[5] Horizontal Fall Arrest System: https://www.elcosh.org/document/2029/d000288/Horizontal+Fall+Arrest+System.html

І.А. Хлиста, А.П. Бондарчук*
аспірант ННІ ІТ Державний університет інформаційно-комунікаційних технологій, Київ, Україна, ivan.khlysta@gmail.com
*д.т.н., професор Державний університет інформаційно-комунікаційних технологій, Київ, Україна, dekan.it@ukr.net

Вирішення проблеми часткового оновлення вмісту веб-сторінки шляхом оптимізації параметрів SPA

Анотація - В тезах розглядаються особливості оптимізації односторінкових вебдодатків як спосіб покращення механізму часткового оновлення html сторінки в браузері. Актуальність застосування оптимізації як покращення користувацького досвіду в контексті щорічного зростання кількості активних вебдодатків. Наводиться порівняльний приклад оптимізації клієнтської частини вебдодатків з використанням фреймворку Angular та бібліотеки React. Позитивний ефект оптимізації вебсторінки підтверджується шляхом порівняння основних показників продуктивності додатку до та після оптимізації на основі метрик Chrome Dev Tools Performance та PageSpeed Insights. Висновки підтверджують актуальність дослідження та стисло відображають досягнуті результати.

Для покращення показників продуктивності, ефективності та конкурентоспроможності будь-якого додатку, а тим паче вебдодатків необхідно проводити комплексну оптимізацію. Головним завданням розробника під час розростання додатку є підтримка не лише якості написаного коду, а й стабільності позитивного користувацького досвіду шляхом постійного покращення згаданих параметрів.

Постановка проблеми: використання техніки часткового оновлення вмісту вебсторінки в односторінкових (від англ. SPA - односторінковий додаток) додатках застосовується сьогодні в більшості технологій для клієнтської розробки та сприймається користувачами як цілком стандартний аспект користувацького досвіду, тобто як щось очікуване. Проте, якщо не зважати на механізми,

* науковий керівник

відповідальні за взаємодію із цією технікою в платформах розробки та постійну оптимізацію, обслуговування проекту, то можна швидко завдати шкоду як додатку, так і погіршити користувацький досвід.

Виходячи з цього, **метою викладеної роботи** є покращення користувацького досвіду під час взаємодії з вебдодатками шляхом підвищення продуктивності додатку завдяки оптимізації його параметрів.

Основні **завдання**, які висуваються для подальшого дослідження включають у себе таке:

- З'ясування змісту поняття техніки часткового оновлення вмісту вебсторінки в браузері.
- Дослідження проблеми використання техніки під час функціонування односторінкових додатків на базі фреймворку Angular та бібліотеки React.
- Аналіз параметрів від яких залежить продуктивність вебдодатку в цілому.
- Порівняння механізмів Change Detection та Virtual Dom.
- Визначення підходів, які можуть бути використані для оптимізації вебдодатку для досягнення оптимальної продуктивності.

Аналіз **останніх наукових** досліджень показує, що під технікою часткового оновлення вмісту вебсторінки варто розуміти певний набір засобів реагування на зміни виду (від англ. view - вид) статичної вебсторінки під час перегляду та взаємодії із нею користувача. Часткове оновлення змісту вебсторінки є сукупністю використання одночасно декількох технологій, серед яких провідну роль відіграє технологія AJAX, завдяки якій сучасна клієнтська частина не повинна більше завантажувати усю отриману від веб-сервера інформацію одним пакетом (технології лінивого/модульного завантаження lazy loading/AMD) та технологія SPA, яка завантажує лише одну html сторінку в браузері та відповідно оновлює її, аналізуючи дії користувача. Оскільки SPA не використовуються в "чистому" виді, а представляють конкретні платформи для розробки (такі як фреймворк Angular та бібліотека React.js), то відповідно процес оновлення змі-

* науковий керівник

сту виду сторінки набуває нового значення, оскільки уже не є загальним, а безпосередньо залежить від особливостей обраної розробником платформи.

Використання згаданих вище стратегій управління механізмами перевірки зміни виду сторінки та її перебудови (Change Detection для фреймворку Angular та Virtual Dom для React.js) є найбільш ефективним способом оптимізації додатку та підвищення його продуктивності.

Перевірка змін виду сторінки у фреймворці Angular відбувається під час порівняння механізмом Change Detection вихідної html сторінки та її стану після взаємодії із користувачем. Change Detection містить декілька стратегії управління процесом перевірки зміни стану, якими може керувати розробник. Як правило це - Default та onPush стратегії. Якщо Default стратегія автоматизує перевірку, а саме запускає її щоразу, коли фіксуються зміни чи то внаслідок кліку користувача, чи завантаження даних, то стратегія onPush дозволяє розробнику вирішувати, коли запускати процес перевірки стану. Оскільки це місткий та ресурсозатратний процес, то чим рідше його запускати, тим більше ресурсів можна заощадити.

Використання Change Detection-у по своїй суті ніби “передає” кожній компоненті маленький шматочок механізму для аналізу її структури та відповідного, оперативного реагування на ситуацію. Результати аналізу реагування додатку розробленого із використанням onPush методу виявились на 25-30% швидшими в порівнянні з показниками методу Default за рахунок скорочення непотрібних перевірок стану щоразу, коли користувач взаємодіє із додатком.

У бібліотеці React.js існує схожий механізм функціонування, який називається Virtual Dom, суть якого так само полягає в порівнянні структури DOM-у вебсторінки при початковій ініціалізації та при наступних змінах. Однак в React.js не існує стратегій управління, оскільки доступу до самого Virtual Dom-у не надається і єдиним, що залишається робити розробникам це оптимізувати код за рахунок застосування додаткових інструментів, таких як наприклад чітке використання атрибута key для допомоги механізму розпізнавати зміну струк-

* науковий керівник

тури дерева DOM-у, використання принципів “чистих” функцій та компонентів, а також контролювати локальний стан останніх.

Враховуючи, що обидві технології грунтуються на ООП-підходах до розробки, а отже містять потенційно слабкі місця у вигляді класових компонентів та необхідності розуміння розробником життєвих циклів компонентів як передумови успішної розробки, то ми наголошуємо на важливості інструменту тестування, як попередження небажаної поведінки компонентів або додатку в цілому.

Не менш важливими для оптимізації є також правильне застосування загальних паттернів програмування для розробки додатку. Серед них наприклад концепція запам’ятовування (від англ. memoization - запам’ятовування) вхідних даних в межах компонентів додатку. Її використання дозволяє відмовитись від оновлення певної частини виду сторінки, за яку в додатку відповідає певний компонент, в разі, якщо вхідні дані є незмінними, просто порівнюючи їх із попередніми вхідними даними, збереженими локально.

React.js реалізує цю концепцію завдяки наявності в його функціоналі хуків (від англ. - hook - гачок) useMemo та useCallback, у той час, як Angular значною мірою спирається на механізм Change Detection.

Дотичною до мемоізації є тема кешування, однак в контексті оптимізації клієнтської частини доречніше згадати про застосування локального сховища та інструментів браузера, які піддаються оптимізації за рахунок використання HTTP заголовків (якщо комунікація між сервером та клієнтом встановлена на основі REST API), а також Service Workers.

Насамкінець підкреслюємо важливість правильно спроектованої архітектури додатку для подальшої безболісної оптимізації та запровадження розглянутих в дослідженні рекомендацій.

За результатами дослідження, вдалось виявити оптимальні підходи до управління змінами та визначити найбільш ефективні тактики для виконання перебудови виду сторінки. У дослідженні порівнюються різні стратегії оптимізації продуктивності та оцінюється їх вплив на швидкодію та ефективність кліє-

* науковий керівник

нтської частини вебдодатків, ідентифікуються чинники, які негативно впливають на їх продуктивність в контексті найбільш популярних платформ для клієнтської розробки Angular та React.js.

Наприкінці дослідження показники продуктивності вебдодатків, отриманих під час проведення дослідження були протестовані на основі метрик Chrome Dev Tools Performance та PageSpeed Insights. Як наслідок, показники продуктивності відповідно до категорії FCP та FP зросли на 16.1% та 14.3% та 14.7% і 13.2% для Angular та React додатків відповідно, у той час, як в категорії TTI відбувся ріст на 10% та 9%. Показник TBT знизився на 4.4%.

Підсумовуючи, можемо дійти висновку, що в першу чергу правильний підхід до оптимізації техніки часткового оновлення вмісту вебсторінки є вихідною точкою для максимальної продуктивності клієнтської частини вебдодатку, за яким звичайно слідують і інші спеціальні платформо-орієнтовані особливості.

Перспективи подальших досліджень можуть включати:

Дослідження інших методів оптимізації продуктивності вебдодатків, які доповнюють управління механізмами перевірки зміни виду сторінки, таких як використання Web Workers.

Поглиблене дослідження впливу різних факторів на продуктивність клієнтських веб-додатків та розробка стратегій оптимізації для кожного з цих факторів. Наприклад, дослідження можуть розглядати вплив обсягу даних, складності компонентів, частоти змін даних та інших факторів на продуктивність.

Розробка документації, посібників та навчальних матеріалів, які допоможуть розробникам ефективно використовувати розглянуті стратегії для оптимізації продуктивності своїх додатків.

Загалом, подальші дослідження в цій сфері можуть сприяти появі більш ефективних та продуктивних підходів до розробки вебдодатків, а також поліпшенню користувацького досвіду та задоволенню від використання додатків з боку кінцевих користувачів.

Список використаних літературних джерел

* науковий керівник

[1] Angular essentials, Kumar D. (2019) (1st ed.). Packt Publishing.

[2] Mastering Angular Components - Second Edition, G. Kunz. (2018). [Online]. Available: https://learning.oreilly.com/library/view/mastering-angular-components/9781788293532/4f31314e-3dc3-4de0-b3d2-408fdfafc032.xhtml.

[3] Springer, S (2021). React. Das umfassende Handbuch (1. Nachdruck). Rheinwerk Verlag, Bonn. ISBN 978-3-8362-6877-6

[4] Stefanov, S (2021). React: Up & Running: Building Web Applications, O'Reilly Media; 2nd edition. ISBN: 978-1492051466

В.Д. Ляшко, Б.О. Бекас [*], Р.В. Івасюк[*]
магістрант кафедри КН НЛТУ України, Львів, Україна,
lyashko.v@nltu.lviv.ua.
[*]старший викладач кафедри ІСКМ, НЛТУ України, Львів, Україна,
bohdan.bekas@nltu.edu.ua, https://orcid.org/0000-0002-3571-9600
[3] асистент кафедри ІПЗ, НЛТУ України, Львів, Україна, ivasyuk@nltu.edu.ua.

Аналіз програмного комплексу інституційного репозиторію

Анотація - У роботі проведено порівняльний аналіз популярних систем та програмного забезпечення для створення репозитарію університету. Розглянуто основні функціональні особливості системи DSpace з метою вивчення можливостей їх використання в університеті. Охарактеризовано основні елементи та концепції моделі системи DSpace, переваги й недоліки використання програмного продукту. Обґрунтовано доцільність використання систем DSpace.

Ключові слова – інституційний репозитарій, цифрова бібліотека, електронний архів, сервіс, системи управління репозиторіями.

Поняття електронної, цифрової, віртуальної бібліотеки, електронного архіву, інституційного репозитарію нині не є усталеними, і їх досить часто використовують як синоніми. Враховуючи специфіку бібліотек наукових і вищих навчальних закладів (ВНЗ), науковці [1] вважають доцільним створення інституційного репозитарію. Під терміном «інституційний репозитарій» розумітимемо не лише сховище електронних наукових матеріалів наукової установи або вищого навчального закладу, а й сервіс, який надається своїм працівникам для

* науковий керівник