КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ

Vol. 2, No. 1, 2020

https://doi.org/10.23939/csn2020.01.033

УДК 004.021; 004.383

ПІДВИЩЕННЯ ШВИДКОСТІ РОБОТИ ВЕБ-ДОДАТКІВ

Ю. С. Клушин, Ю. Б. Захарчин

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин

Односторінкові веб-додатки – це технологія веб-додатка, яка складається з однієї веб-сторінки, яка взаємодіє з користувачем, динамічно генеруючи поточну сторінку, а не завантажує цілі нові сторінки з сервера. Наведено методику створення веб-додатка на основі SPA технології (односторінковий веб-додаток) як метод підвищення швидкості роботи веб-додатків на основі використання сучасних фреймворків, інструментів та засобів розроблення клієнтської та серверної частини односторінкового веб-додатка. На основі цієї методики розроблений власний веб-додаток і на його основі визначена швидкість відгуку, яка є меншою ніж оптимальна швидкість відгуку для односторінкових веб-додатків. Дано роз'яснення, завдяки яким рішенням збільшується швидкість відгуку та продуктивність роботи в односторінковому веб-додатку та чому створення багатосторінкового сайту є не найкращою ідеєю.

Ключові слова: односторінковий веб-додаток, база даних, багатосторінковий веб- додаток, нереляційна база даних, реляційна база даних, Backend технології, сервер, JavaScript.

Вступ

Веб-технології стрімко розвиваються, багатосторінкові сайти стають неактуальними, адже швидкість та продуктивність таких сайтів є дуже низькою. Така низька швидкість пов'язана із вічним «діалогом» користувача із сервером, адже на кожну дію користувача з сервера запитується нова сторінка і відбувається повне перезавантаження сайту, навіть якщо зміни були незначними. На зміну їм прийшли швидкі, динамічні, інтерактивні, нелегкі в проектуванні і розробленні односторінкові «сайти», але завдяки великій кількості логіки як на клієнтській, так і на серверній частинах їх можна називати повноцінними веб-додатками. Саме в цей момент перед розробниками постало питання підвищення швидкості і продуктивності веб-додатків, адже користувачі стають все більш нетерплячими та вимогливими, тому неправильно обрані методи та інструменти розроблення веб-додатка можуть призвести до провального проекту, який залишиться поза увагою клієнтів [1].

Одним із методів підвищення швидкості роботи веб-додатка є проектування односторінкового веб-додатка (SPA) – це інтерактивний веб-додаток, який унаслідував поведінку нативних програм. Цю технологію розроблено в момент збільшення популярності використання в веб-сайтах мови програмування *JavaScript* та появи *AJAX* технології, адже саме можливість звернення до сервера 'на льоту', не перезавантажуючи сторінку, та динамічна зміна змісту сторінки і є основою технології односторінкових додатків.

Односторінкові додатки завантажують сторінку зі сервера лише один раз, а всі інші операції відбуваються за рахунок асинхронних *AJAX* запитів *і JavaScript* скриптів. Важливою концепцією

технології є те, що значна частина логіки додатка виноситься на користувацьку сторону, тобто, при першому завантаженні додатка, з серверу надходить коренева сторінка і багато *JavaScript* скриптів і всі ці дані розміщуються на стороні користувача (в браузері користувача). Процес завантаження першої сторінки може бути тривалим, оптимальна швидкість завантаження односторінкових веб-додатків становить 5 секунд, але залежно від кількості даних, які сервер має надіслати при першому завантаженні та можливостей користувацького ПК, це значення може змінюватись. Перевагою односторінкових веб-додатків над традиційними багатосторінковими є те, що після завантаження всіх даних із серверу додаток працює дуже швидко, адже більше не витрачається час на ресурсозатратні запити на сервер. Усі інші дані надходять із сервера в універсальному форматі *JSON*, і із цих запитів динамічно генерується інтерфейс додатка. Важливо зазначити, що динамічна генерація інтерфейсу відбувається без перезавантаження сторінки, що приємно при роботі із веб- додатком, адже складається враження роботи із звичайним нативним додатком.

Побачивши перспективи розвитку технології односторінкових веб- додатків, нею зацікавились та почали активно розвивати такі компанії-гіганти, як Google та Facebook.

Розглянувши роботу сьогодні відомих соціальних мереж, можна побачити, як відбувався перехід із традиційних багатосторінкових сайтів на односторінкові веб-додатки. Взявши за приклад сервіс Google “GMAIL”, де завантажувалась перша сторінка сайту із даними, у цьому випадку повідомленнями для того, щоб дізнатись, чи прийшло нове повідомлення, користувачу необхідно було перезавантажити сторінку, відбувався запит на сервер, і незалежно від того, було нове повідомлення чи ні, сервер повертав абсолютно нову сторінку. Це було дуже незручно, тому компанія Google вирішила застосувати односторінкову технологію на своєму продукті, робота якого виглядає так: при першому завантаженні веб-додатка зі сервера надходить коренева сторінка та багато *JavaScript* коду, який розміщується на стороні клієнта, тобто в браузері. Генерується перша «Головна» сторінка. Тепер якщо користувачу надходить нове повідомлення, йому не потрібно перезавантажувати сторінку, – це відбувається автоматично, адже передбачено *AJAX* запити (запити на 'льоту'), які робляться із певною періодичністю, якщо нове повідомлення надійшло, в веб-додатку без перезавантаження сторінки, динамічно та точково формується інтерфейс сторінки, тобто, з сервера повернеться не нова сторінка, а дані в *JSON* форматі, які підставляться в відповідну компонету та динамічно змінять інтерфейс сторінки.

Оскільки для створення односторінкового веб-додатка необхідно прикласти значних зусиль, для спрощення розробки компанії Google та Facebook створили власні фреймворки, бібліотеки та інструменти для створення односторінкових веб-додатків. Використання запропонованих ними засобів розроблення не лише значно пришвидшують розроблення веб-додатка, але й дають змогу проектувати ще більш високошвидкісні та зручні в використанні додатки [1, 2].

Використання фреймворків під час проектування веб-додатка значно спрощує життя розробникам: мала кількість коду порівняно із традиційними багатосторінковими сайтами та можливість повторного використання коду, що неможливо під час розроблення багатосторінкового сайту. Цей підхід називається компонентним, сторінку веб-додатка ділять на маленькі незалежні частинки, які за потреби мають можливість інтегруватись в будь-яку частину додатка.

Сильною стороною технології створення односторінкових веб-додатків, є кросбраузерність та мобільність. Тобто, легка переносимість коду на мобільну платформу, оскільки для передавання даних із сервера використовується універсальний формат *JSON*. Це дає змогу легко переносити додаток на мобільну платформу, майже не змінюючи серверної частини додатка завдяки тому, що серверна та клієнтські частини чітко розділені, потрібно просто за допомогою інших інструментів – таких як *React Native* – створити спроектовану під мобільну платформу клієнтський інтерфейс.

Односторінкові веб-додатки мають слабку безпеку та SEO оптимізацію: для підвищення безпеки веб-додатка необхідно залучати сторонні інструменти, які допомагають протидіяти хакерським атакам. Також дуже важко налаштувати SEO оптимізацію, але ця проблема не є суттєвою, адже SEO оптимізація більше потрібна для товарних сайтів чи сайтів-візитівок.

Аналіз останніх досліджень та публікацій

Для того, щоб зрозуміти, завдяки чому збільшується швидкодія додатка, побудованого за допомогою SPA-технології, потрібно розглянути технологію, за допомогою якої додатки розроблялись раніше, – це так звана традиційна *Multi Page Application* (MPA) технологія. Ці дві технології являють собою веб-додатки, якими ми користуємось на наших девайсах, але їхня поведінка суттєво відрізняється.

Односторінковий додаток (SPA) – це сучасніший підхід до розроблення додатків, який використовують компанії-гіганти Google, Facebook, Twitter.

SPA – це програма, яка працює всередині браузера і не потребує перезавантаження сторінок під час його завантаження.

Багатосторінковий додаток (MPA) вважають традиційним підходом до розроблення додатків. MPA – це програма, яка вимагає перезавантаження сторінки кожного разу, коли вміст сторінки змінюється. Кожного разу, коли дані обмінюються від браузера до серверу, в сервера запитується абсолютно нова сторінка для відображення у веб-браузері. Ця операція потребує дуже багато часу для створення сторінок на стороні сервера, – ця технологія називається Server Side Render, тобто, сервер віддає браузеру вже готову сторінку.

Багатосторінкові додатки мають класичну архітектуру. Кожна сторінка надсилає запит на сервер і повністю оновлює дані, навіть якщо вони незначні. Так втрачаються продуктивність і швидкість роботи додатка.

Кожна технологія має свої переваги та недоліки.

Перевагами багатосторінкових додатків є:

- Проста SEO оптимізація. Архітектура MPA дозволяє легко оптимізувати кожну сторінку для пошукових систем;
- легкість розробки. Розроблення багатосторінкового додатка не потребує використання великої кількості технологій, чим суттєво полегшує розробку.

Недоліками технології є:

- непереносимість додатка на мобільну платформу. Для того, щоб перенести додаток на мобільну платформу, серверну частину додатка потрібно створити заново;
- велика залежність *fronend p*озробників від *backend* розробників.

Односторінкові веб-додатки це – сучасна технологія веб-додатка, яка складається з однієї веб-сторінки, яка взаємодіє з користувачем, динамічно генеруючи поточну сторінку, а не завантажує цілі нові сторінки з сервера [1].

Такий підхід дозволяє уникнути переривання користувацької роботи, змушуючи додаток мати поведінку настільної програми. У SPA-коді весь необхідний код *HTML, CSS та JavaScript* – отримується з завантаженням однієї сторінки або динамічно підвантажується та додається на сторінку, зазвичай у відповідь на дію користувача. Ця функція можлива завдяки *AJAX* [2, 3}, який дозволяє клієнту взаємодіяти із сервером без перезавантажень сторінки.

SPA-адреси швидші, ніж традиційні веб-додатки, оскільки вони виконують логіку в веб-браузері, а не на стороні сервера – цей підхід називається *User Side Render*. Після початкового завантаження сторінки браузер обмінюється з сервером лише даними, а не всією *HTML*-сторінкою, що суттєво збільшує пропускну здатність.

Безумовно, технологія є новою, але і вона має свої переваги та недоліки.

Перевагами технології є:

- висока швидкість. Оскільки SPA не оновлює всю сторінку, а лише точково змінює її частину, це значно підвищує швидкість роботи;
- швидкість розроблення. Існує безліч готових бібліотек та фреймворків, які допомагають створювати веб-додатки;
- незалежність розроблення. Над проєктом можуть паралельно працювати розробники *backend* частини та *frontend* частини. Адже технологія розділила розроблення цих двох частин додатка;

- легка переносимість на мобільну платформу. Оскільки для передавання даних з сервера використовують універсальний формат *JSON*, це дає змогу легко переносити додаток на мобільну платформу, майже не змінюючи серверної частини додатка.

Недоліками технології є:

- погана оптимізація SEO. SPA працює на базі *JavaScript* та завантажує інформацію на запит із клієнтської частини. Пошукові системи не можуть змоделювати цієї поведінки. Тому більшість сторінок недоступні для сканування пошуковими роботами;
- неактивний JavaScript. Деякі користувачі відключають *JavaScript* у своїх браузерах, а без нього додаток не може функціонувати.

Якщо порівняти між собою SPA та MPA технології за швидкістю [4,5], то:

– SPA-додаток завантажується швидше. Він завантажує більшість своїх ресурсів лише один раз. Сторінка не завантажується повністю, коли користувач запитає навий фрагмент даних;
– MPA-додаток завантажується повільніше, адже браузер повинен перезавантажити всю сторінку з нуля, коли користувач хоче отримати доступ до нових даних або перейти на іншу частину веб-додатка.

Багатосторінковий додаток запитує нову *HTML*-сторінку з сервера після кожної дії користувача в додатку. Це призводить до обробки багаточисельних запитів між клієнтською та серверними частинами.

Односторінковий, своєю чергою, використовує *AJAX*-технологію, що дозволяє оновлювати тільки частину додатка, замість цілого веб-додатка [3].

SPA відправляє *AJAX*-запит на сервер та повертає дані в форматі *JSON* із сервера.

Односторінковий додаток (SPA) – це сучасніший підхід до розроблення додатків, його використовують компанії-гіганти Google, Facebook, Twitter. SPA – це програма, яка працює всередині браузера і не потребує перезавантаження сторінок під час його завантаження.

Технологія SPA може надати майже безмежні можливості; в цьому дуже легко переконатись, – потрібно просто глянути на різноманітні додатки, якими ми користуємось майже кожного дня.

Одними із перших, хто побачив переваги SPA-додатків над традиційними MPA, стала компанія Google, – вони створили безліч сервісів на основі SPA-технології.

Компанії-гіганти, такі як Google та Facebook, дуже популяризували технологію, адже під свої проекти створили зручні інструменти розроблення – різноманітні фреймворки, такі як *Angular ma React*, які значно спростили створення односторінкових додатків [6–8].

Постановка задачі

Метою статті є методіка створення веб-додатка на основі SPA-технології (односторінковий веб-додаток) як метод підвищення швидкості роботи веб-додатків на основі використання сучасних фреймворків, інструментів та засобів розроблення клієнтської та серверної частин односторінкового веб-додатка. На основі цієї методіки розробити власний веб-додаток і на його основі продемонструвати швидкість відгуку, яка повинна бути меншою або дорівнювати оптимальній швидкості відгуку односторінкових веб-додатків, а також пояснити, завдяки яким рішенням збільшується швидкість відгуку та продуктивність роботи в односторінковому веб-додатка та чому створення багатосторінкового сайту є не найкращою ідеєю.

Алгоритм створення веб-додатка на основі SPA-технології

Створюючи власний веб-додаток, необхідно було детально дослідити технологію односторінкових веб-додатків, ознайомитись із існуючими аналогами односторінкових веб-додатків. Ознайомитись із найпопулярнішими фреймворками, бібліотеками та інструментами для проєктування односторінкових веб-додатків. На основі їх аналізу обрати найоптимальніші засоби розроблення. Для розроблення односторінкових веб-додатків рекомендується використовувати один із

запропонованих стеків розроблення *MERN або MEAN*. Перший характерний тим, що для розроблення користувацького інтерфейсу використовують бібліотеку *React* як базу даних *MongoDB* та для серверного розроблення *Node.js* та *Express.js*. Стек *MEAN* відрізняється лише тим, що для розроблення користувацького інтерфейсу використовується фреймворк *Angular* [9]. Використання одного із вищезгаданих стеків у розробці є дуже доцільним, адже базовою мовою програмування для них всіх є мова *JavaScript*, що економить час, який необхідно було б витратити на вивчення інших мов програмування [10–11]. За їхньою допомогою можна створити власний швидкий односторінковий веб-додаток, який в ідеальному середовищі розробки дає швидкість відгуку 5 секунд. Ця величина може відрізнятись в умовах експлуатації продукту, оскільки швидкість відгуку також залежатиме від пропускної здатності мережі, швидкості мережі та можливостей персонального комп'ютера чи смартфона.

На основі проведеного аналізу та досліджень, для розроблення користувацького інтерфейсу вибираємо:

1. *Java Script* бібліотеку *React* [12], яка ідеально підходить для створення односторікових вебдодатків, адже він відображає контентні зміни сторінки без перезавантаження поточної сторінки. Використання React сумісно з іншими бібліотеками, дає безмежні можливості розроблення;
2. *Node.js* для серверної частини;
3. *Express.js* фреймворк;
4. *mongoDB* нереляційна база даних.

Для реалізації проекту використовуємо стек *MERN*, який на всіх рівнях стеку використовує *JavaScript* та використання *API RESTful*, основною властивістю якої є проектування розподілених систем за допомогою обмежень. Основні обмеження виглядають так:

- Клієнт-серверна архітектура;
- Взаємодія без збереження стану;
- Логічний інтерфейс додатка.

Розглянемо алгоритм створення *SPA-додатка* за допомогою стеку *MERN*.

1. Встановлення програмного забезпечення *Node.js*.
2. Створення серверної частини додатка.

– Налаштування сервера та залежностей.
– Розроблення програмного забезпечення сервера та його запуск.
– Створення бази даних.
– Організація маршутів.
– Додавання об'єктів до бази даних та тестування сервера

3. Створення користувацького інтерфейсу:

– налаштування *webpack*;
– запуск проекту;
– встановлення додаткових залежностей та структура даних;
– інтеграція *backend в frontend;*
– результати створення користувацького інтерфейсу.

Встановлення програмного забезпечення

Як серверну технологію *React* використовуємо *Node.js*, який потрібно встановити за посиланням https://nodejs.org/ [13]. Після встановлення *Node.js* рекомендовано переконатись в успішному встановленні та перевірити їх версії. Всі решта залежностей можна буде встановлювати згодом у міру необхідності завдяки менеджеру пакетів *npm* – додатка для управління пакетами.

Створення серверної частини додатка

Архітектуру серверної частини показано на рис. 1. Налаштовують сервер та залежності так.

Для запуску *JavaScript* коду в *backend* частині необхідно запустити сервер, який компілюватиме код. Сервер можна створити двома методами:

1. Використати вбудований модуль *http* у вузлі.

2. Використати *Express.js.*

У розробленому веб-додатку використовується *Express.js*, це *HTTP* модуль *Node.js*, для створення повнофункціонального *API RESTful*. Він потребує дуже мало коду, для встановленя пакетів *express* використовую команду *npm install express*.

Щоб встановити всі необхідні залежності, використовують команду *npm install express body-parser cors mongoose nodemon*, яка означає таке:

- *Express* – серверний фреймворк;
- *body parser* – відповідає за те, щоб вимкнути body із запиту;
- nodemon – перезавантажує сервер коли він побачить зміни;
- *cors* – пакет проміжного програмного забезпечення;
- *mongoose* – для моделювання об'єктів *MongoDB* для *Node.js.*

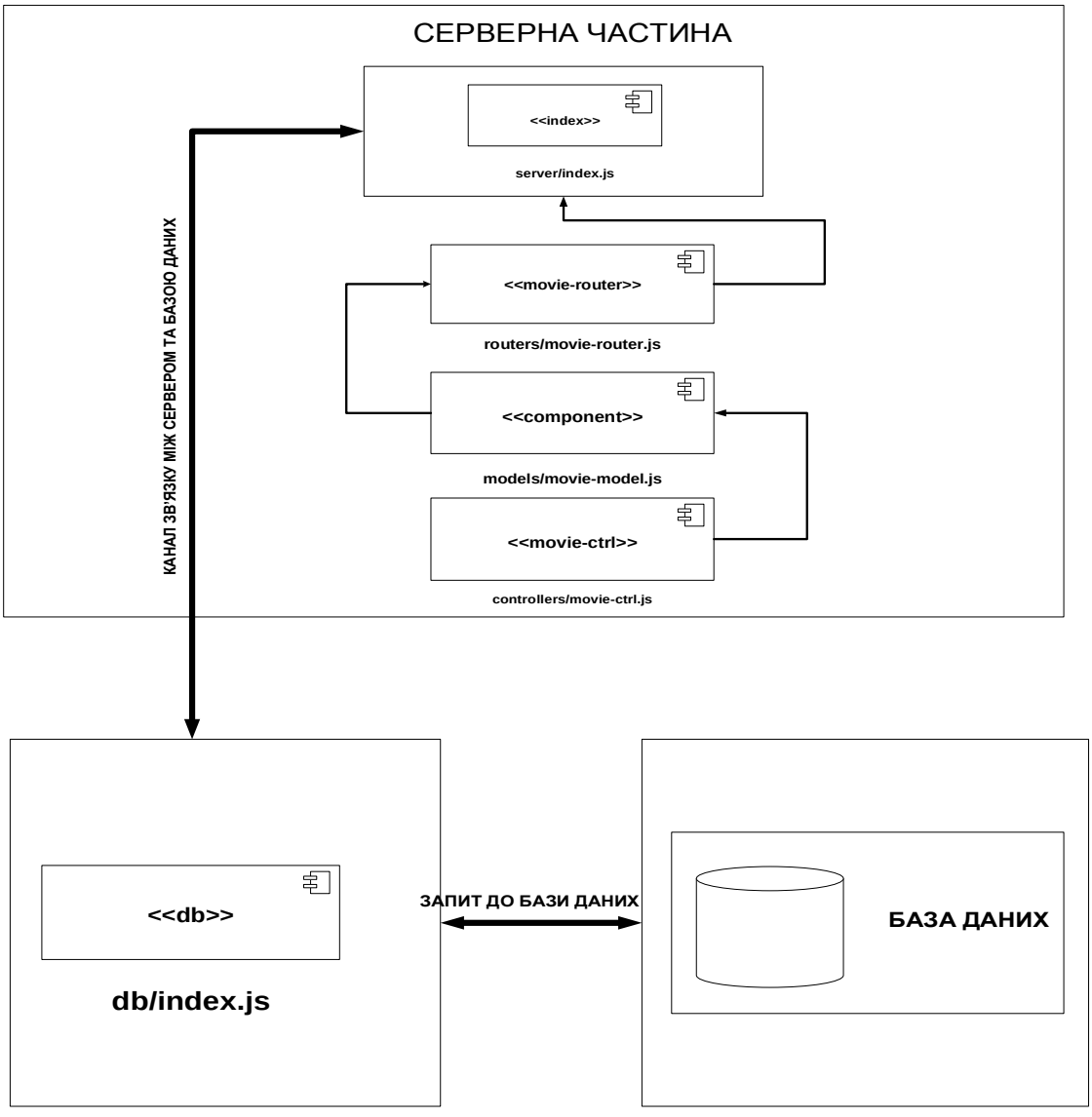

Рис. 1. Архітектура серверної частини

Розроблення програмного забезпечення сервера та його запуск

Встановивши всі необхідіні залежності, можна створювати *Node.js* файл, який запустить сервер на порті 3000. На цьому етапі розробки створюється серверне програмне забезпечення, командою *node index.js* запускається сервер, у випадку успішного виконання в консолі буде виведено повідомлення:

Server running on port 3000.

Створення бази даних

Для створення бази даних використовуємо нереляційну базу даних *mongoDB.* Для початку роботи необхідно встановити *mongoDB* за таким посиланням: https://www.mongodb.com/download-center/community

Після успішного встановлення необхідно запустити *mongoDB с*ервер командою *mongod*.

Наступним етапом створюємо базу даних з назвою **cinema.**

Після успішного створення бази даних її необхідно поєднати з сервером, для чого створюється дерикторія *db/,* в якій міститиметься база даних, а за допомогою *mongoose* бібліотеки встановлюємо з'єднання. Об'єкт фільму для бази даних матиме такий вигляд:

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema
const Movie = new Schema(
{
name: { type: String, required: true },
time: { type: [String], required: true },
rating: { type: Number, required: true },
},
{ timestamps: true },
)
module.exports = mongoose.model('movies', Movie)
```

Цей об’єкт є шаблонним об’єктом.

Створення маршутів та операції *CRUD* та кінцеві точки *REST* має такий вигляд:

```
const express = require(’express')
const MovieCtrl = require(’../controllers/movie-ctrl')
const router = express.Router()
router.post('/movie', MovieCtrl.createMovie)
 router.put('/movie/:id', MovieCtrl.updateMovie)
 router.delete('/movie/rid', MovieCtrl.deleteMovie)
 router.get('/movie/rid', MovieCtrl.getMovieById)
 router.get('/movies', MovieCtrl.getMovies)
module.exports = router]
```

Для тестування та додавання об'єктів до бази даних використовуємо інструмент *MongoDB Compass.*

Для тестування необхідно додати кілька об'єктів до бази даних. Варто наголосити, що об’єкти створюються в форматі *JSON*:

_id: 0bjectld("5eb9742882S4ee22fc983a2l“)

name: "Avengers: Endgame"

> time:Array

rating:8.8

_id: 0bjectld("5eb9743d82S4ee22fc9S3a22")

name:"Avengers: Endgame"

> time: Array rating:8.8

_id: Objectld("5eb9745l82S4ee22fc9S3a23")

name: "The Lord Of The Rings: The return of the king"

> time: Array rating:8.9

Створення користувацького інтерфейсу. Налаштування webpack

Для налаштування середовища розробки використовуємо утиліту *create-react- app*, яка налаштовує початкову структуру проєкту, скачує необхідні залежності і налаштовує webpack.

Create-react-app не опрацьовує бекенд-логіку чи логіку баз даних, а лише надає команди для користувацького інтерфейсу, тому його можна сміло використовувати з будь-якою серверною технологією. Це одна із дуже важливих концепцій односторінкових веб-додатків, незалежність і повна ізоляція серверної та клієнтських частин. Це дозволяє одночасно працювати двом розробникам над різними задачами, а не очікувати один одного, а в подальшому легко поєднати клієнтську та серверну частини за допомогою API.

Ця утиліта створює таку систему файлів:

• *node_modules/* містить всі зовнішні бібліотеки *JavaScript*, які використовуються додатком;

• директорія *public/* містить базові файли HTML, JSON і зображень. Це кореневі ресурси проекту, саме в цій директорії зберігається базова сторінка *HTML* і вона виступатиме як коренева для проєкту. Вона є основою запуску проєкту і основною частиною *React* додатку;

• директорія *src/* містить код *React JavaScript* для проєкту. Переважно саме в цій директорії і ведеться вся робота;

• файл *.gitignore* містить кілька директорій і файлів за замовчуванням, які система контролю *git* ігноруватиме, зокрема і директорію *node_modules;*

• *README.md* – це файл розмітки, який містить багато корисної інформації про додаток *create-react-app*, зокрема огляд команд і посилання на розширені опції конфігурацій.

Запуск проєкту

Щоб запустити проект, запускаємо сервер командою *yarn start*, після чого в браузері автоматично запуститься стандартний односторінковий *React* веб-додаток.

Після успішного запуску сервера, в браузері запускається односторінковий веб-додаток, і з яким тепер можна працювати та будувати його згідно із задуманою архітектурою, яку зображено на рис. 2.

Рис. 2. Архітектура клієнтської частини

Результати створення користувацького інтерфейсу

Результатом створення користувацького інтерфейсу є зручний та швидкий односторінковий додаток, функціонал якого полягає в додаванні та видаленні кіносеансів (рис. 3).

Видалення та додавання елементів на сторінку відбувається без перезавантаження сторінки, перехід на інші сторінки додатка, також не спричиняють перезавантаження сторінки, весь контент додатка автоматично генерується в тому випадку, яку *DOM* структура зазнала якихось змін.

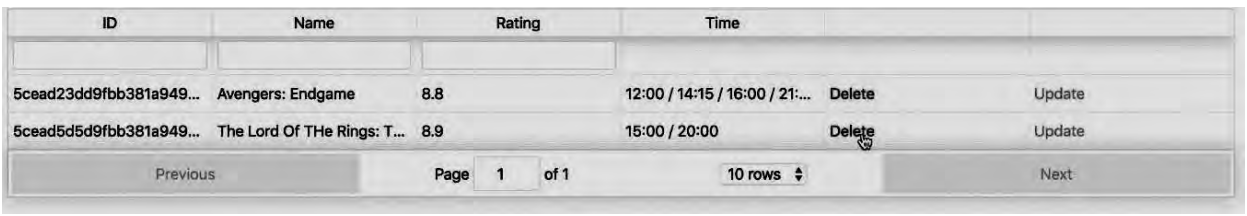

Рис. 3. Сторінка додавання та видалення полів

У додатку існує сторінка всього списку сеансів з можливістю відредагувати, тобто видалити неактуальні сеанси.

За результатом тестування швидкості роботи односторінкового веб-додатка середній час завантаження головної сторінки склав 2,24 секунд, що є значно менше ніж оптимальний відгук односторінкових веб-додатків, який становить 5 секунд, та на 0,25 секунд більше за мінімальний час відгуку кореневої сторінки додатка. Час завантаження першої сторінки традиційного багатосторінкового додатка 7,08 секунд.

Порівнюючи отримані дані, варто зазначити, що після першого завантаження односторінкового веб-додатка тривалих запитів більше не буде, як і перерендерингу сторінки. У традиційних багатосторінкових сайтах час завантаження завжди буде більшим.

Висновки

Розглянуто методику підвищення швидкості роботи веб-додатка. Ця методика полягає в використанні технології SPA. Особливість цієї технології полягає в одній кореневій сторінці, яка завантажується з сервера при першому завантаженні веб-додатка, надалі інтерфейс сторінки формується динамічно, завантажуючи із сервера лише дані в форматі *JSON*. Ще однією особливістю SPA-технології, яка суттєво підвищує швидкість веб-додатка, є перенесення значної частини коду на сторону клієнта, що істотно збільшує швидкість та продуктивність роботи додатка та зменшує навантаження на сервер. Також у статті поетапно досліджено процес створення власного односторінкового додатка з використанням обраного фреймворку *React* як користувацького інтерфейсу *Node.js* та фреймворку *Express.js* для серверної розробки та *MongoDB* для бази даних. Надано алгоритм встановлення необхідного програмного забезпечення та розроблено власний швидкий односторінковий додаток. Результат тестування швидкості роботи створеного односторінкового веб-додатка: середній час завантаження головної сторінки 2,24 секунди, що є значно менше за оптимальний відгук односторінкових веб-додатків, який становить 5 секунд, та на 0,25 секунди більше за мінімальний час відгуку кореневої сторінки додатка.

Отже, технологія SPA суттєво збільшує швидкість роботи та продуктивність додатка. За підтримки компаній Google та Facebook вона ставатиме ще популярнішою.

Список літератури

1. Black C., Building a Single Page Web Application with Knockout.js / Black C., Ly D. Packt Publishing, 2014. 152 p.

2. Monteiro F., Learning Single-page WebApplicationDevelopment / Monteiro F. – Packt Publishing, 2014. 214 p.

3. Mikovski M., Development of one-page web applications / Mikovski M. DMK Press, 2014. 512 p.

4. Brown E., Web Development with Node and Express / Brown E. O`Really, 2017. 336 p.

5. Young A., Meck B., Cantelon M., Node.js in Action / Young A., Meck B., Cantelon M. Manning, 2018. 432 p.

6. Chinnathambi K., Learning react / Chinnathambi K. – Addison Wesley, 2019. 368 p.

7. Tielens M. T, React in Action / Tielens M. T. Manning, 2019. 368 p.

8. Banker K., Bakkum P., Shaun V., Hawkins T., MongoDB in Action / Banker K., Bakkum P., Shaun V., Hawkins T. Manning, 2016. 482 p.

9. Official site of AngularJS. – Access mode https://angularjs.org/ Access date: 13.05.2020.

10. Seshardi S., AngularJS: Up and Running. / Seshadri S., Green B. – O'Reilly Media, 2014. 322 p.

11. Herrington J., Learning AngularJS. / Herrington J. – Packt Publishing, 2015. 235 p.

12. Official ReactJS website. – Access mode https://uk.reactjs.org/Access date: 13.05.2020.

13. The official website of Node.JS. Access mode https://nodejs.org/en/Access date: 13.05.2020.

INCREASE THE SPEED OF WEB APPLICATIONS

Y. Klushyn, Y. Zakharchin

Lviv Polytechnic National University,
Computer Engineering Department

The article presents a method of creating a web application based on SPA technology (one-page web application), as a method of increasing the speed of web applications based on the use of modern frameworks, tools and tools for developing client and server part of a one-page web application. One-page web applications are web application technologies that consist of a single web page that interacts with the user, dynamically generating the current page rather than downloading entire new pages from the server. Based on this technique, we developed our own web application and based on it we determined the response rate, which is less than the optimal response rate for single-page web applications. An explanation is given as to which solutions increase response speed and performance in a one-page web application, and why creating a multi-page site is not the best idea.

Keywords: single-page web application, database, multi-page web application, non-relational database, relational database, Backend technologies, server, JavaScript.