МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «УНІВЕРСИТЕТ «КРОК»

Фаховий коледж Університету «КРОК»

ДИПЛОМНА РОБОТА

за темою

«Розробка та створення сайту магазину товарів для дому»

Студент 4 курсу групи ПЗ-20к

Шок Владислав Павлович

(прізвище, ім'я та по-батькові студента)

Керівник дипломної роботи

к. фіз.-мат. н.

(посада керівника)

Кириченко В.В.

(прізвище, ім'я та по-батькові керівника)

До захисту

(резолюція «До захисту»)

Шок

(підпис студента)

12.06.2024

(дата)

(підпис викладача)

Київ, 2024 рік

Скорочення

HTML (HyperText Markup Language) – мова розмітки, яка застосовується при створенні веб-сторінок в Інтернеті.

CSS (Cascading Style Sheets) – мова стилів, за допомогою якої описують вигляд документів HTML.

JS (JavaScript) – мова програмування, яка використовується для створення інтерактивних елементів на веб-сторінках.

PHP (Hypertext Preprocessor) – широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом, що особливо підходить для веб-розробки.

MVC (Model-View-Controller) – архітектурний шаблон для розробки програмного забезпечення, який розділяє додаток на три взаємозалежні частини: модель, представлення і контролер.

DOM (Document Object Model) – об'єктна модель документа, яка використовується для доступу до HTML і XML документів.

SQL (Structured Query Language) – мова програмування, яка використовується для управління і запиту даних в реляційних базах даних.

MySQL – система управління реляційними базами даних з відкритим вихідним кодом.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, заснований на підмножині мови JavaScript.

SEO (Search Engine Optimization) – процес покращення видимості веб-сайту в органічних результатах пошукових систем.

API (Application Programming Interface) – інтерфейс програмування додатків, який дозволяє взаємодію між програмними компонентами.

SSL (Secure Sockets Layer) – протокол для встановлення захищених з'єднань через Інтернет.

AOP (Aspect-Oriented Programming) – аспектно-орієнтоване програмування, парадигма програмування, яка дозволяє модульно розділяти функціональність програми.

IoC (Inversion of Control) – інверсія керування, принцип програмування, який передбачає передачу контролю над створенням об'єктів контейнеру.

DI (Dependency Injection) – впровадження залежностей, техніка, при якій об'єкт отримує інші об'єкти, від яких він залежить.

NoSQL – тип систем управління базами даних, які не використовують реляційну модель.

Структура

ЗМІСТ

Вступ

У наш час, коли Інтернет стає невід'ємною частиною життя людей, електронна комерція набирає обертів, перетворюючись не просто на модну тенденцію, а й на життєво необхідний інструмент для процвітання будь-якого бізнесу.

Згідно з статистичними даними, щорічно спостерігається стрімке зростання обсягів онлайн-продажів, а кількість покупців, які віддають перевагу онлайн-шопінгу, неухильно зростає.

У сучасному світі інтернет-магазини стали невід'ємною частиною електронної комерції, забезпечуючи зручність та доступність покупок для мільйонів користувачів по всьому світу.

Інтернет-магазини пропонують зручний спосіб шукати, порівнювати та купувати товари. Покупці можуть легко здійснювати покупки в будь-який час доби, що робить процес швидшим та зручнішим.

Цілодобова доступність. Інтернет-магазин відчинений цілодобово, 7 днів на тиждень, що дає можливість клієнтам здійснювати покупки в будь-який зручний для них час.

За статистичними даними, минулого року загальний об'єм продажів в інтернеті досягає 151 млрд грн. Саме тому присутність в мережі стає ключовим фактором успіху продавця. Проте важливо створити функціональний магазин, який відризнятися на фоні конкурентів.

Створення та розробка інтернет- магазину є складним та багатоаспектним процесом, який вимагає глибокого розуміння технічних аспектів, бізнес- процесів та потреб користувачів.

1. Значення інтернет-магазинів

На сьогодні веб-сайти стають все більше популярні . Практично будь-яка компанія має свій веб-додаток, творці новітніх технологій та продуктів можуть розраховувати на успіх з додаванням реклами в інтернеті.

В умовах зростання конкуренції та розвитку сучасних технологій, магазинам необхідно будувати інформаційні системи з набором необхідних інструментів для успішного ведення бізнесу в нинішніх умовах.

Швидкий розвиток Інтернету привів до необхідності створення сайтів для надання різних послуг. Перелік сайтів дуже великий. Є сайти, які містять коротку інформацію та послуги, також існують великі каталоги з детальною характеристикою товарів, їх зображеннями та цінами. Зазвичай, такий онлайн-каталог створюється для того, щоб люди могли знайти детальний опис і зображення товару.

Інтернет-магазин це система, створена за технологією системи електронної комерції. Як і звичайний магазин, інтернет-магазин реалізує такі основні функції, як представлення товару покупцеві, обробка замовлень, продаж товару та його доставка.

Інтернет-магазин поєднує елементи прямого маркетингу з традиційним магазином. У порівнянні зі звичайними магазинами відмінною рисою інтернет-магазинів є те, що вони можуть надавати велику кількість товарів і послуг, а також надавати клієнтам велику кількість інформації для прийняття рішення про його придбання.

Наразі, швидко зростає онлайн-аудиторія, у містах зростають онлайн-продажі, експерти виділяють тенденцію до зростання онлайн-продажів.

Кількість інтернет-магазинів збільшується з кожним місяцем, так як це зручно для покупців, не кажучи вже про економію часу.

1.1 Аналіз існуючих інтернет-магазинів

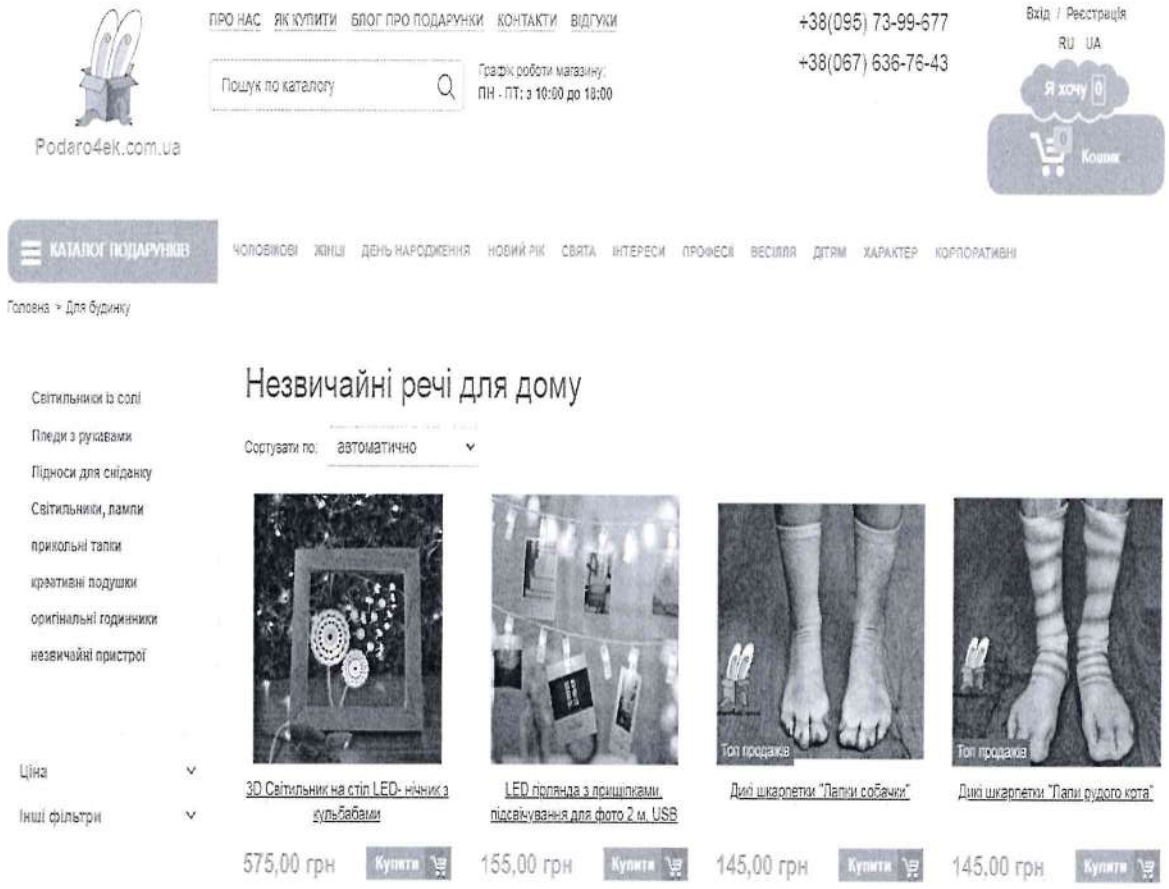

Рис.1.1. Сайт Podaro4ek.

Даний магазин виглядає досить зручно, є всі базові функції для користування. На перший погляд не складний дизайн, тож можна швидко розібратися з управлінням та навігацією по сайту. Є перелік досить неординарних та цікавих товарів. З плюсів одразу видно контактну інформацію, тож можна в випадку не до розумінь зв'язатись з підтримкою

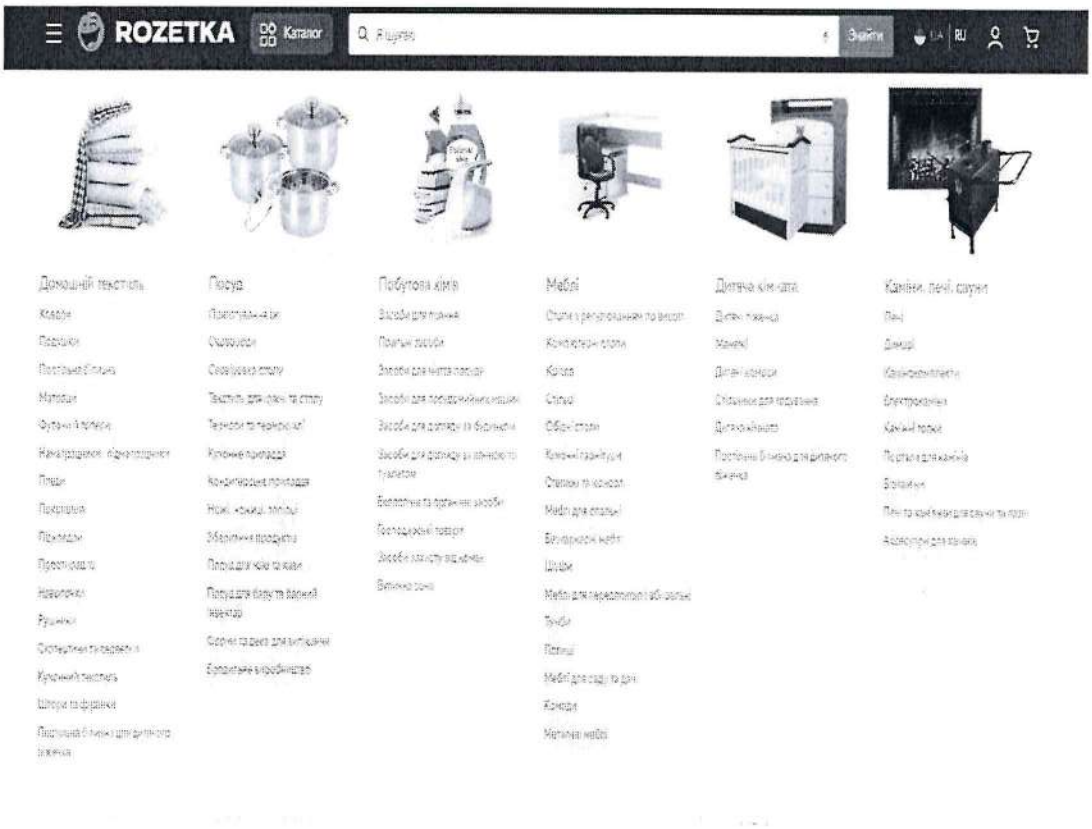

Рис.1.2. Сайт Rozetka

Цей магазин виглядає не дуже привабливо, але це не мішає йому бути багато-функціональним. В цьому сайті є дуже великий, та обширний перелік категорій , для любих потреб. Тож не зважаючи на дизайн та візуальну оцінку він є досить не поганим . Має всі можливі функції які потрібні інтернет магазину.

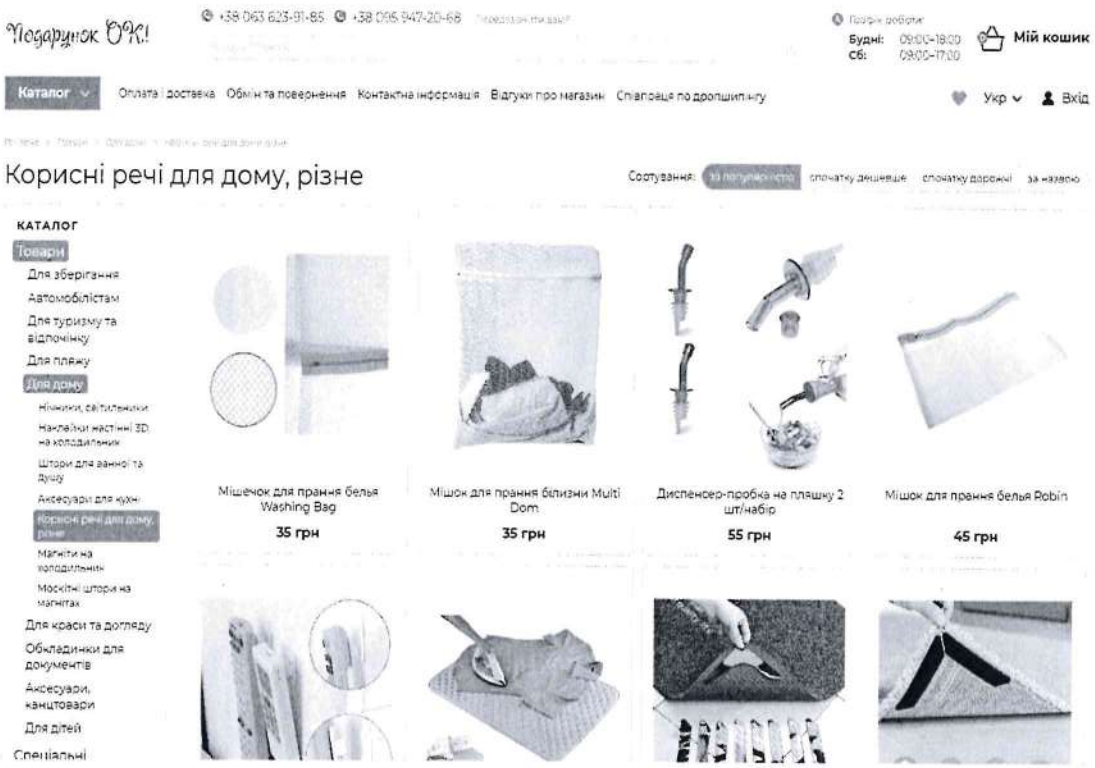

Рис.1.3. Сайт Подарунок ОК!

Цей магазин дуже схожий на (Рис.1.1) Простий дизайн, є весь потрібний функціонал. Має різноманітні види сортування товарів. Швидкий доступ до контактної інформації. Вказаний час роботи магазину. Та є можливість змінити мову. В цілому досить не погано.

1.2 Критерії оцінювання сайту

1. Функціональність

- Повнота функцій - відповідність наявних функцій потребам користувачів (каталог товарів, кошик, оформлення замовлення, особистий кабінет).
- Коректність роботи - відсутність помилок та багів, правильне функціонування всіх елементів сайту.

2. Зручність використання

- Навігація - легкість та інтуїтивність переміщення по сайту, логічна структура меню та категорій.
- Пошук - ефективність пошуку товарів, швидкість та точність результатів.
- Процес оформлення замовлення - прості та зрозумілі кроки оформлення замовлення, мінімальна кількість необхідних дій для завершення покупки.
- Адаптивність - коректне відображення сайту на різних пристроях (мобільні телефони, планшети, комп'ютери).

3. Дизайн

- Візуальна привабливість - сучасний та привабливий дизайн, відповідність стилю бренду.
- Читабельність - використання чітких шрифтів, контрастних кольорів, які забезпечують зручність читання.
- Відповідність стандартам - дотримання сучасних стандартів вебдизайну та рекомендацій щодо зручності використання.

4. Безпека

- Захист даних - використання SSL-сертифікатів для шифрування даних, забезпечення конфіденційності персональної інформації користувачів.
- Захист від атак - наявність захисту від DDoS-атак, SQL-ін'єкцій, XSS (міжсайтового скриптингу) та інших загроз.
- Автентифікація та авторизація - надійні методи автентифікації користувачів, захист облікових записів.

5. Продуктивність

- Швидкість завантаження - час завантаження сторінок сайту, оптимізація зображень та інших ресурсів.
- Стабільність - відсутність збоїв та падінь сайту під час високого навантаження.

- Масштабованість - можливість сайту адаптуватися до зростаючої кількості користувачів та збільшення обсягу даних.

6. Контент

- Якість контенту - відповідність контенту потребам та інтересам користувачів, актуальність та коректність інформації.
- Оновлення контенту - регулярність оновлення інформації на сайті, додавання нових товарів, акцій, статей.
- SEO-оптимізація - відповідність контенту вимогам пошукових систем, використання ключових слів, метатегів, якісних зображень.

7. Підтримка та обслуговування

- Технічна підтримка - наявність служби підтримки користувачів, швидкість та якість обслуговування.
- Документація та допомога - наявність інструкцій, FAQ, керівництв для користувачів.

1.3 Функціональності сайту

1. Головна сторінка

- Промоакції, знижки та спеціальні пропозиції.
- Персоналізовані рекомендації товарів.
- Меню категорій товарів.
- Пошуковий рядок.

2. Каталог товарів

- Структуровані категорії та підкатегорії товарів.
- Фільтрація за ціною, брендом, рейтингом тощо.
- Сортування товарів за популярністю, новизною, ціною.
- Короткі описи товарів з ціною, рейтингом та кількістю відгуків.

3. Сторінка товару

- Детальні описи товарів, характеристики, зображення.
- Відгуки та рейтинги користувачів.
- Кнопка "Додати в кошик".
- Рекомендовані товари.

4. Кошик

- Перегляд та редагування товарів у кошику.
- Підсумок вартості замовлення з урахуванням податків та доставки.
- Оформлення замовлення

2. Основні поняття та технології

Фронт-енд розробка є ключовою складовою процесу створення веб-сайтів та додатків, яка займається розробкою інтерфейсу користувача. Це включає в себе роботу з HTML, CSS та JavaScript, щоб створити зручне та привабливе візуальне середовище для взаємодії з користувачем. Фронт-енд розробка також враховує аспекти доступності, продуктивності та сумісності з різними браузерами та пристроями.

HTML (HyperText Markup Language) забезпечує структуру веб-сторінок, дозволяючи розміщувати текст, зображення, відео та інші елементи на сторінці. CSS (Cascading Style Sheets) відповідає за оформлення та стилізацію цих елементів, забезпечуючи гармонійний вигляд та належну презентацію контенту. JavaScript додає інтерактивності та динаміки, дозволяючи створювати реактивні елементи, обробляти події, виконувати анімації та працювати з даними в реальному часі.

Фронт-енд розробники також використовують різні фреймворки та бібліотеки, такі як React, Angular, Vue.js, Bootstrap, які спрощують та прискорюють процес розробки. Вони забезпечують готові компоненти та шаблони, що допомагають зосередитися на логіці та функціоналі додатку.

Фронт-енд розробка є важливою не лише для естетичного вигляду веб-сторінок, але й для покращення загального користувацького досвіду, забезпечення швидкої та безперебійної роботи додатків. Вона грає вирішальну роль у створенні інтуїтивно зрозумілих та привабливих інтерфейсів, що відповідають сучасним стандартам і очікуванням користувачів.

2.1 Bootstrap

Bootstrap - це відкритий та безкоштовний HTML, CSS та JS фреймворк для швидкої верстки адаптивних дизайнів сайтів та WEB-додатків. Найчастіше його використовують для фронтенд розробки сайтів та інтерфейсів адмінпанелей.

Цей фреймворк сильно полегшує розробку сайтів завдяк и наявності шаблонів дизайну на основі HTML і CSS для типографіки, форм, кнопок, таблиць, навігації, плагінів JavaScript та ін. Все це також дає вам можливість легко створювати адаптивні дизайни.

Він є дуже популярним серед розробників тому, що він дозволяє верстати сайти в делькілка разів швидше, ніж на чистому CSS та JS. Він є добре спроектованим фронтенд фреймворком, який досить просто можна налаштувати під себе за допомогою редагування Sass змінних та використання міксинів. Фреймворк Bootstrap являє собою набір CSS та JS файлів.

Адаптивний дизайн: Bootstrap використовує гнучку сіткову систему, що дозволяє створювати веб-сайти, які автоматично підлаштовуються під розміри екранів різних пристроїв. Це досягається за допомогою CSS медіа-запитів.

Готові компоненти: Фреймворк включає безліч готових компонентів, які можна легко додати до проекту. Це значно знижує час розробки і дозволяє створювати складні інтерфейси без необхідності писати багато коду з нуля.

Теми та кастомізація: Bootstrap надає можливість легко налаштовувати зовнішній вигляд сайту за допомогою змінних Sass. Це дозволяє розробникам створювати унікальні дизайни, зберігаючи при цьому переваги фреймворку.

Плагіни JavaScript: Bootstrap включає в себе низку плагінів на основі jQuery, які додають додаткову функціональність, таку як модальні вікна, каруселі, спливаючі підказки та багато іншого. Документація та спільнота: Одна з найбільших переваг Bootstrap – це його детальна документація та велика спільнота розробників. Це робить фреймворк легким у вивченні та використанні.

2.2 Структура HTML документу

Структура HTML документа є фундаментальним аспектом веб-розробки, оскільки саме вона визначає організацію і логіку представлення контенту на веб-сторінці. HTML є мовою розмітки, яка використовується для створення веб-документів. Зрозуміти основи структури HTML документа важливо як для

початківців, так і для досвідчених розробників, оскільки це впливає на доступність, SEO - оптимізацію пошукових систем, та зручність користування веб-сторінкою.

HTML документ починається з оголошення типу документа, або DOCTYPE, що вказує браузеру на версію HTML, яка використовується. Для сучасних веб-сторінок це зазвичай виглядає так: <!DOCTYPE html>. Цей виклик є обов'язковим та забезпечує коректне відображення сторінки в браузерах.

Після оголошення DOCTYPE йде кореневий елемент документа, тег <html>, який обгортає весь вміст веб-сторінки. Він складається з двох основних частин: <head> та <body>.

Основний зміст сторінки знаходиться в секції <body>. Тут розміщуються всі видимі елементи сторінки: заголовки, абзаци, зображення, відео, таблиці, форми та інші мультимедійні елементи. Наприклад, заголовки визначаються тегами з <h1> до <h6>, абзаци тегом <p>, зображення тегом <img>, посилання тегом <a>.

HTML також підтримує семантичні теги, <header>, <footer>, <nav>, <article> та <section>, які додають додаткову інформацію про структуру та зміст документа. Використання семантичних тегів робить код більш читабельним та доступним, як для людей, так і для пошукових систем.

Наприклад, тег <header> використовується для визначення верхньої частини сторінки або розділу, що зазвичай містить заголовок або навігацію, тоді як <footer> визначає нижню частину сторінки, яка може містити контактну інформацію або посилання.

Взагалі, структура HTML документа є дуже важливим аспектом в веб-розробці. Вона визначає, як інформація буде організована та відображена на веб-сторінці. Використання вірної структури забезпечує кращу доступність, зручність користування та ефективність веб-сторінки, що є дуже важливим для успіху будь-якого веб-проекту.

2.3 JavaScript

JS відіграє вирішальну роль у фронтенді, додаючи інтерактивність до статичних HTML і CSS. Основна функція JS полягає у взаємодії з користувачем та динамічній зміні контенту сторінки. Наприклад, за допомогою JS можна створювати виповзаючі меню, слайдери, модальні вікна та інші елементи, які реагують на дії користувача, такі як натискання кнопок, переміщення миші або введення тексту.

JS дозволяє реалізовувати складні обчислення та логіку на стороні клієнта. Це включає в себе валідацію даних, створення анімацій, обробку подій та інше. Наприклад, при заповненні форми, JS може перевіряти правильність введених даних до їх відправки на сервер, що зменшує навантаження на сервер та покращить ефективність обробки даних.

Фреймворки та бібліотеки на основі JavaScript, такі як React, Angular і Vue.js, ще більше розширюють можливості фронт-енд розробки. Вони надають інструменти та методології для створення складних та масштабованих веб-додатків. React, наприклад, дозволяє будувати компоненти, які можна повторно використовувати, що спрощує процес розробки і підтримки коду. Angular надає повний набір інструментів для створення динамічних додатків з чітко визначеною структурою, а Vue.js поєднує в собі найкращі риси обох, надаючи гнучкість та простоту використання

2.4 JQuery

Історія створення jQuery починається у 2006 році, коли Джон Ресіг представив першу версію бібліотеки на конференції BarCamp NYC. З тих пір jQuery швидко набув популярності і став однією з найпопулярніших бібліотек JavaScript. З часом бібліотека продовжувала розвиватися, отримуючи нові

функції і поліпшення, що робило її ще більш потужною і зручною у використанні.

jQuery – це популярна бібліотека JavaScript, створена для спрощення роботи з HTML-документами, обробки подій, анімації та розробки Ajax-додатків. Вона була створена Джоном Ресігом у 2006 році з метою полегшення написання JavaScript-коду. Основна ідея jQuery полягає у використанні простого і зрозумілого синтаксису, що дозволяє швидко і ефективно маніпулювати елементами веб-сторінок.

jQuery значно спрощує доступ до елементів DOM (Document Object Model) та маніпуляції з ними. Завдяки використанню CSS-подібних селекторів, розробники можуть легко вибирати елементи і виконувати з ними різноманітні операції, такі як зміна стилів, додавання або видалення класів, зміна атрибутів та інше. Крім того, jQuery пропонує зручні методи для обробки подій, що дозволяє легко додавати інтерактивність до веб-сторінок. Наприклад, за допомогою jQuery можна легко обробляти події натискання на кнопку, наведення курсора на елемент або зміни значення поля форми.

Однією з ключових функцій jQuery є підтримка Ajax-запитів. Це дозволяє розробникам створювати динамічні веб-додатки, що можуть взаємодіяти з сервером без перезавантаження сторінки. Це значно покращує користувацький досвід і робить веб-додатки більш інтерактивними. Крім того, jQuery підтримує анімацію, що дозволяє розробникам створювати плавні переходи і ефекти, покращуючи візуальну привабливість веб-сторінок.

Переваги jQuery включають його простоту і зручність використання. Завдяки інтуїтивно зрозумілому синтаксису, розробники можуть швидко навчитися використовувати jQuery і ефективно працювати з ним. Бібліотека також забезпечує високу сумісність з різними браузерами, що дозволяє уникнути проблем з кросбраузерністю. Крім того, велика спільнота користувачів і розробників забезпечує наявність великої кількості плагінів і розширень, що розширюють функціональність jQuery.

Однак jQuery має і свої мінуси. З розвитком сучасних стандартів JavaScript, таких як ES6, багато з функцій jQuery стали доступними безпосередньо в самому JavaScript. Це зменшує потребу у використанні бібліотеки, особливо у нових проектах. Крім того, використання jQuery може збільшувати розмір завантажуваного коду, що може впливати на швидкість завантаження сторінок. Деякі розробники також критикують jQuery за те, що він заохочує написання неконцизного коду, що може ускладнити підтримку великих проектів.

Таким чином, jQuery – це потужний інструмент для веб-розробників, що значно спрощує роботу з JavaScript і HTML-документами. Він пропонує простий і зручний синтаксис для маніпуляції DOM, обробки подій, анімації та розробки Ajax-додатків. Хоча сучасні стандарти JavaScript зменшили потребу у використанні jQuery, бібліотека все ще залишається корисним інструментом для багатьох розробників завдяки своїй простоті, сумісності з різними браузерами та великій спільноті користувачів.

3. Огляд існуючих технологій для створення веб-сайтів

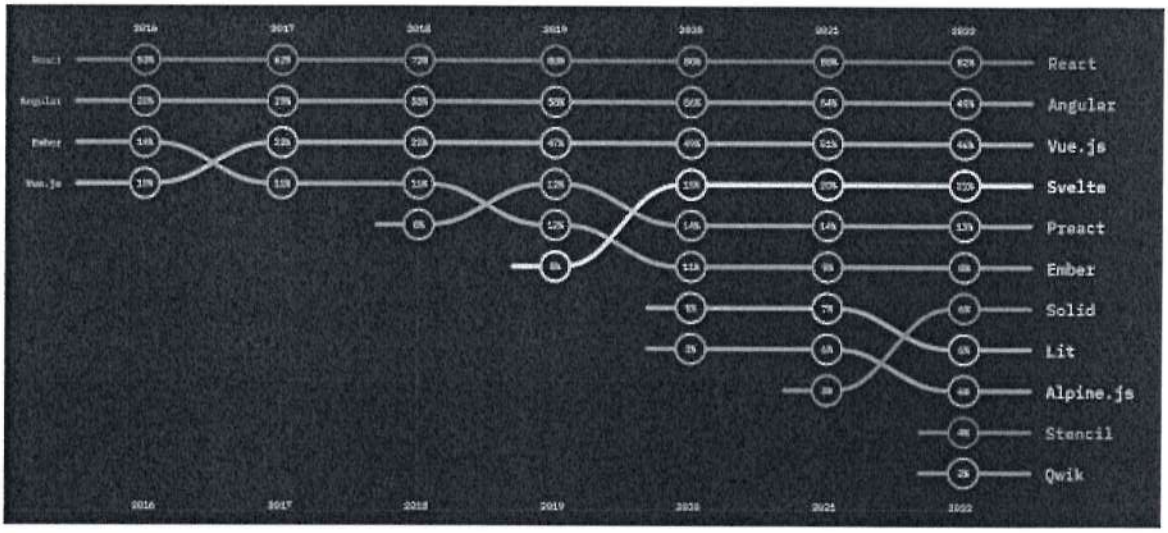

Рис.3.1. Аналітика фреймворків.

На рисунку видно , що react.js , angluar , vue.js є трьома найпопулярнішими фреймворками станом на 2022 рік, а за ними йде svetle та preact. Розглянемо кожен з них.

3.1 React

React – це безкоштовна бібліотека JS, розроблена компанією Meta та активно підтримувана широкою спільнотою розробників. Основним призначенням React є створення інтерфейсу користувача. Він набув великої популярності завдяки своїм функціональним можливостям та простоті використання.

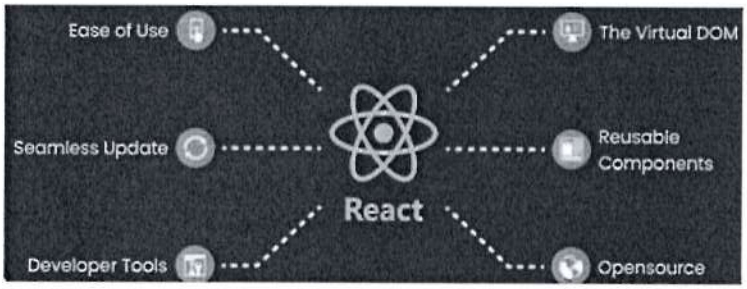

Рис.3.2. Фреймворк.

Однією з головних переваг React є використання віртуального DOM (Document Object Model). Віртуальний DOM є абстракцією реального DOM, який

використовується в браузерах для представлення HTML-документів. Замість безпосередньо взаємодіяти з реальним DOM, React створює віртуальне представлення документа, що зберігається в пам'яті. При зміні стану компонентів

React порівнює віртуальний DOM з реальним DOM та ефективно оновлює тільки змінені елементи. Це дозволяє покращити продуктивність додатка, зменшити затрати на зміни в DOM та покращити відгук користувача. Компонентна архітектура є ще однією ключовою особливістю React.

Додатки React побудовані на компонентах, які є самодостатніми, перевикористовуваними блоками. Кожен компонент має власний стан і може приймати вхідні параметри. Використання компонентів спрощує розробку та підтримку програмного забезпечення, оскільки вони можуть бути використані багаторазово і дотримуються принципу розділення відповідальностей. У React дані передаються зверху вниз, що називається односпрямованим потоком даних. Це означає, що зміни стану компонента відбуваються тільки вгору по ієрархії компонентів, що спрощує відлагодження і виявлення помилок в програмі. Крім того, цей підхід сприяє підтримці чистоти та передбачуваності даних.

Ще однією перевагою React є його невелика крива навчання. Він поєднує базові концепції HTML та JavaScript з додатковими можливостями бібліотеки. Це робить його відносно простим для освоєння, особливо для розробників, які вже знайомі з цими мовами.Крім веб-додатків, React також здатний розробляти мобільні додатки за Допомогою фреймворка React Native. Це дозволяє розробникам використовувати ту ж саму компонентну модель та навички, щоб створювати як веб-додатки, так і мобільні додатки з високою швидкістю та продуктивністю.

Загалом, React - це потужна бібліотека для розробки інтерфейсу користувача, яка забезпечує продуктивну і ефективну розробку динамічних веб-додатків. Він спрощує роботу з DOM, пропонує компонентну архітектуру для побудови програмного забезпечення та підтримує розробку як веб-додатків, так і мобільних додатків.

3.2 Angular

Angular - це популярний фреймворк для розробки веб-додатків, який був розроблений компанією Google. Він базується на мові програмування TypeScript і використовується для створення ефективних та масштабованих веб-додатків з багатофункціональним інтерфейсом користувача. Ось деякі особливості, які роблять Angular таким популярним серед розробників:

- Angular базується на компонентній архітектурі, що означає, що додаток будується з окремих компонентів, які мають свою внутрішню логіку, шаблони та стилі. Це сприяє легкому управлінню кодом, реорганізації та повторному використанню компонентів;
- Angular надає двостороннє зв'язування даних, що означає, що зміни в даних моделі автоматично відображаються в користувацькому інтерфейсі і навпаки. Це дозволяє розробникам легко і зручно управляти даними та їх відображенням;
- Angular має потужну систему директив, яка дозволяє розробникам розширювати HTML і додавати нову функціональність до елементів сторінки. Директиви можуть використовуватись для маніпулювання DOM, створення анімацій, обробки подій та багатьох інших задач;
- Angular пропонує механізм сервісів, який дозволяє створювати розподілені компоненти, які надають певну функціональність для всього додатку. Сервіси можуть використовуватись для обробки бізнес-логіки, отримання даних з сервера, кешування та багатьох інших завдань;
- Angular має потужний модуль маршрутизації, який дозволяє розробникам створювати веб-додатки з багатьма сторінками та перехідами між ними. Це спрощує створення розширюваних та добре організованих додатків з різними роутами;
- Angular надає потужні інструменти для тестування веб-додатків,

включаючи модульні тести, інтеграційні тести та автоматичне тестування. Це дозволяє розробникам впевнено тестувати свій код та забезпечити стабільну та надійну роботу додатку;

Angular і AngularJS (попередня версія Angular) мають деякі схожі особливості, але також відрізняються. AngularJS базується на JavaScript, тоді як Angular використовує TypeScript, що робить його більш потужним та типізованим. Angular також має покращену продуктивність, швидкість та масштабованість, завдяки використанню віртуального DOM та компонентної архітектури. Однією з ключових відмінностей між Angular та AngularJS є спосіб прив'язки даних. AngularJS використовує двосторонню прив'язку даних, тоді як Angular застосовує односторонню прив'язку даних, що сприяє полегшенню управління даними та виявленню помилок. Крім того, Angular має більше вбудованих функцій, покращену маршрутизацію, вбудовану підтримку тестування та широкий набір інструментів для розробки. Він також надійніший та масштабований, що робить його популярним серед розробників для створення великих та складних веб-додатків

3.3 vue.js

vue.js – це прогресивний фреймворк JavaScript для розробки користувацького інтерфейсу. Він використовується для побудови односторінкових додатків і дозволяє створювати ефективні та динамічні веб-інтерфейси.

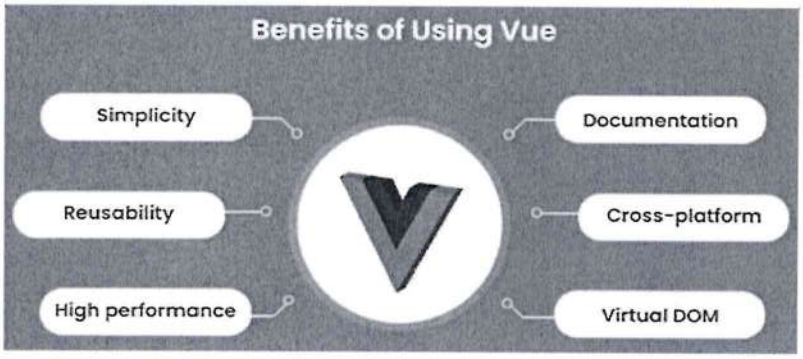

Рис.3.3 Фреймворк.

Основні властивості та особливості vue.js: Декларативний синтаксис: Vue.js пропонує декларативний підхід до розробки, що дозволяє зосередитися на описі того, що потрібно зробити, а не як це зробити. Це включає в себе шаблони, де ви можете описати структуру вашого інтерфейсу користувача, а також директиви, які дозволяють легко взаємодіяти з елементами DOM. Такий підхід робить код більш зрозумілим та легким для розуміння.

Компонентна архітектура: Vue.js побудований на основі компонентної архітектури, де весь інтерфейс користувача розбитий на невеликі, самодостатні та перевикористовувані компоненти. Кожен компонент має свою власну логіку та представлення, що дозволяє легко керувати окремими частинами додатку. Компоненти можна збирати разом, створюючи більш складні інтерфейси

Це спрощує розробку, підтримку та тестування коду, а також забезпечує більшу модульність та розширюваність додатку.

Реактивність: Vue.js має вбудовану систему реактивності, що дозволяє автоматично оновлювати відображення даних при їх зміні. Це означає, що коли дані змінюються, Vue.js автоматично оновлює відповідні частини інтерфейсу користувача без необхідності вручну керувати оновленнями. Це спрощує розробку динамічних та інтерактивних інтерфейсів, забезпечуючи швидку відповідь на дії користувача.

Віртуальний DOM: Vue.js використовує віртуальний DOM для оптимізації процесу оновлення інтерфейсу користувача. Віртуальний DOM є проміжним шаром між реальним DOM та даними додатку. Замість безпосереднього оновлення реального DOM, Vue.js спочатку оновлює віртуальний DOM, а потім порівнює його з реальним DOM та виконує необхідні зміни тільки в тих частинах, де відбулися зміни. Це робить процес оновлення швидшим та ефективнішим.

Широкий набір функціональності: Vue.js надає багатий набір функціональності для розробки веб-додатків. Він має вбудовану підтримку директив, фільтрів, обробників подій, роутингу, стану додатку та багато іншого.

Крім того, Vue.js підтримує плагіни, які дозволяють легко розширювати функціональність фреймворку та використовувати сторонні бібліотеки.

Простота вивчення та використання: Vue.js має дружній та простий для вивчення синтаксис, що робить його легким для початку розробки. Він надає чітку та добре написану документацію з багатою базою знань та прикладами. Крім того, він має активну спільноту розробників, яка допомагає одне одному, надає відповіді на питання та сприяє розвитку фреймворку.

Легка інтеграція: Vue.js може бути легко інтегрований з існуючими проектами, незалежно від того, чи це простий веб-сайт, або складний односторінковий додаток. Він підтримує поступове впровадження, що дозволяє розробникам поступово переходити до використання Vue.js в існуючому проекті без необхідності переписування вже наявного коду.

Висока продуктивність: Завдяки віртуальному DOM та ефективній системі реактивності, Vue.js забезпечує високу продуктивність веб-додатків. Він має швидку швидкість виконання, швидкість оновлення інтерфейсу користувача, а також ефективну роботу з пам'яттю.

Підтримка мобільних додатків: Vue.js також має дочірній фреймворк під назвою Vue Native, який дозволяє розробляти мобільні додатки з використанням Vue.js. Це дозволяє розробникам використовувати свої навички та досвід з Vue.js для створення переносних мобільних додатків для платформ, таких як iOS та Android.

3.4 svelte

Svelte.js - це модерний фреймворк JavaScript, який дозволяє створювати швидкі та ефективні інтерфейси користувача. Він відрізняється від інших фреймворків тим, що здійснює компіляцію компонентів на етапі збірки, що дозволяє генерувати чистий JavaScript-код без необхідності завантажувати додаткову бібліотеку на стороні клієнта.

Svelte.js надає простий та потужний спосіб оголошення змінних, які автоматично спричиняють повторний рендеринг відповідних частин інтерфейсу. Це дозволяє створювати реактивні компоненти без необхідності вручну визначати спостерігачі для даних. Також він пропонує простий та лаконічний синтаксис, що дозволяє швидко створювати компоненти та взаємодіяти з ними. Він має чітку структуру, що спрощує розробку та підтримку проектів. Оскільки Svelte.js генерує оптимізований JavaScript-код, розмір кінцевого додатку зазвичай є значно меншим порівняно з іншими фреймворками. Це дозволяє швидше завантаження сторінки та зменшує використання ресурсів на стороні клієнта.Svelte.js підтримує повторне використання компонентів, що дозволяє створювати бібліотеки компонентів та використовувати їх в різних проектах. Це полегшує розробку та забезпечує консистентність інтерфейсу. Завдяки своїй компіляції на етапі збірки та ефективному управлінню реактивністю, Svelte.js забезпечує високу швидкість та продуктивність додатків. Це особливо важливо для великих проектів з багатою функціональністю.

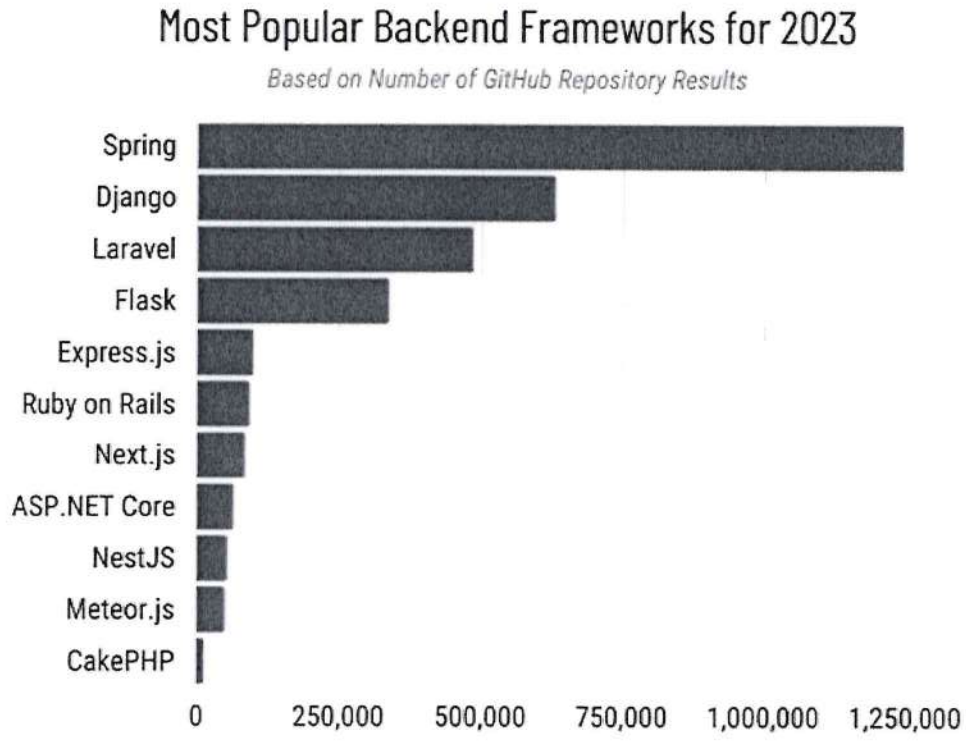

Рис.3.4. Аналітика фреймворків.

3.5 Spring

Spring Framework є одним з найпопулярніших фреймворків розробки додатків на мові Java завдяки своїм можливостям, простоті використання та широкому спектру модулів. Він дозволяє розробникам ефективно будувати робустні, масштабовані та безпечні додатки з використанням сучасних підходів та кращих практик.

Spring Framework підтримує архітектуру Model-View-Controller (MVC), яка дозволяє логічно розділити компоненти додатку на модель (Model), представлення (View) та контролер (Controller). Ось детальніше про ці компоненти та їх роль в Spring Framework:

Модель (Model): Модель представляє дані або стан додатку. В Spring Framework модель може бути представлена класами Java або об'єктами, які зберігають дані. Ці дані можуть бути передані в представлення для відображення користувачу або використані в бізнес-логіці додатку.

Представлення (View): Представлення відповідає за відображення даних користувачу. У Spring Framework представлення може бути HTML-шаблоном, JSP-сторінкою, Thymeleaf-шаблоном або будь-яким іншим форматом відображення. Представлення отримує дані з моделі і відображає їх для користувача.

Контролер (Controller): Контролер є посередником між моделлю та представленням. Він обробляє запити від користувача, взаємодіє з моделлю для отримання або оновлення даних, і передає ці дані відповідному представленню для відображення. У Spring Framework контролери можуть бути представлені як класи Java, анотовані спеціальними анотаціями, які вказують, які запити вони обробляють.

Spring Framework також надає можливість організувати додаток у вигляді модулів. Модулі дозволяють логічно розділити функціональність додатку на незалежні частини. Кожен модуль може мати свої контролери, моделі та представлення, які співпрацюють з іншими модулями. Це полегшує розробку та

підтримку великих та складних додатків, оскільки розробники можуть працювати над окремими модулями незалежно один від одного. Модулі в Spring Framework можуть бути організовані за допомогою модульних конфігурацій тазалежностей. Кожен модуль може мати свою конфігурацію, яка визначає біни,аспекти та інші компоненти, використовувані в рамках модуля. Залежності між модулями можуть бути вказані за допомогою інструментів інверсії керування (IoC), таких як Dependency Injection (DI), що дозволяють модулям взаємодіяти між собою. Такий підхід до модульної організації та архітектури додатків робить Spring Framework гнучким та легко розширюваним фреймворком для розробки різноманітних додатків на мові Java.

Інверсія керування (Inversion of Control - IoC) та внедрення залежностей (Dependency Injection - DI) є важливими концепціями у Spring Framework, які сприяють створенню гнучких, масштабованих та легко тестуємих додатків. Інверсія керування (IoC) означає, що контроль над створенням та управлінням об'єктами передається контейнеру (такому як контейнер Spring), а не самому додатку. Замість того, щоб додаток самостійно створював та управляв залежностями, він описує свої потреби у вигляді специфікацій або метаданих, які контейнер використовує для створення та налагодження об'єктів.

Впровадження залежностей (Dependency Injection - DI) є одним з способів реалізації IoC. Воно використовується для впровадження залежностей між об'єктами, де залежності передаються об'єкту зовні. Це робиться через конструктор, методи або інші механізми, що дозволяють контейнеру впровадити залежності без прямого знання про конкретні реалізації залежностей.

Spring Framework надає розширений механізм впровадження залежностей (DI), який дозволяє описати залежності за допомогою конфігураційних файлів або анотацій. Він підтримує конструкторний, методичний та полеовий DI. Це дозволяє легко змінювати залежності, використовувати різні реалізації та спрощувати тестування через заміну залежностей на мок-об'єкти або інші замінники.

Аспекти та аспектно-орієнтоване програмування (Aspect-Oriented Programming - AOP) є ще однією важливою функцією у Spring Framework. AOP дозволяє розділити логіку, яка перетинається з багатьма частинами додатку (наприклад, логування, транзакції, безпека) у окремі компоненти, відомі як аспекти. Аспекти відокремлюються від основної логіки програми та виконуються на основі певних правил, які називаються зрізи (pointcuts).

Spring Framework надає потужний механізм аспектно-орієнтованого програмування, який дозволяє описувати аспекти та визначати зрізи за допомогою анотацій або XML-конфігурацій. AOP дозволяє впроваджувати додаткову функціональність у додаток, таку як логування, транзакції або безпека, з мінімальним впливом на основну логіку програми.

3.6 Express.js

Express.js є популярним фреймворком для розробки серверних додатків на мові JavaScript. Він базується на Node.js і дозволяє створювати швидкі, масштабовані та надійні веб-додатки.

Основні риси Express.js: Легкість використання: Express.js пропонує простий та інтуїтивно зрозумілий інтерфейс, який дозволяє швидко розпочати розробку веб-додатків. Він має мінімальну кількість обов'язкових правил та конфігурацій, що дозволяє розробникам швидко приступити до написання коду.

Маршрутизація: Express.js надає потужні засоби для визначення маршрутів та обробки запитів. Ви можете визначити шляхи (routes) для різних URL-адрес, ініціювати різні дії при отриманні запитів GET, POST, PUT або DELETE, та виконувати обробку запитів за допомогою middleware-функцій.

Middleware: Middleware - це функції, які виконуються перед або після обробки запиту. Вони дозволяють виконувати різноманітні операції, такі як обробка даних, аутентифікація, авторизація, реєстрація журналу, компресія

даних та багато іншого. Express.js дозволяє легко використовувати і налаштовувати middleware для ваших додатків.

Шаблонізація: Express.js підтримує різні шаблонізатори (наприклад, EJS, Pug, Handlebars), які дозволяють генерувати HTML-сторінки на основі даних. Шаблонізація дозволяє вам створювати динамічний контент та повторно використовувати код.

Розширюваність: Express.js дозволяє вам розширювати його функціональність за допомогою пакетів (middleware або додаткових модулів). Ви можете використовувати пакети, такі як Passport.js для аутентифікації, Socket.io для реал-тайм комунікації, Multer для обробки файлів, та багато інших. Статичні файли: Express.js дозволяє зручно обслуговувати статичні файли, такі як CSS, JavaScript або зображення, завантажені на сервер. Ви можете налаштувати шлях до статичних файлів та використовувати їх у своїх сторінках. Підтримка REST API: Express.js є чудовим інструментом для розробки REST API. Ви можете визначити різні маршрути для обробки запитів POST, GET, PUT та DELETE, та повертати дані у форматі JSON.

4. Бек-енд розробка

Бекенд – це частина веб-додатку або веб-сайту, яка відповідає за обробку даних, логіку бізнес-процесів, аутентифікацію та авторизацію користувачів, а також взаємодію з базами даних і зовнішніми сервісами. Іншими словами, бекенд – це серверна сторона додатку, яка обробляє запити від користувача і повертає необхідні дані або виконує певні дії. Якщо фронтенд – це те, що бачить і з чим взаємодіє користувач, то бекенд – це те, що працює за кулісами для забезпечення функціональності та стабільності додатку.

Для розробки бекенду використовуються різні технології та мови програмування. Най популярні з них це :

- Node.js. Це серверне середовище, яке дозволяє запускати JavaScript на сервері. Node.js відомий своєю високою продуктивністю та асинхронною природою, яка робить його ідеальним для обробки великої кількості одночасних запитів.

- Python. Завдяки своїй простоті і читабельності, Python став дуже популярним для бекенд розробки. Фреймворки, такі як Django і Flask, забезпечують зручний та ефективний спосіб створення серверних додатків.

- Java. Це одна з найпоширеніших мов програмування для бекенду, особливо в корпоративному середовищі. Java відома своєю стабільністю, масштабованістю та безпекою. Spring Framework є одним з найпопулярніших інструментів для розробки на Java.

- PHP: Хоча PHP інколи критикують за його менш строгі стандарти, він залишається популярним вибором для бекенд розробки, особливо для створення контент-орієнтованих веб-сайтів. Фреймворки, такі як Laravel, значно спрощують розробку на PHP.

Архітектура бекенд систем грає ключову роль у забезпеченні надійності, маштабування та ефективності веб-сайтів. Однією з основних концепцій у бекенд розробці є серверно-клієнтська модель, де сервер обробляє запити від клієнтів (фронтенд) і повертає відповідні відповіді. Однак сучасні веб-додатки

часто потребують більш складної архітектури, щоб впоратися з великим обсягом даних і забезпечити високу продуктивність та стійкість.

Одним з підходів до архітектури бекенд систем є монолітна архітектура. У монолітних додатках весь код і функціональність об'єднані в єдину, нероздільну програму. Це простіше для розробки та розгортання на ранніх стадіях, оскільки весь додаток розгортається як одне ціле. Однак монолітна архітектура може стати проблематичною при масштабуванні або внесенні змін. Коли додаток зростає, стає важко впроваджувати нові функції або виправляти помилки без впливу на всю систему. Велика база коду може ускладнювати роботу над проектом для великих команд розробників і знижувати гнучкість.

На противагу монолітній архітектурі, мікросервісна архітектура розділяє додаток на набір невеликих, незалежних сервісів, кожен з яких відповідає за певну функціональність. Це дозволяє легше масштабувати окремі компоненти, спрощує оновлення і обслуговування, а також підвищує стійкість додатку. Мікросервіси часто взаємодіють через API (Application Programming Interface) і можуть бути розгорнуті незалежно один від одного, що значно знижує ризики при оновленнях та дозволяє легко впроваджувати нові технології. Така архітектура полегшує інтеграцію з іншими системами та підтримує різні мови програмування та технології для різних сервісів, що дозволяє вибирати найбільш відповідні інструменти для кожного завдання.

Крім цього, важливим аспектом є вибір бази даних. Реляційні бази даних, такі як MySQL і PostgreSQL, добре підходять для додатків з чітко структурованими даними і складними запитами. Вони підтримують транзакції, забезпечуючи цілісність даних і дозволяючи виконувати складні запити з об'єднанням декількох таблиць. Однак для деяких додатків реляційні бази даних можуть виявитися менш ефективними через свою структуру.

Нереляційні бази даних (NoSQL), такі як MongoDB і Redis, забезпечують гнучкість і високу продуктивність для великих обсягів неструктурованих або частково структурованих даних. Вони краще підходять для додатків, які потребують швидкого запису і читання даних або мають складну і змінну

структуру. MongoDB, наприклад, зберігає дані в форматі JSON-подібних документів, що дозволяє зберігати складні структури даних. Redis використовується для зберігання даних у вигляді ключ-значення і часто застосовується для кешування, щоб забезпечити швидкий доступ до часто використовуваних даних.

Архітектура бекенд систем також включає питання безпеки, надійності і ефективності. Захист даних користувачів і запобігання несанкціонованому доступу є критично важливими аспектами бекенд розробки. Це включає використання SSL/TLS для шифрування даних, автентифікацію і авторизацію користувачів, захист від атак типу SQL-ін'єкцій і XSS (Cross-Site Scripting), а також впровадження механізмів резервного копіювання і відновлення даних.

Бекенд розробка є ключовою частиною створення функціональних і надійних веб-додатків. Розуміння основних понять і принципів бекенду, використання відповідних технологій і мов програмування, а такожправильний вибір архітектури системи є критичними для успіху будь-якого веб-проекту.

5. Бази даних

База даних - це структуроване сховище даних, яке дозволяє ефективно зберігати, змінювати та вилучати інформацію. Вони організовані таким чином, щоб забезпечити легкий доступ до даних і управління ними. Бази даних можуть бути різних типів, включаючи реляційні, нереляційні (NoSQL), графові та часові.

- Реляційні бази даних (RDBMS). Ці бази даних використовують таблиці для зберігання даних і забезпечують потужні механізми для управління відносинами між цими таблицями. MySQL, PostgreSQL, Oracle Database.
- Нереляційні бази даних (NoSQL). Ці бази даних не використовують таблиці та забезпечують гнучкіші моделі зберігання даних, які підходять для роботи з великими обсягами нереляційних даних. MongoDB, Cassandra, Redis.
- Графові бази даних. Використовують графи для моделювання даних та є ефективними для роботи з даними, що мають складні відносини. Neo4j, OrientDB.
- Часові бази даних. Призначені для зберігання часових даних та аналізу даних у часових проміжках. InfluxDB, TimescaleDB.

5.1 Використання баз даних

- Зберігання даних. Основне призначення бази даних - це зберігання великих обсягів інформації в структурованому вигляді. Це дозволяє ефективно зберігати дані, забезпечуючи їхню цілісність та доступність.
- Обробка транзакцій. Бази даних дозволяють виконувати транзакції, які є важливими для фінансових додатків та інших систем, де необхідно забезпечити цілісність даних.
- Аналітика та звітність. Бази даних використовуються для аналізу даних та створення звітів, що допомагає організаціям приймати обґрунтовані рішення на основі аналізу даних.

• Інтеграція даних. Бази даних дозволяють інтегрувати дані з різних джерел, забезпечуючи єдиний погляд на інформацію. Це особливо корисно для великих корпорацій, які працюють з різними системами.

• Висока доступність та відновлення після збоїв. Бази даних повинні бути надійними та забезпечувати безперервний доступ до даних. Вони підтримують механізми резервного копіювання та відновлення даних.

5.2 MySQL

MySQL — це одна з найпопулярніших систем управління реляційними базами даних (СУБД), яка використовує мову програмування SQL для роботи з даними. MySQL була створена шведською компанією MySQL AB у середині 1990-х років і стала важливим елементом у світі розробки програмного забезпечення, особливо у веб-розробці.

Основна мета MySQL полягає у забезпеченні ефективного зберігання, обробки та отримання даних. MySQL використовується у багатьох великих веб-додатках і платформах, таких як Facebook, Twitter, YouTube і WordPress, завдяки її продуктивності, надійності та простоті використання. Відкритий вихідний код MySQL дозволяє розробникам вільно використовувати, змінювати та розповсюджувати її, що зробило її популярною як серед малих стартапів, так і серед великих корпорацій.

Однією з головних переваг MySQL є її висока продуктивність і швидкість роботи. MySQL оптимізована для швидкого читання даних, що робить її ідеальною для застосувань, де швидкість доступу до інформації є критично важливою. MySQL також підтримує різні типи таблиць, такі як InnoDB і MyISAM, що дозволяє налаштувати базу даних відповідно до специфічних вимог проєкту. InnoDB, наприклад, підтримує транзакції і забезпечує цілісність даних, тоді як MyISAM пропонує швидший доступ до даних у випадках, коли транзакції не є необхідними.

MySQL забезпечує високу надійність і стабільність, що є критично важливим для будь-якої СУБД. Вона підтримує реплікацію даних, що дозволяє створювати резервні копії і відновлювати дані у випадку збою. Це також дозволяє розподіляти навантаження на кілька серверів, що підвищує продуктивність і доступність системи.

Простота використання і налаштування є ще однією вагомою перевагою MySQL. Вона має зрозумілий і логічний синтаксис, який дозволяє розробникам швидко освоїтися з її функціоналом. Багато популярних інструментів і платформ, такі як PHPMyAdmin, забезпечують зручний інтерфейс для роботи з MySQL, що робить управління базами даних доступним навіть для користувачів без глибоких технічних знань.

Історія MySQL почалася у 1995 році, коли Майкл Віденіус, Давид Аксмарк і Алан Ларсон розробили її як альтернативу для інших СУБД. У 2008 році компанія Sun Microsystems придбала MySQL AB, а у 2010 році Sun Microsystems була придбана Oracle Corporation, яка продовжує розвивати і підтримувати MySQL до сьогодні.

Приклад таблиці на мові sql

```
CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `name` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--
-- Дамп данных таблицы `user`
--

INSERT INTO `user` (`id`, `name`, `email`, `password`, `role`) VALUES
(4, 'Test Testov', 'test@gmail.com', '123456', 'admin');
```

6. Архітектура коду MVC

MVC – це архітектурний шаблон для розробки програмного забезпечення, який допомагає розділити застосунок на три основних частини. При розробці систем з інтерфейсом користувача, слідуючи патерну MVC потрібно розділяти систему на три частини. Їх можна назвати модулями чи компонентами.

Архітектура MVC дозволяє нам розділити код програми на 3 частини: Модель (Model), Вид або Подання (View) та Контролер (Controller). Вперше вона була описана в 1978 році, і призначалася для програм з графічним інтерфейсом (віконцями і кнопками), але пізніше була адаптована і для веб-додатків.

Кожна складова має своє призначення. Перший компонент Model. Він містить всю бізнес-логіку програми. Другий компонент це View. Цей компонент відповідає за відображення даних користувачу. Усе, що бачить користувач, генерується цим компонентом. Третій компонент Controller. В ньому зберігається код який відповідає за обробку дій користувача.

Поділ на частини не означає, що в коді має бути рівно 3 файли (або 3 папки з файлами, або 3 класи) з назвами model, view і controller. MVC нічого не говорить нам про те, як організовувати файли з кодом. На практиці модель часто займає основний обсяг програми, і представлена у вигляді великої кількості різнотипних класів - сутностей, сервісів, класів роботи з БД, і для кожного виду класів роблять окремі папки

Будь які дії користувача в системі обробляється в контролері. Модель незалежна частина системи. Настільки незалежна що вона не повинна нічого знати про 2 других компонента Вид та Контролер .

Основне призначення Виду - надавати інформацію з Моделі у зручному для сприйняття користувача форматі. Основне обмеження Виду — він не повинен змінювати модель. Основне призначення Контролера – обробляти дії користувача. Саме через Контролер користувач вносить зміни до моделі. Точніше дані, які зберігаються в моделі

Модель містить у собі всю логіку програми, вона зберігає та обробляє дані, при цьому не взаємодіючи з користувачем безпосередньо (звернутися до Моделі

можна тільки з коду, викликаючи її функції). Наприклад, збереження інформації в БД, перевірка правильності введених у форму даних - це завдання Моделі, але отримання цих даних від користувача або виведення інформації на екран або обробка натискання на кнопку, ні.

Наприклад, у програмі PHP Модель не повинна звертатися до зовнішніх змінних на кшталт ($_GET)($_POST)($_SESSION)($_COOKIE) і не повинна нічого виводити через (echo).

Model це не один клас чи набір однотипних класів. Це основна частина програми, яка може містити багато різних класів: послуги, класи для взаємодії з БД, сутності, валідатори.

View відображає дані, які він передав. У веб-додатку воно зазвичай складається з HTML-шаблонів сторінок, у десктопних або мобільних додатках. View - це код, який відповідає за відображення інформації на екрані, відображення кнопок та інших елементів інтерфейсу.

Controller відповідає за виконання запитів, що надійдуть від користувача. У веб-застосунку зазвичай контролер розбирає параметри HTTP-запиту з ($_POST) ($_GET), звертається до моделі, щоб отримати або змінити якісь дані, і в кінці викликає View, щоб відобразити результат виконання запиту. Число контролерів визначається кількістю розділів або сторінок сайту. У десктопних програмах Контролер відповідає за обробку натискань на кнопки та інших впливів від користувача.

У веб-додатку зазвичай Controller - це набір однотипних класів, кожному розділу на сайті відповідає свій клас, і в ньому робляться методи (їх називають "дії", "action") для окремих сторінок (наприклад: для розділу новин - клас NewsController, в ньому методи latestAction - виведення сторінки останніх новин, archiveAction - сторінка архіву новин, viewAction - сторінка перегляду однієї новини). Тут створюється деяка плутанина, від того що клас називається NewsController ("контролер новин"), але фактично містить у собі не один, а кілька методів-контролерів для окремих сторінок. Іноді роблять і по-іншому - на

кожну сторінку свій клас з однією дією, але на практиці зручніше групувати дії разом.

Весь функціонал програми міститься у Model, Controller і View надають лише можливість користувачеві взаємодіяти з моделлю та відображати дані з неї.

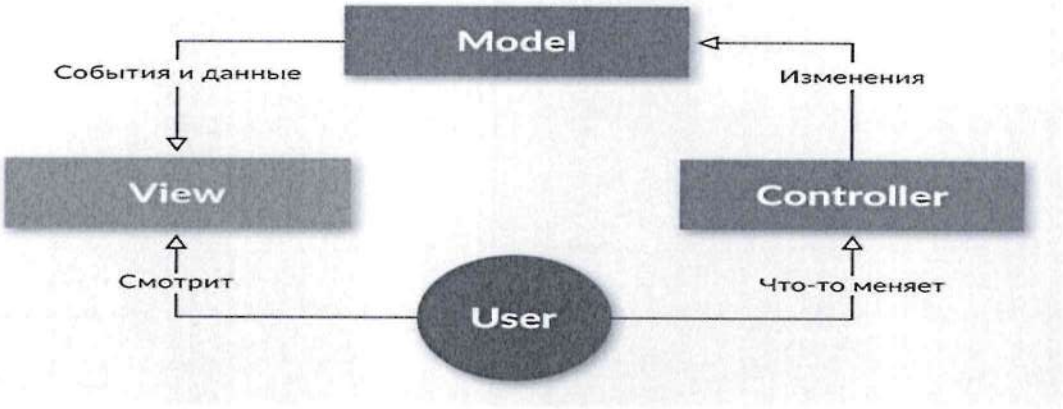

Рис.6.1. Схема MVC паттерну.

6.1 Наглядний приклад роботи MVC

```php
/**
 * Action для страницы "Добавить категорию"
 */
public function actionCreate()
{
    // Проверка доступа
    self::checkAdmin();

    // Обработка формы
    if (isset($_POST['submit'])) {
        // Если форма отправлена
        // Получаем данные из формы
        $name = $_POST['name'];
        $sortOrder = $_POST['sort_order'];
        $status = $_POST['status'];

        // Флаг ошибок в форме
        $errors = false;

        // При необходимости можно валидировать значения нужным образом
        if (!isset($name) || empty($name)) {
            $errors[] = 'Заполните поля';
        }


        if ($errors == false) {
            // Если ошибок нет
            // Добавляем новую категорию
            Category::createCategory($name, $sortOrder, $status);

            // Перенаправляем пользователя на страницу управлениями категориями
            header("Location: /admin/category");
        }
    }

    require_once(ROOT . '/views/admin_category/create.php');
    return true;
}
```

Рис.6.2. Код додавання категорій.

Робота модуля контролер для додавання категорії в адмін панелі. Цей код перевіряє , чи має користувач права адміністратора, перевіряє чи форма була відправлена, як що так то отримує значення полів з форми. Створює прапор для відслідковування помилок , перевіряє чи поле name не порожнє . як що порожнє, додає помилку. Як що помилок немає додає нову категорію. Та нарешті перенаправляє користувача на сторінку управління категоріями.

```php
<?php include ROOT . '/views/layouts/header_admin.php'; ?>

<section>
    <div class="container">
        <div class="row">

            <br/>

            <div class="breadcrumbs">
                <ol class="breadcrumb">
                    <li><a href="/admin">Админпанель</a></li>
                    <li><a href="/admin/order">Управление категориями</a></li>
                    <li class="active">Добавить категорию</li>
                </ol>
            </div>


            <h4>Добавить новую категорию</h4>

            <br/>

            <?php if (isset($errors) && is_array($errors)): ?>
                <ul>
                    <?php foreach ($errors as $error): ?>
                        <li> - <?php echo $error; ?></li>
                    <?php endforeach; ?>
                </ul>
            <?php endif; ?>

            <div class="col-lg-4">
                <div class="login-form">
                    <form action="#" method="post">

                        <p>Название</p>
                        <input type="text" name="name" placeholder="" value="">

                        <p>Порядковый номер</p>
                        <input type="text" name="sort_order" placeholder="" value="">

                        <p>Статус</p>
                        <select name="status">
                            <option value="1" selected="selected">Отображается</option>
                            <option value="0">Скрыта</option>
                        </select>

                        <br><br>

                        <input type="submit" name="submit" class="btn btn-default" value="Сохранить">
                    </form>
                </div>
            </div>
```

Рис.6.3. Код додавання категорій.

Отже, цей код відображає форму для додавання нової категорії в адміністративній панелі, включаючи необхідні елементи, такі як заголовок, форма введення даних і відображення помилок, якщо такі є.

7. Огляд розробленого сайту

Рис.7.1. Головна сторінка.

Перша сторінка яку бачить користувач це головна сторінка. На цій сторінці користувач бачить всю панель навігації вкладки: магазин, про магазин, контакти, каталог товарів, кошик, та вхід в особистий кабінет.

Рис.7.2. Сторінка авторизації

Сторінка авторизації. За допомогою цієї сторінки користувач заходить у свій персональний кабінет.

Рис.7.3. Сторінка реєстрації

Сторінка реєстрації веб-сайту , на цій сторінки користувач може створити персональний обліковий запис який потім буде використовувати для придбання товарі для дому.

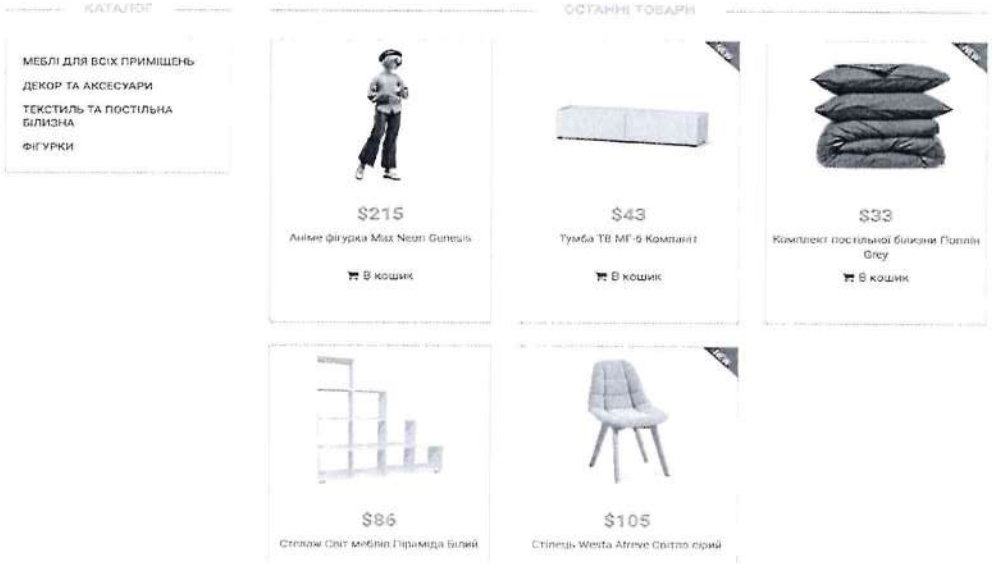

Рис.7.4. Каталог товарів.

Сторінка каталог товарів на цій сторінці користувач має можливість переглядати доступні йому товари для покупки. А також може перейти по певній категорії в лівому меню щоб обрати те, що йому цікаво.

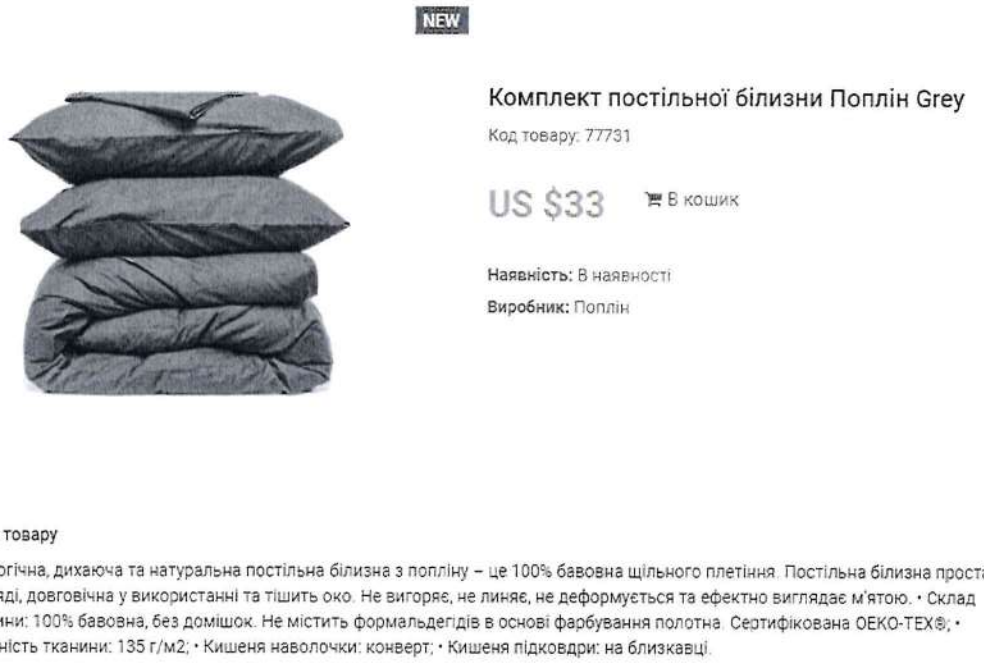

Рис.7.5. Сторінка товару.

Сторінка товару. На цій сторінки користувач може переглянити фотографію продукту, а також побачити чи є даний продукт в наявності, його виробника та детальний опис товару.

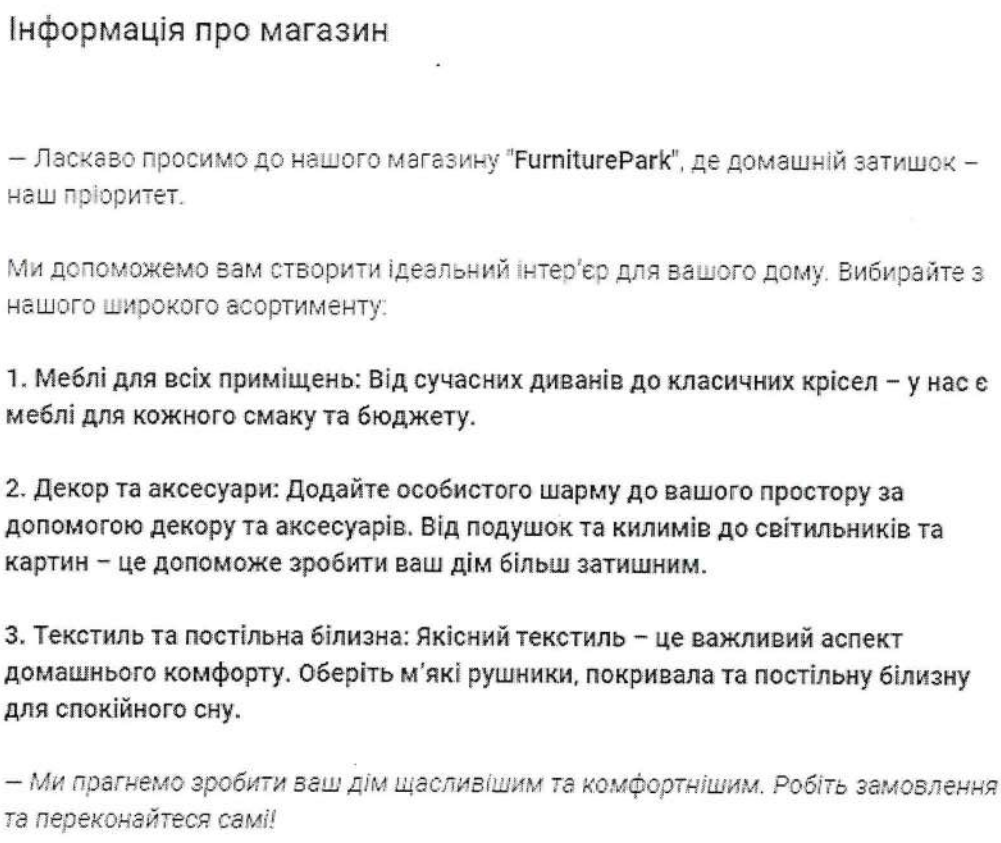

Рис.7.6. Інформація про магазин.

Сторінка про магазин тут користувач може побачити всю інформацію про магазин . Та посміхнутися прочитавши останні слова.

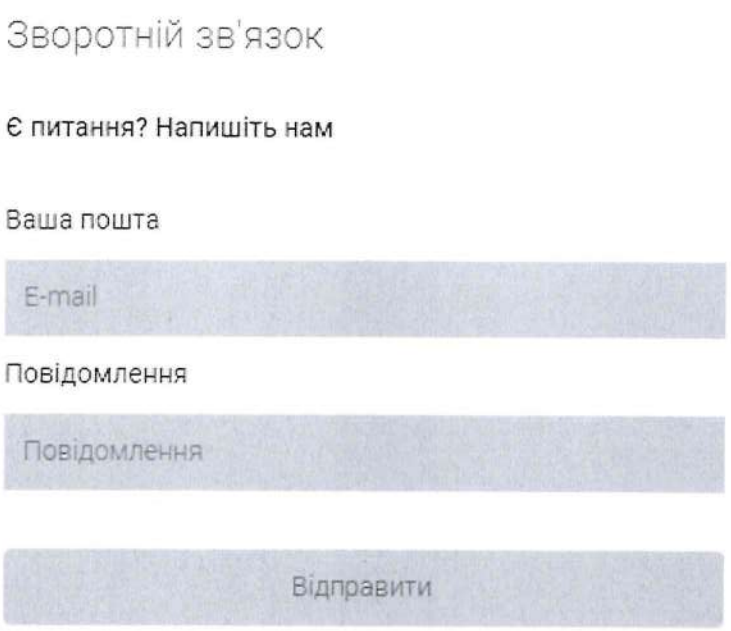

Рис.7.7. Зворотній зв'язок.

Це сторінка зворотнього звязку на якій користувач може залишити повідомлення що до його проблеми або щось на подобі цього та отримати відповідь від адміна.

Рис.7.8. Кошик

Сторінка кошику , тут користувач може переглянути товар який він хоче придбати , побачити ціну та оформити саме замовлення.

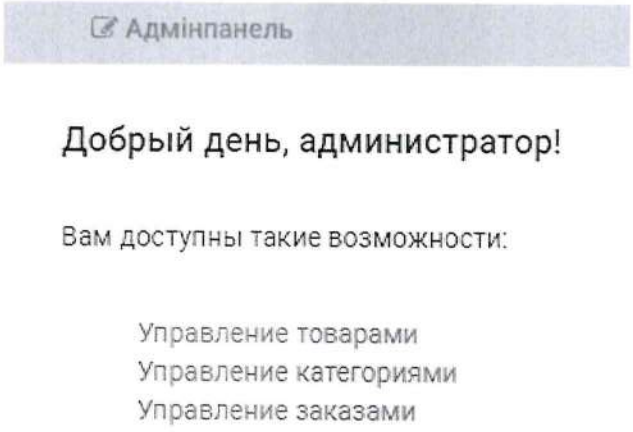

Рис.7.9. Панель адмнінстратора

Сторінка адмін панелі на якій адміністратор сайту може редагувати товар, добавляти чи видаляти його. Також можна керувати категоріями а саме додавати чи видаляти товар.

Рис.7.10. Керування товаром.

Рис.7.11. Керування категоріями.

Висновок

У процесі розробки та створення сайту інтернет-магазину товарів для дому було виконано низку важливих етапів. На початковому етапі було проведено детальний аналіз існуючих рішень, що дозволило визначити кращі практики і уникнути поширених помилок. Після цього було здійснено вибір відповідних технологій та інструментів для реалізації проекту. Для розробки сайту використовувалися сучасні мови програмування, зокрема JavaScript, PHP, CSS та HTML, що забезпечило гнучкість і масштабованість майбутнього продукту.

Основною метою розробки було створення зручного та функціонального інтернет-магазину. Сайт має інтуїтивно зрозумілий інтерфейс користувача, який дозволяє легко орієнтуватися і швидко знаходити потрібні товари. Каталог товарів структурований за категоріями, що полегшує пошук і вибір продукції. Зручна система пошуку та сортування забезпечує швидке знаходження товарів за різними критеріями, такими як ціна, бренд або рейтинг.

Користувачі мають можливість переглядати детальні описи товарів, включаючи фотографії, інформацію про наявність, виробника та детальні характеристики. Сторінка кожного продукту містить відгуки та рейтинги інших користувачів, що допомагає приймати обґрунтовані рішення про покупку. Крім того, сайт дозволяє створювати особистий кабінет, де користувачі можуть зберігати історію замовлень, зберігати улюблені товари та керувати своїм профілем.

Важливою частиною сайту є кошик для покупок, який забезпечує зручний перегляд і редагування товарів перед покупкою. Процес оформлення замовлення простий і зрозумілий, що мінімізує кількість кроків, необхідних для завершення покупки.

Для адміністраторів сайту була розроблена панель управління, яка дозволяє легко керувати товарним асортиментом, редагувати існуючі товари, додавати нові та видаляти ті, що вже не актуальні. Крім того, адміністратори можуть управляти категоріями товарів, що забезпечує гнучкість у структурі каталогу.

Сайт відповідає сучасним вимогам до веб-розробки, має привабливий дизайн і забезпечує всі основні функції, необхідні для успішного функціонування інтернет-магазину.

Однак, для підвищення конкурентоспроможності та покращення користувацького досвіду було виявлено декілька недоліків. Зокрема, сайт потребує додаткових фільтрів пошуку, щоб користувачі могли більш точно знаходити потрібні товари. Також було зазначено, що інформація про товари на сторінках продуктів обмежена і потребує розширення. Крім того, відсутній онлайн-чат для швидкого зв'язку з підтримкою, що є важливим елементом для оперативного вирішення питань користувачів.

На основі проведеного аналізу було встановлено, що веб-сайт є достатньо конкурентоспроможним, але потребує подальшого вдосконалення.

Рекомендується додати функцію онлайн-чату для забезпечення швидкої підтримки користувачів, розширити опис товарів на сторінках продуктів та впровадити додаткові фільтри пошуку. Це допоможе підвищити зручність користування сайтом і покращити загальний користувацький досвід.

Загалом, розроблений сайт є функціональним, зручним для користувачів та відповідає сучасним стандартам веб-розробки, але потребує деяких вдосконалень для підвищення його конкурентоспроможності та покращення користувацького досвіду.

Література

Веденєєв М. І. Веб-розробка для початківців: Практичний посібник. – Київ: Видавництво "Освіта", 2018. – 320 с.

Дейвіс Дж. Bootstrap. Сучасна фронтенд-розробка: Посібник для початківців. – Харків: Видавництво "Промінь", 2019. – 288 с.

Сміт Р. React для професіоналів: Вивчення компонентної архітектури. – Львів: Видавництво "Альфа", 2020. – 352 с.

Андерсон К. JavaScript та jQuery: Розробка інтерактивних веб-сайтів. – Одеса: Видавництво "Чорноморська хвиля", 2019. – 296 с.

Павленко О. І. PHP та MySQL: Створення динамічних веб-додатків. – Дніпро: Видавництво "Перлина", 2018. – 378 с.

Стюарт Б. HTML5 та CSS3: Покроковий посібник для розробників. – Київ: Видавництво "Техніка", 2017. – 310 с.

Ченг А. Основи веб-дизайну: Від ідеї до реалізації. – Запоріжжя: Видавництво "Методика", 2019. – 260 с.

Григоренко М. В. Інтернет-магазини: Теорія та практика розробки. – Харків: Видавництво "Фенікс", 2020. – 400 с.

Вудс Дж. Введення в Angular: Професійна розробка веб-додатків. – Вінниця: Видавництво "Поділля", 2018. – 284 с.

Кравченко І. О. Сучасні бази даних: Реляційні та NoSQL. – Київ: Видавництво "Наука", 2021. – 330 с.

Галієв А. І. Інтернет-комерція: Стратегії та інструменти успіху. – Донецьк: Видавництво "Східний світ", 2020. – 305 с.

Додаток А

Код програми на мові розмітки HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="">
    <title>Головна</title>
    <link href="/template/css/bootstrap.min.css" rel="stylesheet">
    <link href="/template/css/font-awesome.min.css" rel="stylesheet">
    <link href="/template/css/prettyPhoto.css" rel="stylesheet">
    <link href="/template/css/price-range.css" rel="stylesheet">
    <link href="/template/css/animate.css" rel="stylesheet">
    <link href="/template/css/main.css" rel="stylesheet">
    <link href="/template/css/responsive.css" rel="stylesheet">
  </head><!--/head-->

  <body>
    <div class="page-wrapper">

      <header id="header"><!--header-->
        <div class="header_top"><!--header_top-->
          <div class="container">
            <div class="row">
```

```
                    <div class="col-sm-6">
                        <div class="contactinfo">
                            <ul class="nav nav-pills">
                                <li><a href="#"><i class="fa fa-phone"></i> +38 093 000 11
22</a></li>
                                <li><a href="#"><i class="fa fa-envelope"></i>
test@gmail.com</a></li>
                            </ul>
                        </div>
                    </div>
                    <div class="col-sm-6">
                        <div class="social-icons pull-right">
                            <ul class="nav navbar-nav">
                                <li><a
href="https://youtu.be/3UqzqOjdLr4?si=0uS8YR1mv8a3KTNh"><i class="fa fa-
facebook"></i></a></li>
                                <li><a
href="https://youtu.be/3UqzqOjdLr4?si=0uS8YR1mv8a3KTNh"><i class="fa fa-
google-plus"></i></a></li>
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </div><!--/header_top-->

        <div class="header-middle"><!--header-middle-->
```

```
            <div class="container">
              <div class="row">
                <div class="col-sm-4">
                  <div class="logo pull-left">
                    <a href="/"><img src="/template/images/home/logo.png"
alt="" /></a>
                  </div>
                </div>
                <div class="col-sm-8">
                  <div class="shop-menu pull-right">
                    <ul class="nav navbar-nav">
                      <li><a href="/cart">
                          <i class="fa fa-shopping-cart"></i> Кошик
                          (<span id="cart-count"><?php echo Cart::countItems();
?></span>)
                        </a>
                      </li>
                      <?php if (User::isGuest()): ?>
                        <li><a href="/user/login/"><i class="fa fa-lock"></i>
Вхід</a></li>
                      <?php else: ?>
                        <li><a href="/cabinet/"><i class="fa fa-user"></i>
Обліковий запис</a></li>
                        <li><a href="/user/logout/"><i class="fa fa-unlock"></i>
Вихід</a></li>
                      <?php endif; ?>
                    </ul>
```

```
                    </div>
                </div>
            </div>
        </div>
    </div><!--/header-middle-->

    <div class="header-bottom"><!--header-bottom-->
        <div class="container">
            <div class="row">
                <div class="col-sm-12">
                    <div class="navbar-header">
                        <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                            <span class="sr-only">Toggle navigation</span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                            <span class="icon-bar"></span>
                        </button>
                    </div>
                    <div class="mainmenu pull-left">
                        <ul class="nav navbar-nav collapse navbar-collapse">
                            <li><a href="/">Головна</a></li>
                            <li class="dropdown"><a href="#">Магазин<i class="fa fa-angle-down"></i></a>
                                <ul role="menu" class="sub-menu">
                                    <li><a href="/catalog/">Каталог товарів</a></li>
                                    <li><a href="/cart/">Кошик</a></li>
```

```
                    </ul>
                </li>
                <li><a href="/about/">Про магазин</a></li>
                <li><a href="/contacts/">Контакти</a></li>
            </ul>
        </div>
    </div>
</div>
</div>
</div><!--/header-bottom-->

</header><!--/header-->
```