УДК 004.738.5

ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ МАРКЕТПЛЕЙСУ АВТОМОБІЛІВ

Білий Д.О., здобувач вищої освіти *e-mail*: *Yuichirohakya227@gmail.com*
Науковий керівник: к.т.н., доцент Холодняк Ю.В.
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Актуальність та постановка проблеми. Розробка веб-сайтів і маркетплейсів є ключовим компонентом сучасної цифрової економіки. Зростання популярності електронної комерції підкреслює потребу в ефективних платформах, які забезпечують зручність і функціональність для користувачів. У контексті автомобільної галузі маркетплейси стають невід'ємним елементом для задоволення попиту на нові та вживані транспортні засоби.

Основні виклики включають створення зручного інтерфейсу, що спрощує взаємодію користувачів із платформою, забезпечення швидкого доступу до великого обсягу даних, реалізацію функціоналу для ефективного пошуку та фільтрації товарів, а також підтримку високого рівня безпеки та продуктивності при масштабуванні системи.

У цій статті розглядається питання вибору сучасних інструментів веб-розробки, що відповідають вимогам до створення маркетплейсу для автомобілів, аналізуються їх переваги, недоліки та застосовність у межах зазначеної проблематики.

Основні матеріали дослідження. Для ефективної розробки маркетплейсу важливо обрати інструменти, які відповідають ключовим критеріям: швидкість розробки, продуктивність, масштабованість, простота інтеграції та підтримка сучасних стандартів. Розглянемо основні технології, що можуть бути використані.

1. Фронтенд (React).

React – це бібліотека JavaScript для побудови інтерфейсів користувача. Однією з ключових особливостей React є використання віртуального DOM, що забезпечує високу продуктивність. Завдяки віртуальному DOM React оновлює тільки ті частини інтерфейсу, які зазнали змін, а не перерендерює всю сторінку. Це робить бібліотеку ідеальною для розробки додатків з інтенсивною взаємодією користувача, де необхідно швидко реагувати на дії.

React підтримує компонентний підхід до розробки, де інтерфейс складається з незалежних, багаторазових компонентів. Це дозволяє розробникам створювати модульний код, який легко підтримувати та масштабувати. Наприклад, у маркетплейсі автомобілів можна використовувати окремі компоненти для відображення карток автомобілів, форм пошуку чи фільтрації.

React має широку екосистему, яка включає численні бібліотеки та інструменти, що розширюють функціонал. Для управління станом додатка можуть використовуватись такі інструменти, як Redux або Context API. Бібліотеки для управління формами, як-от Formik, і для валідації, такі як Yup, інтегруються з React без зайвих зусиль, спрощуючи обробку введення користувачів і забезпечуючи коректність даних.

2. Бекенд (Node.js + Express.js).

Node.js є високопродуктивним серверним середовищем виконання, що працює на основі JavaScript і використовує модель асинхронного вводу/виводу, яка базується на обробці подій. Такий підхід забезпечує масштабованість,

дозволяючи обробляти велику кількість одночасних запитів без блокування потоків. Ця особливість робить Node.js ідеальним вибором для розробки систем, які вимагають швидкої реакції на запити в реальному часі, таких як маркетплейси.

Express.js, як легкий та гнучкий фреймворк для Node.js, значно спрощує розробку серверних додатків. Він забезпечує зручний інтерфейс для створення API, налаштування маршрутизації та обробки HTTP-запитів і відповідей. Завдяки широкому набору вбудованих функцій та можливості розширення за допомогою сторонніх модулів, Express.js дає змогу розробникам швидко реалізовувати складні серверні функції, такі як автентифікація користувачів, обробка даних форм, інтеграція з базами даних та реалізація middleware для управління запитами.

Крім того, комбінація Node.js та Express.js дозволяє створювати бекенд із гнучкою та модульною архітектурою, яка легко адаптується до змін вимог. Це важливо для маркетплейсу автомобілів, де можливе зростання кількості користувачів і даних, а також інтеграція додаткових функцій, таких як фільтри, сортування, онлайн-оплата чи повідомлення. Висока продуктивність та масштабованість Node.js у поєднанні з простотою налаштування і багатофункціональністю Express.js забезпечують стабільну роботу платформи навіть під великим навантаженням.

3. База даних (PostgreSQL).

PostgreSQL – це потужна реляційна база даних із відкритим кодом. Її відмінною рисою є можливість працювати з великими обсягами даних і підтримувати складні запити завдяки потужному SQL-двигуну. PostgreSQL активно розвивається і має відкритий код, що дозволяє адаптувати її функціонал до потреб конкретного проєкту.

Однією з основних переваг PostgreSQL є підтримка транзакцій з властивостями ACID (атомарність, узгодженість, ізольованість, довговічність), що забезпечує високу надійність і коректність обробки даних. Завдяки цьому PostgreSQL підходить для систем, де важлива цілісність даних, зокрема для маркетплейсу автомобілів, де велика кількість записів про транспортні засоби постійно оновлюється.

У нашому випадку PostgreSQL використовуватиметься для збереження даних у форматі CSV про автомобілі, включаючи інформацію про моделі, технічні характеристики, ціну та рік випуску. Дані будуть імпортовані у форматі CSV, що зручно для первинного завантаження інформації у базу. PostgreSQL дозволяє швидко виконувати операції імпорту та експорту великих обсягів даних, а також працювати з ефективними індексами для прискорення пошуку.

4. Стилізація (SCSS).

SCSS (Sassy CSS) дозволяє організувати стильові файли більш структуровано, підтримуючи змінні, вкладеність і міксіни, що полегшує розробку адаптивного інтерфейсу.

У нашому маркетплейсі автомобілів SCSS відіграє ключову роль у створенні адаптивного дизайну, що забезпечить зручність використання на різних пристроях, від смартфонів до настільних комп'ютерів. Завдяки змінним можна легко керувати кольоровою схемою та іншими параметрами стилю, забезпечуючи відповідність корпоративному бренду або вимогам користувача. Вкладеність і міксіни спрощують розробку складних інтерфейсів, таких як фільтри, форми або картки автомобілів, забезпечуючи єдність стилю та легкість підтримки коду.

Таким чином, використання SCSS у проєкті дозволяє створити естетичний, адаптивний і підтримуваний інтерфейс, що є критично важливим для сучасного маркетплейсу, орієнтованого на високу якість користувацького досвіду.

5. Обробка форм (Formik + Yup).

Formik є популярною бібліотекою JavaScript для управління формами у вебдодатках, яка значно спрощує процес створення, обробки та валідації форм. Її використання допомагає розробникам уникнути зайвого коду та зосередитися на логіці роботи додатка, зокрема, на обробці введених користувачем даних.

Formik забезпечує зручну інтеграцію з інструментами для валідації даних, такими як Yup. За допомогою Yup можна створювати схеми перевірки, які визначають правила валідації для кожного поля форми. Наприклад, у маркетплейсі автомобілів можна встановити такі правила: рік випуску автомобіля має бути числом у певному діапазоні, поле ціни не може бути порожнім, а електронна адреса на маркетплейсі автомобілів Formik може бути використаний для створення форм пошуку, реєстрації, додавання оголошень чи зворотного зв'язку. Завдяки автоматизації процесів валідації та управління станом форм, розробка стає ефективнішою, а кінцевий продукт – зручнішим для користувачів.

Таким чином, Formik у поєднанні з Yup дозволяє гарантувати коректність введення даних, скорочує кількість помилок у формі та забезпечує високу якість взаємодії користувача з платформою. Це робить її важливим інструментом для сучасних вебдодатків, таких як маркетплейс автомобілів.є відповідати стандартному формату.

Висновки. У статті було здійснено аналіз основних інструментів, що будуть використані для розробки маркетплейсу автомобілів. було обрано наступні інструменти:

– React для побудови інтерфейсу користувача, що забезпечить високу продуктивність і зручність;
– Node.js та Express.js для створення бекенд-частини з масштабованою архітектурою;
– PostgreSQL для збереження даних про автомобілі;
– SCSS для адаптивного дизайну;
– Formik і Yup для управління формами та їх валідації.

Ці технології відповідають вимогам до швидкості, масштабованості та зручності підтримки, що робить їх оптимальними для створення маркетплейсу автомобілів.

Список використаних джерел:

1. React. *React*. URL: https://reactjs.org (дата звернення: 06.12.2024).
2. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org (дата звернення: 06.12.2024).
3. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org (дата звернення: 06.12.2024).
4. Formik. *Formik: Build forms in React, without the tears*. URL: https://formik.org (дата звернення: 06.12.2024).
5. GitHub - jquense/yup: Dead simple Object schema validation. *GitHub*. URL: https://github.com/jquense/yup (дата звернення: 06.12.2024).