моделювання складних процесів. У техніці вони допомагають розробляти системи управління, наприклад, проєктувати електричні схеми або створювати алгоритми для автоматизації роботів.

В економіці ці засоби застосовуються для побудови моделей, що аналізують фінансові потоки, або для оптимізації розподілу ресурсів у великих організаціях. Отже, Maple і SIMULINK забезпечують ефективний інструментарій для розв'язання спеціалізованих задач у науці, техніці та бізнесі.

Maple і SIMULINK забезпечують потужні можливості для моделювання, аналізу та вдосконалення складних систем у різних галузях. Залежно від характеру завдання Maple підходить для точних розрахунків і теоретичного обґрунтування, тоді як SIMULINK ідеально підходить для динамічного моделювання та перевірки поведінки систем у реальних умовах. Їх поєднання дає змогу досягти балансу між глибоким аналітичним підходом і практичною реалізацією, що робить ці платформи універсальними для інженерних і наукових досліджень.

Перспективи розвитку Maple та SIMULINK зосереджуються на покращенні їх взаємодії, що допоможе автоматизувати більш складні процеси моделювання та зменшити час на налаштування моделей. Очікується розширення бібліотек SIMULINK, що дасть змогу створювати ще більш спеціалізовані блоки для широкого спектра галузей, як-от аерокосмічна або біомедична інженерія. До того ж розробка нових алгоритмів для аналізу результатів симуляції допоможе спростити інтерпретацію даних і зробити процес оптимізації моделей швидшим і точнішим.

Список використаних джерел

1. Використання математичного пакету Maple для розв'язування та моделювання задач. URL: https://lib.pnu.edu.ua:8080/bitstream/123456789/3286/1/Використання математичного пакету maple для розв'язування та моделювання задач_методичка.pdf (дата звернення: 02.12.2024).

2. Ніколюк П. К. Моделювання систем: навч. посіб. для здобувачів вищої освіти спеціальності 122 «Комп'ютерні науки». Вінниця: ДонНУ, 2023. 33 с. (дата звернення: 02.12.2024).

3. Introduction: Simulink Modeling. URL: https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction&section=SimulinkModeling (дата звернення: 02.12.2024).

УДК 004.43:004.774.6

Мишківська Я. В., здобувачка вищої освіти,
Січко Т. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

АНАЛІЗ ПРОДУКТИВНОСТІ ХУКІВ У ПОРІВНЯННІ З КЛАСОВИМИ КОМПОНЕНТАМИ

Донецький національний університет імені Василя Стуса, м. Вінниця

Користувацький інтерфейс (UI) є критично важливим для залучення та утримання користувачів. Інтуїтивність, адаптивність і привабливий дизайн за-

безпечують позитивний досвід взаємодії з вебзастосунком. React (інколи React.js, ReactJS) – це JavaScript-бібліотека для створення гнучких та сучасних користувацьких інтерфейсів (UI) для вебзастосунків. Вона дає змогу розробити все те, з чим користувач вебресурсу може взаємодіяти напряму: привабливе оформлення сайта, ефектні анімації, адаптивний дизайн, який підлаштовується під різні пристрої, тощо [1]. Впровадження хуків стало переломним моментом у розвитку React, оскільки вони значно спростили управління станом і побічними ефектами у функціональних компонентах, підвищивши ефективність розробки. Цей перехід актуалізував питання про вибір між класовими та функціональними компонентами залежно від вимог проєкту.

До появи хуків класовий підхід був основним способом розробки в React. Він базується на ES6-класах і використовує методи життєвого циклу для управління станом та побічними ефектами. Особливості класових компонентів у React включають використання this.state для зберігання стану, методів життєвого циклу для управління логікою та необхідність прив'язки контексту this для методів.

Хуки – це функції, за допомогою яких можна «зачепитися» за стан та методи життєвого циклу React із функційних компонентів [2]. Найцікавіша їх особливість полягає в тому, що їх можна викликати лише на найвищому рівні компонента. На практиці це означає, що не можна викликати хуки всередині умов (if), циклів, інших функцій або після раннього return. Це правило забезпечує виклик хуків в одному і тому ж порядку під час кожного ререндера компонента [3].

У табл. 1 наведено порівняльний аналіз кожного підходу.

Таблиця 1 – Порівняння хуків і класових компонентів

Критерій	**React Hooks (Функціональні компоненти)**	**Класові компоненти**
Час рендерингу	• Швидший завдяки відсутності методів життєвого циклу. • Оптимізація через мемоізацію (React.memo, useCallback, useMemo)	• Повільніший через обробку методів життєвого циклу
Споживання пам'яті	• Економніше: не створюються екземпляри класів	• Більше витрат пам'яті через створення екземплярів класів
Читабельність коду	• Код короткий, декларативний, зосереджений на функціях	• Громіздкий код із численними методами життєвого циклу
Обробка побічних ефектів	• Використання useEffect, що дає змогу гнучко контролювати залежності	• Використання методів життєвого циклу (componentDidMount, componentDidUpdate)
Гнучкість повторного використання логіки	• Легко реалізується через кастомні хуки	• Повторне використання складніше: потрібні HOC або Render Props
Продуктивність у великих додатках	• Висока за умови правильної мемоізації залежностей	• Може знижуватися через складну логіку в методах життєвого циклу
Підтримка тестування	• Зручно тестувати завдяки ізольованим функціональним підходам	• Тестування складніше через наявність стану та контексту класів

Отже, хуки забезпечують вищу продуктивність, економію ресурсів і спрощення коду, що робить їх ідеальним вибором для сучасних проєктів. Вони спрощують управління станом і побічними ефектами, даючи змогу зосередитися на логіці компонента. Класові компоненти залишаються доцільними для підтримки старих систем із наявною логікою, але їх поступовий перехід на хуки підвищить ефективність розробки.

Список використаних джерел

1. Як стати React розробником. Що потрібно знати та вміти – з нуля до рівня спеціаліста. ITVDN. URL: https://itvdn.com/ua/blog/article/how-to-become-a-react-developer (дата звернення: 26.11.2024).

2. Огляд хуків. React Documentation. URL: https://uk.legacy.reactjs.org/docs/hooks-overview.html (дата звернення: 26.11.2024).

3. Розбираємося з хуками в React. DOU. URL: https://dou.ua/forums/topic/47858/ (дата звернення: 28.11.2024).

4. Павлов Д. Л., Січко Т. В. Принцип роботи Web API та його застосування. Прикладні інформаційні технології: матеріали всеукр. наук.-практ. конф. (м. Вінниця, 2023). С. 166–168.

УДК 004.652

Морозюк А. А., здобувач вищої освіти,
Зелінська О. В., канд. техн. наук, доцент,
доцент кафедри інформаційних технологій

ІНТЕГРАЦІЯ ПРИНЦИПІВ ПРОДУКТОВОЇ АНАЛІТИКИ У ДИЗАЙНІ ІНФОРМАЦІЙНИХ СИСТЕМ

Донецький національний університет імені Василя Стуса, м. Вінниця

Сучасні інформаційні системи відіграють ключову роль у нашому житті, забезпечуючи функціональність і підтримуючи роботу різних сфер – від бізнесу до охорони здоров’я. Однак часто такі системи не відповідають очікуванням користувачів, оскільки дизайнери ігнорують важливість аналізу поведінки аудиторії та розуміння її реальних потреб. Як наслідок, це призводить до низького рівня залучення, проблем з інтуїтивністю інтерфейсу та навіть відмови від використання продукту. Саме тому інтеграція підходів продуктової аналітики у дизайн інформаційних систем набуває все більшої актуальності. Використання даних про поведінку користувачів та їхні вподобання дає змогу створювати продукти, які не лише виконують свої функції, а й приносять задоволення від взаємодії. Поєднання даних та дизайнерського мислення допомагає уникнути розриву між тим, чого потребують користувачі, і тим, що пропонують розробники.

Метою цієї роботи є опис основних принципів продуктової аналітики та аналіз способів їх впровадження у процес проєктування інформаційних систем. Під час дослідження будуть розглянуті ключові аспекти інтеграції аналітичних даних у дизайн, а також приклади їх використання для створення більш ефективних продуктів.