НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПОЛТАВСЬКА ПОЛІТЕХНІКА
ІМЕНІ ЮРІЯ КОНДРАТЮКА

ЗБІРНИК МАТЕРІАЛІВ

77-Ї НАУКОВОЇ КОНФЕРЕНЦІЇ ПРОФЕСОРІВ, ВИКЛАДАЧІВ, НАУКОВИХ ПРАЦІВНИКІВ, АСПІРАНТІВ ТА СТУДЕНТІВ УНІВЕРСИТЕТУ

16 травня – 22 травня 2025 р.

УДК 004.9

О.О. Демочко, студент групи 401-ТК,
Науковий керівник О.Г. Климко, старший викладач
Національний університет
«Полтавська політехніка імені Юрія Кондратюка»

СТВОРЕННЯ ВИСОКОПРОДУКТИВНОГО ПОШУКОВОГО СЕРВІСУ ЗА ДОПОМОГОЮ NEXT.JS, ZUSTAND І FASTAPI

У сучасному світі, у зв'язку з розвитком інформаційних технологій, зростає потреба у високопродуктивних веб-сервісах для роботи з великими обсягами даних, зокрема у пошукових системах, котрі мають здатність швидко і точно знаходити релевантну інформацію. Робота присвячена створенню високопродуктивного пошукового сервісу з використанням Next.js, Zustand і FastAPI, що забезпечує ефективний пошук в умовах великої кількості складних запитів та даних.

Постановка задачі. Розробити та впровадити веб сервіс пошуку, що здатний обробляти великий обсяг одночасних запитів з високою швидкістю.

Мета роботи. Застосування комбінацій передових технологій Next.js (для фронтенду), Zustand (для управління станом на клієнтській стороні) та FastAPI (для створення високопродуктивного бекенду) для розробки та впровадження веб сервісу пошуку, що здатний обробляти великий обсяг одночасних запитів з високою швидкістю.

Розробка включає в себе використання передових технологій для забезпечення швидкої обробки запитів, ефективного зберігання та управління даними на клієнтській стороні та високої продуктивності на сервері.

1. Next.js: Використано для розробки інтерфейсу пошукового сервісу. Завдяки своїм можливостям серверного рендерингу та статичного сайту (SSR & SSG) Next.js дозволяє зменшити час завантаження сторінок і покращити взаємодію користувача з системою.

2. Zustand: Застосовано для управління станом на клієнтській стороні. Завдяки простоті використання та можливості легко інтегрувати з React, Zustand дозволяє швидко оновлювати стан без зайвих витрат на перерендеринг компонентів, що покращує продуктивність при обробці результатів пошуку.

3. FastAPI: Використано для створення бекенд-системи. FastAPI дозволяє створювати асинхронні API, що суттєво зменшує час обробки запитів і збільшує пропускну здатність системи.

Розробка сервісу включала порівняння двох версій серверів, які працюють з веб-клієнтом на базі Next.js та мобільним додатком. Перший

сервер використовував Flask, який був переписаний на FastAPI для підвищення продуктивності та зниження латентності. Це дозволило порівняти два стеки за показниками продуктивності.

1. Flask стек: Використовував бібліотеки: Flask, marshmallow, sqlalchemy, flask-migrate, flask-jwt-extended та інші, характерні для Flask.

2. FastAPI стек: Після переходу на FastAPI стек змінився на більш сучасний і включає FastAPI, pydantic, sqlalchemy, alembic, та інші відповідні бібліотеки для FastAPI.

Для порівняння продуктивності двох серверів був використаний інструмент autocannon, який дозволяє провести нагрузочне тестування API. Тестування проводилось на одній робочій станції, щоб порівняти продуктивність у реальних умовах. Хоча сервери ще знаходяться на стадії оптимізації, результати тестів демонструють значну перевагу FastAPI в порівнянні з Flask, особливо при великих навантаженнях і збільшенні кількості одночасних запитів.

У процесі розробки було проведено кілька етапів тестування:

1. Тестування часу обробки запитів: Порівняння продуктивності сервісів на основі Next.js та традиційних технологій показало значне зменшення часу обробки запитів. Система з FastAPI обробляла запити на 30-40% швидше за Flask.

2. Тестування пропускної здатності: FastAPI сервер зміг обробляти до 10,000 запитів на хвилину при максимальному навантаженні, що є відмінним показником для систем, що обробляють великий обсяг даних.

3. Тестування точності результатів пошуку: Завдяки асинхронній обробці запитів на бекенді та оптимізованому управлінню станом на клієнтській стороні, точність і швидкість пошуку значно покращилися, що було підтверджено тестами на реальних даних.

Висновки

Розроблений пошуковий сервіс продемонстрував високу продуктивність і здатність ефективно обробляти великі обсяги запитів. Використання Next.js, Zustand та FastAPI дозволило створити систему, яка не лише швидко реагує на запити користувачів, але й масштабована для великих проектів. Результати тестування показали, що перехід на FastAPI дозволив значно підвищити продуктивність сервісу, зокрема завдяки асинхронній обробці запитів. Ці технології виявилися ефективними для створення високопродуктивних веб-сервісів, здатних працювати з великими обсягами даних і обробляти великі навантаження.

Література:

1. FastAPI Documentation. [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com

2. Zustand Documentation. [Електронний ресурс]. – Режим доступу: https://github.com/pmndrs/zustand

3. Next.js Documentation. [Електронний ресурс]. – Режим доступу: https://nextjs.org/docs