Плечистий Д. Д., к.т.н., доцент,
Войтюк О. В., аспірант
Державний університет «Житомирська політехніка»

БАГАТОЗАЛЕЖНІ ЗВ'ЯЗКИ У РЕКУРСИВНІЙ СТРУКТУРІ ФРОНТЕНД ДОДАТКУ

Побудова рекурсивного дерева елементів є однією із задач для кожного фронтенд розробника і потребує багато часу для аналізу та пошуку правильного рішення. Рівень складності такої задачі полягає у виборі правильних інструментів розробки, коректності їх налаштувань та реалізації складних програмних структур. Рівень складності підвищується еквівалентно збільшенню вхідних умов задачі які визначають фінальний набір функціональності додатку. Одним з таких умов є підтримка багатозалежності зв'язків між різними елементами. Прикладом може бути процес онлайн заповнення форми адреси для відправки пошти: результат вибору країни має автоматично змінити поле для вибору міста, що у свою чергу змінює набір опцій по вибору вулиці. Цей приклад є видом лінійної залежності, тому що усі елементи розташовані у логічній послідовності і під час розробки додатку будуть знаходитись у одній зоні видимості. Популярні випадки залежностей часто знаходяться на одному й тому самому рівні вкладеності і легко можуть бути пов'язані між собою. Водночас гарантією таких випадків є відсутність рекурсивної побудови додатку або спрощена архітектура вхідних об'єктів даних. Проблема багатозалежних зв'язків полягає у можливості існування одного або більше факторів впливу на елемент взаємодії з користувачем, тобто, декілька елементів додатку можуть змінити видимість або поведінку іншого елементу, який може існувати в іншому просторі видимості даних.

Для вирішення проблеми зв'язків у рекурсивних деревах елементів треба вибрати інструмент, який дозволить спостерігати за станом інших елементів та оновлювати цю інформацію на різних рівнях вкладеностей, а не тільки на самому вищому рівні. Вбудовані можливості React дозволять працювати з таки поняттям як «форми», що і є рішенням поставленої проблеми, однак ці можливості обмежені у функціональності та потребують більше часу і знань для впровадження складного рішення.

Проаналізувавши низку інструментів, які можуть легко інтегруватись у складну структуру зв'язків, було знайдено бібліотеку яка володіє зручною системою підписки на зміни даних – це Formik. Основними перевагами цієї бібліотеки є:

- **Спрощений код**: можливість написати менше коду для управління станом форми, обробки подій та валідації. Він надає компактний та зручний інтерфейс для роботи з формами.
- **Управління станом**: автоматично вирішує питання стану форми та обробки подій, дозволяючи зосередитися на логіці додатка, а не на деталях форм.
- **Інтеграція з валідацією**: легко інтегрується з валідацією форм, дозволяючи визначити правила валідації для кожного поля та миттєво отримувати звіти про помилки.
- **Вбудовані компоненти введення**: має вбудовані компоненти введення, такі як **<Field>**, які допомагають зробити код більш читабельним та лаконічним.
- **Форми на основі об'єктів**: Formik працює з формами, що базуються на об'єктах. Це дозволяє зберігати всі значення форми в одному об'єкті, спрощуючи управління станом.
- **Підтримка асинхронних операцій**: легко інтегрується з асинхронними операціями, такими як відправка форми на сервер, дозволяючи легко обробляти такі сценарії.
- **Можливість власних реалізацій**: якщо потрібно, Formik надає можливість власних реалізацій для деяких частин форми, забезпечуючи гнучкість для специфічних вимог.

У підсумку, можна сказати, що обґрунтований вибір технології спостереження за зв'язками пришвидшить вирішення основної проблеми оптимізації промалювання ВЕБ застосунку, а малий поріг входження у таку технологію зменшить кількість часу на побудову складного рішення.

Список використаних джерел

1. Titus Kamunya «8 Best React Form Libraries for Developers» URL: https://geekflare.com/best-react-form-libraries/
2. Jared Palmer «Formik dosc» URL: https://formik.org/
Jordan Walke «React Form» URL: https://legacy.reactjs.org/docs/forms.html