УДК 004.777
DOI: 10.31866/2617-796X.8.1.2025.335535

Ольга Ткаченко,
кандидат фізико-математичних наук, доцент,
доцент кафедри інформаційних технологій,
Державний університет інфраструктури та технологій
(Національний транспортний університет),
Київ, Україна
oitkachen@gmail.com
https://orcid.org/0000-0003-1800-618X

Денис Мосюра,
бакалавр, кафедра інформаційних технологій,
Державний університет інфраструктури та технологій
(Національний транспортний університет),
Київ, Україна
denismosura@gmail.com
https://orcid.org/0009-0002-3145-2416

ВИКОРИСТАННЯ REACT ТА REDUX ПІД ЧАС РОЗРОБКИ ВЕБЗАСТОСУНКІВ: ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ

Розробка вебзастосунків від простих вебсторінок до складних інтерактивних платформ, таких як соціальні мережі, системи e-commerce, корпоративні портали, потребує використання нових підходів, інструментів і технологій. Особливу роль у розробці вебзастосунків відіграють JavaScript-бібліотеки та фреймворки, які спрощують створення ефективних, масштабованих і зручних інтерфейсів.

Метою статті є дослідження проблем та перспектив використання React у поєднанні з Redux під час створення вебзастосунків складної структури, оцінка переваг їх сумісного використання порівняно з аналогічними інструментами та розробка вебзастосунку «FarmerProducts».

Методами дослідження є основні методологічні підходи та технологічні засоби для розробки вебзастосунків. Такими методами, зокрема, є системний і порівняльний аналізи – для виявлення особливостей створення вебзастосунків; метод експертних оцінок, який передбачає аналіз літературних джерел та інформаційних ресурсів, проведення інтерв'ю та опитування експертів, а також процеси розробки й тестування масштабованих і високопродуктивних вебзастосунків на основі використання REACT та REDUX.

Новизною дослідження є аналіз сучасного інструментарію (React, Angular, Vue.js) з розробки вебзастосунків, визначення проблем та перспектив їх застосування, створення авторського вебзастосунку (за допомогою React та Redux), який охоплює інтуїтивно зрозумілий інтерфейс, гнучку базу даних, спрощуючи процес роботи із застосунком завдяки використанню стеку сучасних технологій.

Висновки. У роботі проаналізовано проблеми та перспективи використання сучаного інструментарію розробки вебзастосунків; приділено основну увагу React, який підходить

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

для різних проєктів; проведено порівняльний аналіз React з Angular, Vue.js та Svelte, визначено їхні основні характеристики, переваги, недоліки; продемонстровано залежність вибору інструментарію від контексту проєкту (React у поєднанні з Redux може забезпечити оптимальний баланс між продуктивністю, гнучкістю та масштабованістю); описано авторський вебзастосунок – інтернет-магазин «FarmerProducts»; визначено подальші шляхи вдосконалення цього вебзастосунку.

Ключові слова: веброзробка; фреймворк; React; Redux; frontend-розробка; бібліотеки JavaScript.

Вступ. Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням популярності вебзастосунків, які стали важливою частиною життя кожного. Розробка вебзастосунків (від простих вебсторінок до складних інтерактивних платформ), зокрема таких, як соціальні мережі, системи e-commerce, корпоративні портали, потребує використання нових підходів, інструментів і технологій.

Особливу роль у розробці вебзастосунків відіграють JavaScript-бібліотеки та фреймворки, які обумовлюють спрощення створення ефективних, масштабованих і зручних інтерфейсів.

Бібліотека React, розроблена компанією Facebook, займає провідне місце завдяки своїй гнучкості, продуктивності та широкій екосистемі [https://react.dev/]. React у поєднанні з Redux (Prasandeep, 2024) – інструментом централізованого управління станом – стає інструментарієм для розробки складних вебзастосунків у різних предметних областях, наповнених різним функціоналом.

Актуальність дослідження обумовлена складністю сучасних вебзастосунків, під час розробки яких управління станом, оптимізація продуктивності, створення та застосування компонентів повторного використання (КПВ) стають складними проблемами, які потребують вирішення.

Традиційні підходи (наприклад, використання мови програмування JavaScript, бібліотек на кшталт jQuery [https://jquery.com/]) уже недостатньо ефективні для виконання великих проєктів, зокрема через відсутність чіткої структури наявного програмного продукту та складність його масштабування (Di Gese, 2019).

Сучасні бібліотеки (React, Angular [https://angular.dev/] чи Vue.js [https://vuejs.org/]) забезпечують підтримку вирішення вказаних проблем.

Для React характерним є компонентний підхід та концепція віртуальної об'єктної моделі документа *(Document Object Mode*, DOM), забезпечення швидкої візуалізації інтерфейсів вебзастосунків. Redux підтримує передбачуваність і прозорість у процесі роботи зі станом застосунку. Тому аналіз можливостей, переваг і недоліків цих популярних інструментів розробки вебзастосунків є важливим для розуміння проблем та перспектив їх подальшого використання у веброзробці. Сучасна веброзробка активно та динамічно розвивається. Тому вибір ефективних інструментів для створення інтерфейсів користувача стає важливим фактором успіху всього проєкту з розробки вебзастосунку.

JavaScript є основною мовою програмування, що використовується для frontend-розробки. JavaScript підтримує численні бібліотеки та фреймворки, що пропонують власні підходи до вирішення типових завдань веброзробки. Серед

найбільш популярних рішень слід звернути увагу на React, Angular, Vue.js, а серед менш поширених – на Svelte (Hernandez, 2024) чи Preact (Doglio, n.d.). Аналіз усіх цих інструментів дає змогу зрозуміти їх переваги та недоліки. Тому актуальність проблеми визначення, чому веброзробники для проєктування та реалізації складних вебзастосунків обирають саме React у поєднанні з Redux, не викликає сумнівів і потребує свого дослідження та вирішення.

Мета і завдання дослідження. Метою є дослідження проблем і перспектив використання React у поєднанні з Redux під час створення вебзастосунків складної структури, оцінка переваг їх сумісного використання порівняно з аналогічними інструментами та розробка вебзастосунку «FarmerProducts».

Досягнення мети дослідження передбачає виконання таких завдань:

– визначення основних підходів до frontend-розробки вебзастосунків;

– проведення порівняльного аналізу альтернативних рішень щодо розробки вебзастосунків;

– визначення аспектів інтеграції React із Redux;

– визначення здатності інструментів оптимізувати процес розробки вебзастосунків;

– визначення факторів, що впливають на якість розроблених вебзастосунків;

– використання React та Redux під час реалізації вебзастосунку «FarmerProducts».

Отже, дослідження, що здійснює систематизацію знань про використання React і Redux, може бути відображене на практиці під час їх застосування в реальному проєкті розробки сучасних вебзастосунків на прикладі вебзастосунку «FarmerProducts».

Результати дослідження. React – бібліотека JavaScript з відкритим кодом (facebook/react, n.d.), розроблена у 2013 р. компанією Facebook для створення динамічних і масштабованих інтерфейсів користувача. Концепція React – це компонентний підхід, за якого інтерфейс розбивається на невеликі, незалежні блоки – КПВ, що можна використовувати в різних складниках застосунку.

Особливістю React є віртуальна DOM, яка є проміжним шаром між кодом і реальною DOM у браузері.

Замість повної зміни візуального відображення вебсторінки React здійснює порівняння попереднього та поточного станів віртуальної DOM, оновлюючи лише ті елементи, які були змінені. Такий підхід забезпечує високу швидкість візуалізації (рендерингу), особливо в застосунках із частим оновленням даних, таких як чати чи дашборди.

React підтримує й декларативний підхід, коли веброзробники описують бажаний результат, а не процес його досягнення. Такий підхід спрощує читання та debugging (дебагінг – процес виявлення, аналізу та виправлення помилок або дефектів у програмному коді). Гнучкість React робить його унікальним серед конкурентів. Як бібліотека, а не фреймворк він не нав’язує строгих правил, дозволяючи інтегрувати додаткові інструменти (наприклад, Redux – для управління станом або Next.js – для серверного рендерингу). На рис. 1 зображено приклад коду, написаного за допомогою React, де поєднано синтаксис JavaScript та HTML.

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

```
import React from 'react';
import {Text, View} from 'react-native';
import {Header} from './Header';
import {heading} from './Typography';

const WelcomeScreen = () => (
  <View>
    <Header title="Welcome to React Native"/>
    <Text style={heading}>Step One</Text>
    <Text>
      Edit App.js to change this screen and turn it
      into your app.
    </Text>
    <Text style={heading}>See Your Changes</Text>
    <Text>
      Press Cmd + R inside the simulator to reload
      your app's code.
    </Text>
    <Text style={heading}>Debug</Text>
    <Text>
```

Рис. 1. Код, написаний за допомогою React
Джерело: (Guest, 2021)

React використовує JSX – синтаксис, який поєднує HTML-подібну розмітку з JavaScript-логікою, що суттєво полегшує створення КПВ і робить програмний код більш зрозумілим (у таких випадках говорять про так звану інтуїтивну зрозумілість). React охоплює численні бібліотеки (зокрема, React Router, Material-UI). Крім того, React підтримується активною спільнотою, що регулярно оновлює документацію та відповідні інструменти (React Community, n.d.).

За даними опитування Stack Overflow 2023 року, React утримує лідерство серед frontend-технологій розробки вебзастосунків, що підтверджує його популярність у різноманітних проєктах: від стартапів до великих корпорацій (наприклад, таких як Netflix чи Airbnb (Developer Survey, n.d.)).

Angular, розроблений Google, є повноцінним фреймворком (Angular (web framework), n.d.). Його сучасна версія (Angular 2+), представлена у 2016 р., базується на TypeScript – надбудові над JavaScript зі статичною типізацією, що зменшує ймовірність помилок під час роботи над великими проєктами. Angular пропонує комплексний набір інструментів, зокрема маршрутизацію, обробку форм, HTTP-клієнт і навіть утиліти для unit-тестування. Архітектура Angular спирається на модулі та компоненти, а двостороннє зв'язування даних дає змогу автоматич-

но синхронізувати модель і відображення. Наприклад, зміна значення у формі миттєво відображається в даних відповідного компонента, що зручно для простих CRUD-застосунків. На рис. 2 зображено приклад програмного коду, написаного за допомогою Angular.

```
EmployeeService.ts
import {Injectable} from '@angular/core'
import {Http} from '@angular/http';
import {Observable} from 'rxjs/Observable';
import 'rxjs/Rx';
import {Employee} from './employee.model';
@Injectable()
export class EmployeeService {
  private identifier: string = 'employees';
  constructor(private http: Http) {}
  protected BuildUrl(): string {
    return `http://exampleapi.com/api/${this.identifier}/employees.json`;
  }
  public GetEmployees(): Observable<Employee[]> {
    let requestUrl = this.BuildUrl();
    console.log(`requestUrl: ${requestUrl}`);
    return this.http.get(requestUrl)
      .map((jsonResponse) => <Employee[]>jsonResponse.json());
  }
}
```

Рис. 2. Код, написаний за допомогою Angular
Джерело: (Gorilla Logic, 2020)

Але, крім очевидних переваг, Angular має й певні недоліки: значно більший розмір bundle (*сукупність деяких програмних даних (файлів), об'єднаних за певною ознакою*) порівняно з React, що може сповільнити початкове завантаження вебсторінки.

Двостороннє зв'язування, хоча й зручне, але потребує постійного відстеження змін (особливо у великих застосунках), що може призвести до так званих «брудних перевірок» (dirty checking) (What is dirty checking in AngularJs, n.d.). Робота з Angular вимагає від розробників знання TypeScript і Dependency Injection. Angular оптимально підходить для проєктів з чіткою структурою, таких як ERP-системи, але його використання менш ефективне для швидких прототипів чи стартапів.

Vue.js створено у 2014 р., цей фреймворк можна використовувати як легку бібліотеку або як розширення до повноцінного рішення (Ways of Using Vue, n.d.). Невеликий розмір робить Vue.js одним з найбільш компактних інструментів для веброзробки. Vue.js поєднує реактивну систему даних з можливістю прописування компонентів в одному файлі, де розмітка, стилі та логіка об'єднані в єдину

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

структуру. Vue.js підтримує двостороннє зв'язування (подібно до Angular) та односпрямований потік даних через менеджера стану Vuex (подібно до Redux).

Використовуючи директиву v-model, у Vue.js можна легко реалізувати форму з автоматичним оновленням даних. На рис. 3 зображено приклад коду, написаного за допомогою Vue.js.

```
<template>
  <div>
    <p>You have counted {{this.counter}} times</p>
    <input type="text" ref="input">
    <button @click="submit">Add 1 to counter</button>
  </div>
</template>
<script>
export default {
  name: 'Test',
  data(){
    return {
      counter: 0
    }
  },
  methods: {
    submit(){
      this.counter++;
      this.$refs['input'].focus()
    }
  }
}
</script>
```

Рис. 3. Код, написаний за допомогою Vue.js
Джерело: (Patel, 2020)

На рис. 3 можна побачити, що елемент «button» має атрибут «@click», у який поміщається функція, що буде спрацьовувати під час натискання на цей елемент. У React такий атрибут має назву «onClick», а функція поміщається у фігурні дужки «{}».

Vue.js є привабливим для новачків і невеликих команд через його простоту та гарну документацію, яка вважається однією з найкращих у сфері frontend-розробки. Vue.js може використати Vue навіть у старих проєктах на jQuery (Armstrong, 2020). Однак Vue.js поступається React за розміром спільноти та кількістю сторонніх бібліотек.

Наприклад, для складних UI-компонентів у React є багато готових рішень, а у Vue.js їхній вибір обмежений. Vue.js популярний у Китаї (завдяки підтримці Alibaba), але його екосистема ще значно вужча, ніж у React.

Svelte, представлений у 2016 р., пропонує підхід, який відрізняє його від React та інших конкурентів (Hernandez, 2024). Замість роботи з віртуальною DOM у бра-

узері Svelte компілює JS-код під час збірки, тобто замість бібліотеки, яка працює в режимі run-time (як React), Svelte генерує код, що напряму маніпулює DOM. Наприклад, оновлення списку у Svelte відбувається через звичайні присвоювання (items = newItems), а не через спеціальні методи, як setState у React. Це забезпечує вищу швидкість і менший розмір bundle. На рис. 4 зображено приклад коду, написаного за допомогою Svelte.

```
<script>
  let items = [
    { id: 1, name: "Milk", done: false },
    { id: 2, name: "Bread", done: true },
    { id: 3, name: "Eggs", done: false }
  ];

  const remove = item => {
    items = items.filter(i => i !== item);
  };
</script>

<div>
  <h1>Things to Buy</h1>

  <ul>
    {#each items as item}
      <li>
        <span>{item.name}</span>
        <button on:click={() => remove(item)}>❌</button>
      </li>
    {/each}
  </ul>
</div>
```

Рис. 4. Код, написаний за допомогою Svelte
Джерело: (Dave Ceddia, 2019)

На рис. 4 можна побачити атрибут «on:click», який спрацьовує під час натискання на елемент «button».

Svelte приваблює своєю простотою та відсутністю boilerplate-коду, що робить його ідеальним для невеликих проєктів або застосунків, де продуктивність є критичною (наприклад, у реальному часі чи на слабких пристроях). Екосистема Svelte ще розвивається: кількість бібліотек і компонентів значно менша, ніж у React, а підтримка великих проєктів потребує додаткових інструментів (наприклад, таких як SvelteKit) (Hernandez, 2024). Svelte менш поширений у корпоративному IT-секторі, що може ускладнити пошук досвідчених веброзробників. Але попри все, інноваційність Svelte робить його перспективним конкурентом.

Переваги React перед іншими бібліотеками. Однією з переваг React є його підхід з використанням КПВ. Наприклад, кнопка чи форма авторизації, створені як React-компоненти, можуть застосовуватися на кількох сторінках без дублюван-

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

ня коду (це відрізняє React від Angular, де компоненти пов'язані з модулями та фреймворковою логікою).

У Vue.js структура компонентів в одному файлі (*Single-File Components*) іноді призводить до змішування стилів і логіки, що може ускладнити повторне використання компонентів у великих проєктах. React завдяки JSX і чіткому поділу на функції / класи забезпечує більшу модульність і читкість.

Використання КПВ в React підсилюється його декларативним стилем. Розробник описує, яким на вигляд має бути компонент у певному стані, а React автоматично оновлює інтерфейс під час зміни даних. Наприклад, список завдань у React може бути реалізований через компонент TaskList, який візуалізується на основі масиву даних, тоді як у чистому JavaScript чи jQuery це потребувало б ручного маніпулювання DOM для кожного оновлення. Svelte пропонує схожий декларативний підхід, але його компіляція у базовий JavaScript обмежує гнучкість у динамічних сценаріях, де React виграє завдяки runtime-оптимізації.

Продуктивність React суттєво залежить від використання віртуальної DOM (Virtual DOM and Internals, n.d.), яка є спрощеною копією реальної DOM, що дає змогу React порівнювати попередній і поточний стан інтерфейсу й оновлювати лише змінені елементи. Це особливо корисно в застосунках із частим оновленням даних, таких як стрічки новин чи інтерактивні dashboard (*документи з лаконічно представленими статистичними даними, звітами, часто з елементами інфографіки; програмні інтерфейси, віджети, робочі столи різних операційних систем*).

Порівнюючи React з аналогами, можна зробити висновок, що React має явну перевагу:

– Angular використовує двостороннє зв'язування та Zone.js для відстеження змін, що може призводити до надмірних перевірок у великих застосунках, знижуючи продуктивність;

– Vue.js поступається React у сценаріях з великою кількістю компонентів;

– Svelte втрачає в гнучкості в разі динамічних оновлень, бо весь код генерується статично.

React має розвинуту екосистему, яка охоплює багато бібліотек та інструментів. Наприклад, React Router забезпечує маршрутизацію (React Router Official Documentation, n.d.), Redux – управління станом, а Material-UI – готові UI-компоненти (Ready to use Material Design components, n.d.). Це дає змогу адаптувати React до будь-якого проєкту (від лендингів (односторінкових сайтів) до складних SPA (*Single Page Applications*)).

Angular обмежує вибір через свою монолітну архітектуру (розробник змушений дотримуватися правил фреймворку).

Vue.js пропонує меншу екосистему, і хоча Vuex чи Nuxt.js розширюють його можливості, кількість бібліотек поступається React. У Svelte такі інструменти, як SvelteKit, тільки набирають популярність.

React не нав'язує строгих шаблонів, на відміну від Angular з його модулями чи TypeScript-залежністю. React дає змогу легко інтегрувати сторонні рішен-

ня (наприклад, бібліотеки для анімацій (Framer Motion) чи роботи з графіками (Recharts)) (Wieruch, 2025).

Спільнота React є однією з найбільших у світі frontend-розробки, що забезпечує його постійний розвиток і підтримку.

Використання React під час розроблення інтернет-магазину фермерських продуктів. У цьому авторському проєкті продемонстровано можливості React в поєднанні з Redux у процесі створення сучасних вебзастосунків. Цей проєкт ілюструє, як компонентний підхід React і централізоване управління станом через Redux можна застосувати для реалізації функціонального, інтерактивного та масштабованого рішення.

Інтернет-магазин призначений для продажу органічних продуктів, таких як овочі, фрукти, молочні продукти та м'ясо, від місцевих фермеріві містить типові компоненти системи e-commerce: каталог товарів, кошик, вибір мови інтерфейсу й оформлення замовлення.

Мета розробки інтернет-магазину з продажу фермерських екопродуктів:

– створення зручного інтерфейсу для користувачів, які прагнуть здійснювати онлайн-покупки свіжих фермерських продуктів;

– демонстрація переваг React і Redux у реальному сценарії.

Основними компонентами проєкту інтернет-магазину є:

1. Каталог товарів: відображення списку продуктів із фільтрами за категоріями (наприклад, «Фрукти та овочі») та сортуванням за ціною і оцінками (рис. 5).

2. Кошик: додавання товарів до кошика, зміна кількості, видалення позицій і підрахунок загальної суми (рис. 6).

3. Вибір мови: підтримка двох мов (наприклад, української та англійської) з автоматичним перекладом інтерфейсу. Ця функція реалізовується за допомогою Redux-сховища, де створено окремий слайс для локалізації (рис. 7).

На рис. 8 зображено зовнішній вигляд головної сторінки в разі вибору української мови.

На рис. 9 зображено зовнішній вигляд головної сторінки в разі вибору англійської мови.

4. Можливість пошуку товарів: користувач має змогу знайти товар, що його цікавить за допомогою пошукового рядка, при чому неважливо, якою мовою буде відбуватися запит і, відповідно, пошук (рис. 10 та рис. 11).

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

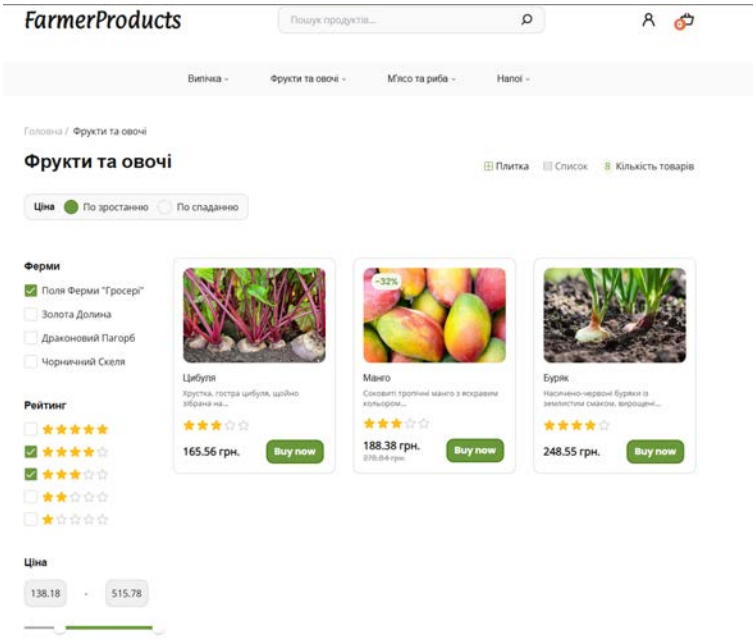

Рис. 5. Відображення продуктів з обраними фільтрами
Джерело: авторська розробка

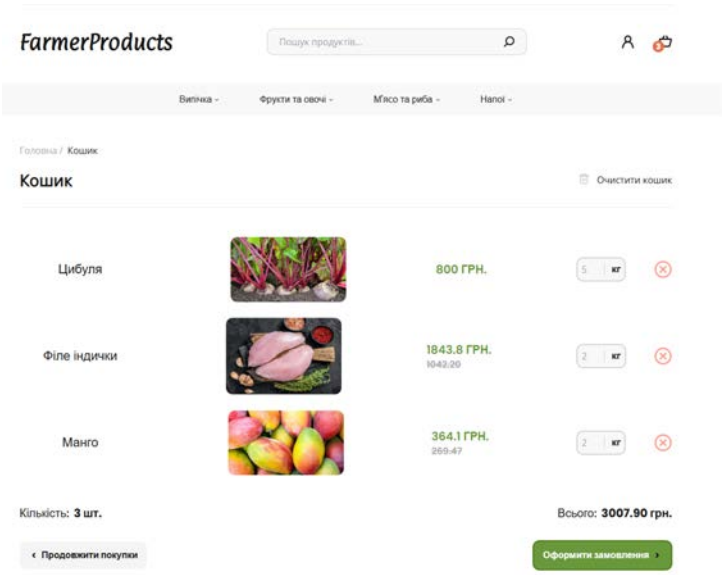

Рис. 6. Відображення роботи кошика, динамічної зміни кількості доданих товарів
Джерело: авторська розробка

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

```
const localizSlice = createSlice({
  name: 'localiz',
  initialState,
  reducers: {

    setLanguage(state, action) {
      state.language = action.payload;
      console.log('language changed to', state.language)
    }
  },
  extraReducers: (builder) => {
    builder
      .addCase(fetchExchangeRate.pending, (state) => {
        state.status = 'loading';
      })
      .addCase(fetchExchangeRate.fulfilled, (state, action) => {
        state.status = 'succeeded';
        state.exchangeRate = action.payload; // Обновляем курс
      })
      .addCase(fetchExchangeRate.rejected, (state, action) => {
        state.status = 'failed';
        state.error = action.payload;
      });
  },
});
```

Рис. 7. Слайс локалізації localizSlice
Джерело: авторська розробка

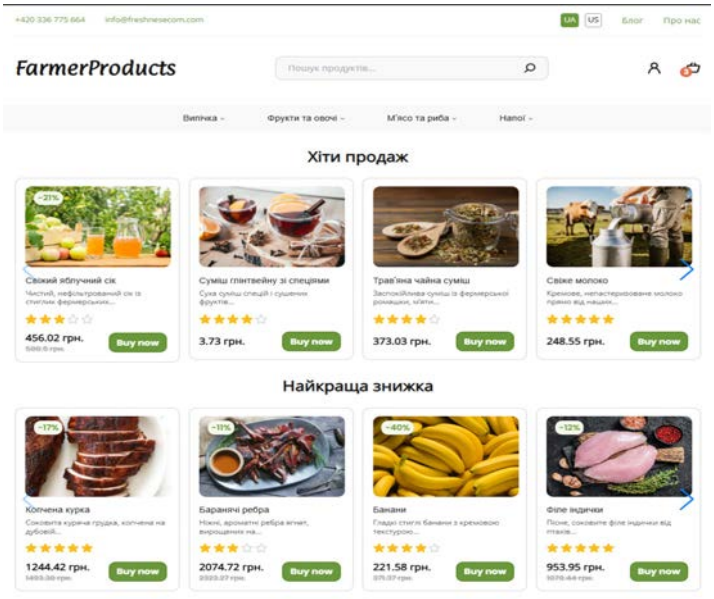

Рис. 8. Зовнішній вигляд головної сторінки українською мовою
Джерело: авторська розробка

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

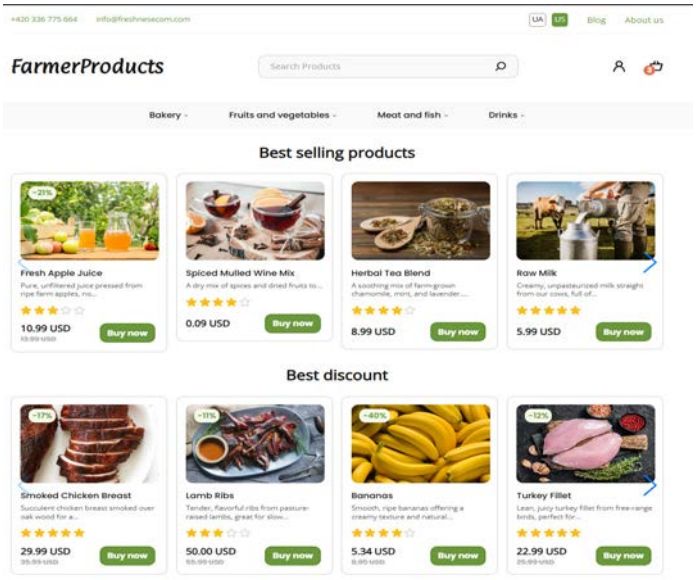

Рис. 9. Зовнішній вигляд головної сторінки англійською мовою
Джерело: авторська розробка

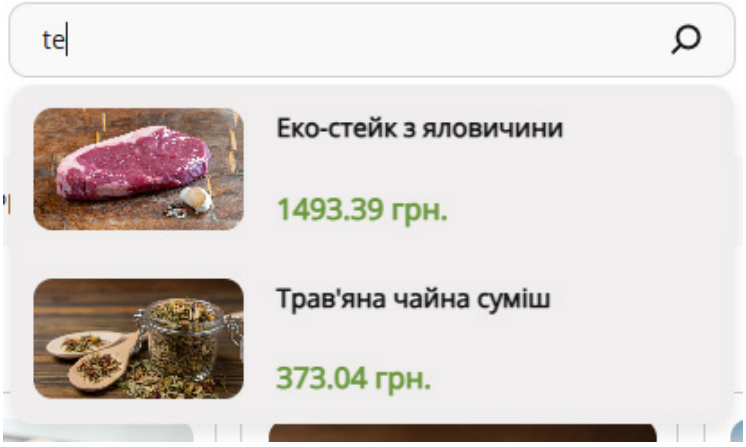

Рис. 10. Пошук продукту англійською мовою
Джерело: авторська розробка

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

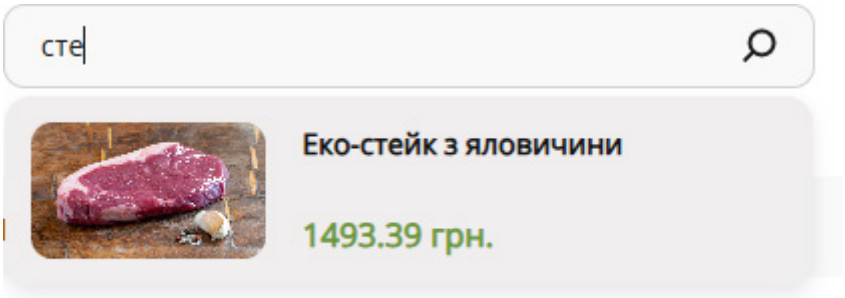

Рис. 11. Пошук українською мовою
Джерело: авторська розробка

На зображеннях головної сторінки інтернет-магазину можна побачити результат компонентного підходу до розробки вебзастосунку. Цей підхід спростив процес розробки, тому що компонент створюється лише один раз, після чого його можна використовувати будь-де в проєкті. Це добре видно по картках товарів. Усі картки – це один і той самий компонент ProductCard, вміст якого змінюється залежно від вкладеної інформації. Розглянемо цей компонент детальніше.

На рис. 12 зображено фрагмент компонента ProductCard. У цьому фрагменті міститься логіка компонента, запити (на отримання даних, поточної мови), створення станів за допомогою React.useState().

На рис. 13 зображено архітектуру React-проєкту. А у табл. 1 описано, які функції виконуються за допомогою відповідних файлів проєкту.

Таблиця 1

Папки, файли та функції

Папка або файл	Функція
assets	Зберігання невеликих зображень, svg-файлів
components	Зберігання всіх створених компонентів застосунку
pages	Зберігання компонентів, які займають повну сторінку
redux	Зберігання конфігураційних файлів Redux-сховища
scss	Зберігання стилізаційних (scss) файлів
server	Зберігання back-end частини застосунку
index.js	Файл, у якому задається архітектура проєкту

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

```
const ProductCard = ({productId, title_us, title_ua, descr_us, descr_ua, imgUrl, price, datedPrice,
rating, freshness, farmId, amount, view, wishlisted, measureId }) => {

  const sale = datedPrice ? Math.round((1 - price / datedPrice) * 100) : null;
  const [farm, setFarm]=React.useState();
  const [category, setCategory]=React.useState();
  const [measure, setMeasure]=React.useState();

  const currLang=useSelector(({localiz})=>localiz.language);
  const currRate=useSelector(({localiz})=>localiz.exchangeRate)



  React.useEffect(()=>{
    const getFarm=async()=>{
      const farm = await getFarmById(farmId);
      setFarm(farm[0]);
    }
    const getCategory=async()=>{
      //console.log('productId', productId)
      const categoryFetch=await getCategoryByProductId(productId);
      //console.log('categoryFetch', categoryFetch)
      setCategory(categoryFetch)
      //console.log('category', category)
    }
    const getMeasure=async()=>{
      let measure=await getMeasures();
      measure=measure.filter(item=>item.measureId==measureId);
      setMeasure(measure[0]);
    }
    getCategory();
    getFarm();
    getMeasure();

  }, [])

  const imgArr=imgUrl?JSON.parse(imgUrl):[''];

  let shrinkedDescrUA=descr_ua;
  let shrinkedDescrUS=descr_us;
  shrinkedDescrUA=shrinkedDescrUA.split(" ").slice(0, 6).join(" ");
  shrinkedDescrUS=shrinkedDescrUS.split(" ").slice(0, 9).join(" ");
```

Рис. 12. Перша частина компонента ProductCard
Джерело: авторська розробка

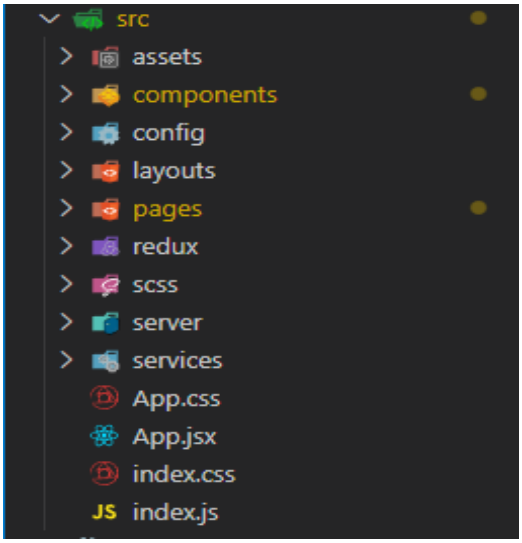

Рис. 13. Архітектура React-проєкту
Джерело: авторська розробка

Висновки. У роботі проаналізовано проблеми та перспективи використання сучасного інструментарію розробки вебзастосунків. Основну увагу приділено React (з його віртуальною DOM, компонентним підходом і гнучкістю), який підходить як для малих, так і для великих проєктів. Проведено порівняльний аналіз React та Angular, Vue.js та Svelte, визначено їхні основні характеристики, переваги, недоліки. Визначено, що жоден з інструментів Angular, Vue.js та Svelte не вирішує всіх завдань складних вебзастосунків, особливо в управлінні станом. Для усунення цієї проблеми React часто доповнюється Redux.

Проведений аналіз продемонстрував залежність вибору інструментарію від контексту проєкту, але саме React у поєднанні з Redux може запропонувати оптимальний баланс між продуктивністю, гнучкістю та масштабованістю.

З урахуванням результатів проведеного аналізу інструментарію розробки вебзастосунків виконано проєкт з розробки авторського вебзастосунку – інтернет-магазину «FarmerProducts». Визначено подальші шляхи вдосконалення цього вебзастосунку.

Отже, можна зробити висновок, що проведене дослідження та його практична реалізація мають важливе значення для розробників вебзастосунків, які цікавляться розвитком відповідного інструментарію та екосистем.

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

REFERENCES

Angular (web framework), n.d. *Wikipedia*. [online] Avialable at: <https://en.wikipedia.org/wiki/Angular_(web_framework)> [Accessed 01 March 2025].

Armstrong, L., 2020. Using Vue in jQuery and JavaScript Sites. *DEV*, [online] 1 March. Avialable at: <https://dev.to/workingwebsites/using-vue-in-jquery-and-javascript-sites-1ib7> [Accessed 01 March 2025].

Dave Ceddia, 2019. *Introduction to Svelte*. [online] 24 July. Avialable at: <https://daveceddia.com/svelte-intro/> [Accessed 01 March 2025].

Developer Survey, n.d. [online] Avialable at: <https://survey.stackoverflow.co/2023/#section-admired-and-desired-web-frameworks-and-technologies> [Accessed 01 March 2025].

Di Gese, I., 2019. jQuery isn't useful anymore. Here's why. Do we still need jQuery to DOM-script web pages?. *Medium*, [online] 29 September. Avialable at: <https://javascript.plainenglish.io/jquery-will-die-soon-heres-why-976a6b8105e1> [Accessed 01 March 2025].

Doglio, F., n.d. Top 7 Frontend Frameworks to Use in 2025: Pro Advice. *Roadmap.sh*. [online] Avialable at: <https://roadmap.sh/frontend/frameworks> [Accessed 01 March 2025].

facebook/react, n.d. *GitHub*. [online] Avialable at: <https://github.com/facebook/react> [Accessed 01 March 2025].

Gorilla Logic, 2020. How to Achieve Aspect-Oriented Programming (AOP) in Angular 2+ *Gorilla Logic*, [blog] 6 February. Avialable at: <https://gorillalogic.com/blog-and-resources/how-to-achieve-aspect-oriented-programming-aop-in-angular-2> [Accessed 01 March 2025].

Guest, 2021. What is React JS? *Simpat Tech*, [online] 23 August. Avialable at: <https://simpat.tech/what-is-react/> [Accessed 01 March 2025].

Hernandez, I., 2024. Svelte vs. React: The Ultimate JavaScript Framework Showdown. *Dreamhost*, [online] 14 August. Avialable at: <https://www.dreamhost.com/blog/svelte-vs-react/> [Accessed 01 March 2025].

Patel, K., 2020. Vue JS FAQs. *Francium Tech*, [blog] 11 April. Avialable at: <https://blog.francium.tech/vue-js-faqs-ddf744dd4093> [Accessed 01 March 2025].

Prasandeep, 2024. How To Integrate Redux with React Based Application: A Step By Step Tutorial. *Semaphore*, [online] 31 January. Avialable at: <https://semaphore.io/blog/redux-react> [Accessed 01 March 2025].

React Community, n.d. [online] Avialable at: <https://react.dev/community> [Accessed 01 March 2025].

React Router Official Documentation, n.d. [online] Avialable at: <https://reactrouter.com/> [Accessed 01 March 2025].

Ready to use Material Design components, n.d. [online] Avialable at: <https://mui.com/material-ui/?srsltid=AfmBOora8qN_ab1KonqyHwKXn3saxNOXMCfK_9sjV7Ng-qGC8FmCKxUc> [Accessed 01 March 2025].

Virtual DOM and Internals, n.d. *React*. [online] Avialable at: <https://uk.legacy.reactjs.org/docs/faq-internals.html> [Accessed 01 March 2025].

Ways of Using Vue, n.d. *Vue.js*. [online] Avialable at: <https://vuejs.org/guide/extras/ways-of-using-vue> [Accessed 01 March 2025].

What is dirty checking in AngularJs and how does it work?, n.d. *Quora*. [online] Avialable at: <https://www.quora.com/What-is-dirty-checking-in-AngularJs-and-how-does-it-work> [Accessed 01 March 2025].

Wieruch, R., 2025. React Libraries for 2025. *Robin Wieruch*, [blog] 17 February. <https://www.robinwieruch.de/react-libraries/> [Accessed 01 March 2025].

UDK 004.777

Olha Tkachenko,
PhD in Physics and Mathematics, Associate Professor,
Associate Professor at the Department of Information Technologies,
State University of Infrastructure and Technology
(National Transport University),
Kyiv, Ukraine
oitkachen@gmail.com
https://orcid.org/0000-0003-1800-618X

Denys Mosiura,
Bachelor, Department of Information Technologies,
State University of Infrastructure and Technology
(National Transport University),
Kyiv, Ukraine
denismosura@gmail.com
https://orcid.org/0009-0002-3145-2416

USING REACT AND REDUX IN WEB APPLICATION DEVELOPMENT: PROBLEMS AND PROSPECTS

Web application development from simple web pages to complex interactive platforms (including social networks, e-commerce systems, corporate portals) requires new approaches, tools and technologies. JavaScript libraries and frameworks play a special role in developing web applications, simplifying the creation of effective, scalable, and convenient interfaces.

The purpose of the article is to research the problems and prospects of using React in combination with Redux when creating web applications with a complex structure, to assess the advantages of their joint use in comparison with similar tools and to develop the web application “FarmerProducts”.

The research methodology consists of the main methodological approaches and technological tools for developing web applications. Such methods, in particular, are: system and comparative analysis – to identify the features of creating web applications; method of expert assessments, which involves the study of literary sources and information resources, conducting interviews and surveys of experts, as well as the processes of developing and testing scalable and high-performance web applications based on the use of React and Redux.

The scientific novelty of the research is the analysis of modern tools (React, Angular, Vue.js) for developing web applications, identifying problems and prospects for their use, creating an author’s web application (using React and Redux), which includes an intuitive interface, a flexible database, simplifying the process of working with the application by using a stack of modern technologies.

ISSN 2617-796X (print)
ISSN 2618-0049 (online)

Conclusions. The paper analyses the problems and prospects of using modern web application development tools; focuses on React, which is suitable for different projects; conducts a comparative analysis of React with Angular, Vue. js and Svelte, identifies their main characteristics, advantages, disadvantages; demonstrates the dependence of the choice of tools on the project context (React in combination with Redux can provide an optimal balance between performance, flexibility and scalability); describes the author's web application – the FarmerProducts online store; identifies further ways to improve this web application.

Keywords: web development; framework; React; Redux; frontend development; JavaScript libraries.

Надійшла 04.03.2025
Прийнята 23.05.2025
Стаття була вперше опублікована онлайн 18.07.2025