    "leaflet": "1.7.1",
    "linkify-it": "3.0.2",
    "lodash": "4.17.20",
    "markdown-linkify": "1.0.3",
    "moment": "^2.29.1",
    "moment-timezone": "0.5.33",
    "nanoid": "3.1.28",
    "prismjs": "1.24.1",
    "query-string": "6.14.0",
    "styled-system": "^5.1.5",
    "swiper": "^6.7.5",
    "vue": "3.0.11",
    "vue-i18n": "9.1.7",
    "vue-router": "4.0.1",
    "vue-scrollto": "2.20.0",
    "vue-toastification": "^2.0.0-rc.1",
    "vue3-autocounter": "1.0.6",
```

```
    "vue3-styled-components": "^1.2.1",
    "vuex": "^4.0.2"
  },
  "devDependencies": {
    "@vue/cli-plugin-babel": "~4.5.9",
    "@vue/cli-plugin-eslint": "5.0.0-beta.1",
    "@vue/cli-plugin-router": "~4.5.9",
    "@vue/cli-plugin-unit-jest": "~4.5.9",
    "@vue/cli-plugin-vuex": "~4.5.9",
    "@vue/cli-service": "~4.5.9",
    "@vue/compiler-sfc": "^3.1.4",
    "@vue/eslint-config-airbnb": "^5.3.0",
    "@vue/test-utils": "^2.0.0-rc.9",
    "babel-eslint": "^10.1.0",
    "eslint": "^7.30.0",
    "eslint-plugin-vue": "^7.12.1",
    "svgo": "^1.1.0",
    "svgo-loader": "^2.1.0",
    "vue-cli-plugin-svg-sprite": "~1.0.0",
    "vue-jest": "^5.0.0-0"
  }
}
```

Рисунок 3.5.3

Файл з використанням API (Рис 3.5.4) в цьому файлі прописана можливість отримувати та відправляти Get та Post запити, та реалізовувати RESTful API у веб-додатку.

```
import axios from 'axios';
import { BASE_API_URL } from '@/const/url';
import { redirectToLogin, setSession, isAuthenticationFailed } from
'@/utils/api';

const createApi = () => {
  const api = axios.create({
    baseURL: BASE_API_URL,
    timeout: 50000,
    withCredentials: true,
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'Access-Control-Allow-Origin': '*',
      'X-Requested-With': 'XMLHttpRequest',
    },
  });

  setSession();

  const onSuccess = (response) => response;

  const onFail = (error) => {
    if (isAuthenticationFailed(error)) {
      redirectToLogin();
    }

    return Promise.reject(error);
  };

  api.interceptors.response.use(onSuccess, onFail);

  return api;
};

const api = createApi();
```

```
export default api;
```

Рисунок 3.5.4

В цьому файлі (Рис 3.5.5) відбувається додавання бібліотек та генерація основного файлу додатку.

```
import { createApp } from 'vue';
import Toast from 'vue-toastification';
import Vue3Autocounter from 'vue3-autocounter';
import VueMarkdownEditor from '@kangc/v-md-editor';
import VMdPreview from '@kangc/v-md-editor/lib/preview';
import vuepressTheme from '@kangc/v-md-editor/lib/theme/vuepress.js';
import enUS from '@kangc/v-md-editor/lib/lang/en-US';
import Prism from 'prismjs';
import toastConfig from '@/configs/toast';
import App from './App';
import router from './router';
import store from './store';
import i18n from './i18n/translations';
import 'vue-toastification/dist/index.css';
import '@kangc/v-md-editor/lib/theme/style/vuepress.css';
import '@kangc/v-md-editor/lib/style/base-editor.css';
import '@kangc/v-md-editor/lib/style/preview.css';

VMdPreview.use(vuepressTheme, {
  Prism,
});

VueMarkdownEditor.lang.use('en-US', enUS);

VueMarkdownEditor.use(vuepressTheme, {
  Prism,
});

createApp(App)
  .use(store)
  .use(router)
  .use(Vue3Autocounter)
  .use(Toast, toastConfig)
  .use(i18n)
  .use(VueMarkdownEditor)
  .use(VMdPreview)
  .mount('#app');
```

Рисунок 3.5.5

3.6 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.

ВИСНОВКИ

У магістерській кваліфікаційній роботі було досліджено та реалізовано рекомендаційну систему на основі контентної фільтрації для оренди серверів та доменів.

Були проаналізовані існуючі методи формування онлайн-рекомендацій, а саме: метод колаборитвної фільтрації, заснований на неявному відгуці, а також метод контентної фільтрації, заснований на алгоритмі TF-IDF та косинусах подібності

Була розглянута проблема контентної фільтрації з використанням негативного неявного відгуку, а також заміщення її контентною фільтрацією у разі, коли не вистачає вільних доменів та серверів для точного формування рекомендацій. Це дозволило збільшити точність надання рекомендацій для користувачів.

Були вирішенні такі задачі:

- Досліджено та покращено алгоритм контентної фільтрації
- Було покращено алгоритм пропонування доменів та серверів користувачу.
- розроблено можливість встановлення серверу на базі системи керування вмістом TMD Hosting;
- розроблено можливість отримання потрібних рекомендацій по оптимізації серверу;
- розроблено можливість отримання потрібних рекомендацій по безпеці серверу;
- удосконалено можливість контентної фільтрації доменів та серверів.
- удосконалено можливість встановлення плагінів на сервер;
- удосконалено можливість отримання статистики з відвідування серверу;
- спроектовано архітектуру проекту на базазі flux;

В результаті створеного веб-додатку подальшого розвитку дістав метод створення та фільтрації серверів та доменів на базі системи керування вмістом

TMD Hosting, який за рахунок управління низкою створених за різними технологіями серверів та їх деталізації, дозволяє користувачеві підвищити продуктивність управління серверами та доменами, адмініструвати декілька серверів 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 серверів, їх деталізації, формування відповідних порад з безпеки та оптимізації серверів, що дозволяє користувачу отримати адаптовані дані для більш якісного управління сервером.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hu Y., Koren Y. and Volinsky C. (2008), “Collaborative filtering for implicit feedback datasets”, Data Mining, ICDM’08. Eighth IEEE International Conference on. IEEE, pp. 263–272
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. Jannach D., Zanker M., Felfernig A. Friedrich G. Recommender Systems. An Introduction. New York: Cambridge University Press 32 Avenue of the Americas, 2011. 352 P.
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. Melville P., Sindhwani V. Recommender systems. Encyclopedia of Machine Learning. 2010. p. 30

18. 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.

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

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

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

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

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

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

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

26. Методичні вказівки щодо розробки та оформлення магістерської атестаційної роботи за спеціальністю 8.05010101 – Інформаційні управляючі системи та технології. Освітньо-кваліфікаційний рівень – магістр / Упоряд.:

Левикін В.М., Міхнов Д.К., Саєнко В.І., Євланов М.В., Міхнова А.В., Керносов М.А. – Харків: ХНУРЕ, 2012. – 28 С.

27. Chalyi S. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних обмежень типу «next» / S. Chalyi, V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 4 (56). – С. 105-109. – doi:https://doi.org/10.26906/SUNZ.2019.4.105.

28. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Моделювання контексту в рекомендаційних системах. Науковий журнал «Проблеми інформаційних технологій», 2018, №. 1(023). С. 21-26.

29. Chalyi S. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних обмежень типу «next» / S. Chalyi, V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 4 (56). – С. 105-109. – doi:https://doi.org/10.26906/SUNZ.2019.4.105.

30. Савчук Т.О., Застосування кластерного аналізу для колаборативної фільтрації / Т.О. Савчук, А.В.Сакалюк // Вісник Хмельницького національного університету. –2011 – №1– С. 186-192 6. Sarwar B. M. Item-based collaborative filtering recommendation algorithms / B. M. Sarwar, G. Karypis, J. A. Konstan // Proceedings of ACM WWW '01, pp. 285–295, ACM, 2001. 7. Hu Y., Koren Y. and Volinsky C. (2008), "Collaborative filtering for implicit feedback datasets", Data Mining, ICDM'08. Eighth IEEE International Conference on. IEEE, pp. 263–272.

31. L. H. Patil, and M. Atique, "A novel feature selection based on information gain using WordNet," Science and Information Conference IEEE, 2013. С. - 625-629.

32. Chalyi S. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних обмежень типу «next» / S. Chalyi, V. Leshchynskyi, I. Leshchynska // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2019. – Т. 4 (56). – С. 105-109. – doi:https://doi.org/10.26906/SUNZ.2019.4.105.

33. Чалий С.Ф., Лещинський В.О., Лещинська І.О. Доповнення вхідних даних рекомендаційної системи в ситуації циклічного холодного старту з використанням темпоральних обмежень типу «NEXT». Системи управління, навігації та зв'язку, 2019. Вип. 4(56). С. 105-109.

34. Y. Lu, and M. Liang, “Improvement of Text Feature Extraction with Genetic Algorithm,” New Technology of Library & Information Service, 2014. C. - 523–525. 36. L. H. Wang, “An Improved Method of Short Text Feature Extraction Based on Words Co-Occurrence,” Applied Mechanics & Materials, vol. , 2014. C. - 842-845.

35. H. Liang, et al, “Text feature extraction based on deep learning: a review:,” Eurasip Journal on Wireless Communications & Networking, vol. 2017, 2017. C. - 211.

36. S. Zhou , et al, “Characteristic representation method of document based on Word2vector,” Journal of Chongqing University of Posts & Telecommunications, vol. 30, 2018. C.

37. Y. Chen, et al, “A fast clustering algorithm based on pruning unnecessary distance computations in DBSCAN for high-dimensional data,” Pattern Recognition, 2018. C. - 83. 40. S. Tan, “Neighbor-weighted K-nearest neighbor for unbalanced text corpus,” Expert Systems with Applications. vol. 28, 2005. C. - 667-671

38. He Shaojun, et al, “The Capability Analysis on the Characteristic Selection Algorithm of Text Categorization Based on F1 Measure Value,” IEEE Explore, C. - 742 -746.

Марущак Денис Русланович
Прізвище, ім’я по батькові
Інформаційних і прикладних технологій
Факультет
122 Комп’ютерні науки
Шифр і назва спеціальності
Комп’ютерні технологій обробки даних
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «ДОСЛІДЖЕННЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ДЛЯ ОРЕНДИ СЕРВЕРІВ ТА ДОМЕНІВ НА ОСНОВІ КОНТЕНТНОЇ ФІЛЬТРАЦІЇ» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

(дата) (підпис здобувача освіти)