УДК 004.738.5

Шкаруба А.В., здобувач
Житомирський державний університет імені Івана Франка

ОНЛАЙН СЕРВІС ДЛЯ ДИНАМІЧНОЇ ВІЗУАЛІЗАЦІЇ ДАНИХ

В епоху стрімкого зростання обсягів цифрової інформації, здатність швидко аналізувати дані й робити правильні висновки з них стає надзвичайно важливим для прийняття швидких рішень в багатьох галузях. Це пов'язано з тим, що вони безперервно надходять від багатьох джерел, наприклад від систем моніторингу за промисловими об'єктами; сенсорів, що керують життєдіяльністю будівель; програм для опрацювання фінансових звітностей тощо. Звичайно ж текст або набір таблиць не дуже сприяють усвідомленому опрацюванню їх людиною. Тож розробка інтерактивних онлайн-сервісів, здатних візуалізувати у динаміці інформацію, є досить актуальним завданням. Основний проблемний момент при розробці таких вебсистем полягає в тому, щоб представляти графічно числові залежності у зручному для користувача вигляді і при цьому зберігати зручність роботи користувача з великими обсягами вхідних даних досить складно.

У цьому контексті, **метою** дослідження є описати ефективну програмну архітектури для спеціалізованого веб-сервісу, який забезпечує швидке завантаження, попередню обробку та інтерактивне відображення динамічної графічної інформації.

Для досягнення поставленої мети було представлено наступну клієнт-серверну архітектуру системи, що чітко розділяє процеси на обробку даних та їх фінальну візуалізацію.

Зазвичай основним джерелом вхідної інформації для сервісу виступають структуровані текстові формати, зокрема файли формату CSV. Також треба зважати й на те, що пряме завантаження та обробка таких числових масивів відбувається безпосередньо на стороні клієнта, що призводить до перевантаження оперативної пам'яті та зависання сторінки браузера. Щоб цього уникнути, варто зробити основний акцент на правильному розподілі обчислювального навантаження. Весь процес первинної обробки варто перенести на серверну частину. Вона може бути реалізована на базі середовища Node.js. У такому випадку сервер здійснює асинхронне зчитування файлів, не блокуючи роботу системи. Після цього він фільтрує інформацію і перетворює її у структурований формат (JSON), який є оптимальним для передачі по мережі.

Важливою складовою надійності системи є використання суворої типізації за допомогою TypeScript. Завдяки цьому на етапі підготовки даних на сервері формуються чіткі об'єкти, які відповідають очікуванням та вимогам клієнтського інтерфейсу. Такий підхід дозволяє браузеру не витрачати ресурси та час на складну обробку, перевірку чи сортування файлу. Натомість клієнтська частина миттєво розпочинає відображення графічної інформації одразу після отримання готового пакета даних від сервера. [2]

Отримавши підготовлений та типізований масив даних, клієнтська частина приділяє увагу виключно на візуалізацію та забезпечення взаємодії з користувачем. Оскільки браузер звільнений від великих обчислень, інтерфейс може швидко реагувати на дії користувача і при цьому відображати складні залежності між наборами даних. Це дозволяє реалізувати плавні анімації, зручне масштабування окремих ділянок графіка для детального аналізу, а також динамічне виведення додаткової інформації при наведенні курсору на конкретні точки. Крім того, така оптимізація процесу відображення значно зменшує вимоги до обчислювальної потужності пристрою користувача. Завдяки цьому з аналітикою можна працювати не лише на сучасних комп'ютерах, але й на звичайних офісних ноутбуках чи мобільних пристроях.[1]

Крім того, така модульна архітектура дозволяє підвищити гнучкість та масштабованість розробленого онлайн-сервісу. У разі виникнення потреби підключити нові джерела інформації або обробляти ще більші обсяги даних, то основні зміни будуть лише на сервері. Клієнтський інтерфейс при цьому не буде потребувати великих змін, що робить створену систему універсальним інструментом, простим у підтримці та подальшій модернізації.

Підводячи короткий **підсумок** зазначимо, що запропонований архітектурний підхід дозволяє ефективно вирішити проблему відображення великих масивів інформації у веб-середовищі. Чіткий розподіл завдань між сервером та браузером, перенесення важкої обробки CSV-файлів на серверну сторону (Node.js) та використання суворої типізації даних забезпечують швидку, стабільну та зручну роботу веб-сервісу. У майбутньому планується розширити функціонал системи, додавши нові типи графіків та можливість автоматичного оновлення інформації в режимі реального часу.

Список використаних джерел:

1. Mardan A. Full Stack JavaScript: Learn Backbone.js, Node.js and MongoDB. - New York: Apress, 2015. - 287 p.

2. Zammetti F. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. - Berkeley: Apress, 2020. - 374 p.