На рисунку 3 можна побачити результат – усі 4 задачі були успішно заплановані на наступний день без накладень. Задача «second task» не мала зазначених годин виконання, тому була поставлена на початок дня. Задача «first task» мала інтервал з 12 по 13 годину, і, оскільки у цей день інтервал був вільним, задача була запланована на цей час без проблем. Аналогічним чином задач «third task», яка мала зазначені години та день. Задача «task four» не могла бути запланована після «first task», оскільки, за умовою, потрібно мати інтервал у 30 хв після задачі, а у такому випадку, перед «third task» не лишиться часу для цього інтервалу, тому вона була запланована після задачі «third task».

Часова складність алгоритму інтервального планування – O(n log n), де n представляє кількість інтервалів. Завдяки своїй ефективності алгоритм знаходить широке застосування в різних сценаріях планування та розподілу ресурсів. Даний алгоритм за бажанням можна доповнювати іншими керованими змінними: наявністю дедлайна, пріоритетністю задач, ресурсами.

Застосування цього підходу до планування полегшує управління завданнями, забезпечуючи відповідний розподіл ресурсів і своєчасне виконання завдань. Ці процеси додатково сприяють збалансованому плануванню та контролю виконання роботи, та мінімізації ризиків пов'язаних із затримками та невиконанням завдань. На основі запропонованого алгоритму розроблено зручний веб-застосунок для індивідуального планування, доступний за посиланням https://lotus-todo-app.web.app.

Список літературних джерел

1. *Brucker, P.: Scheduling Algorithms, 2nd edn. Springer, Heidelberg (2007)*
2. *The Ultimate Guide to Personal Productivity Methods. [Електронний ресурс]. Режим доступу – https://todoist.com/inspiration/personal-productivity-methods*
3. *Planning and Scheduling: Definitions and Differences. [Електронний ресурс]. Режим доступу – https://www.indeed.com/career-advice/career-development/planning-and-scheduling*

УДК 004.428 : 004.514

Ярош О. Л., студент 1 курсу
спеціальності 122 «Комп'ютерні науки» СО Магістр
Бабаков Р. М., д.т.н., доцент,
доцент кафедри інформаційних технологій

МЕТОДИ ОПТИМІЗАЦІЇ ПРОДУКТИВНОСТІ ВЕБ-ЗАСТОСУНКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

Веб-інтерфейси API (Application Programming Interfaces) слугують основою сучасних веб-додатків, забезпечуючи взаємодію між різними програмними компонентами. Тому їхня продуктивність безпосередньо впливає на загальну продуктивність веб-додатків. У цій статті розглядаються основні

методи оптимізації продуктивності внутрішніх веб-інтерфейсів та їхній вплив на користувацький досвід і SEO.

Методи оптимізації продуктивності веб-застосунки:

1. Внутрішня оптимізація продуктивності веб-інтерфейсів охоплює різні стратегії, які допомагають зменшити затримки, підвищити ефективність і збільшити масштабованість веб-додатків:
2. Кешування: Впровадження відповідних стратегій кешування може значно зменшити час відгуку та навантаження на сервер, що призведе до швидшого відгуку API.
3. Оптимізація бази даних: Оптимізація запитів до бази даних, індексування та використання представлень бази даних може значно скоротити час обробки даних на сервері.
4. Балансування навантаження: Розподіл мережевого трафіку між декількома серверами може гарантувати, що жоден сервер не стане вузьким місцем в продуктивності.
5. Обмеження швидкості: Запобігає перевантаженню сервера занадто великою кількістю запитів одночасно.
6. Ефективне використання кодів стану HTTP: Правильне використання кодів стану HTTP може підвищити ефективність зв'язку між клієнтом і сервером, сприяючи швидшому усуненню несправностей і вирішенню проблем.

Оптимізація продуктивності внутрішніх веб-інтерфейсів API може значно покращити взаємодію з користувачем. Швидкі відповіді сервера призводять до швидшого завантаження сторінок, підвищуючи задоволеність і залученість користувачів. Високопродуктивні веб-API також можуть впоратися з більшим користувацьким навантаженням, забезпечуючи безперебійну роботу користувачів навіть у пікові моменти.

Ефективна продуктивність внутрішнього веб-застосунку відіграє важливу роль у SEO. Пошукові системи враховують час завантаження сторінок при ранжуванні веб-сайтів, а ефективні API сприяють швидшому завантаженню сторінок. Таким чином, оптимізовані веб-застосунки можуть опосередковано підвищити рейтинг сайту в пошуковій видачі.

Важливість оптимізації продуктивності внутрішніх веб-застосунку у майбутньому тільки зростатиме зі збільшенням складності веб-додатків. Йдучи в ногу з розвитком веб-технологій і регулярно оптимізуючи веб-інтерфейси, ви можете гарантувати, що веб-сайти продовжуватимуть надавати користувачеві першокласний досвід.

Внутрішні методи оптимізації продуктивності веб-інтерфейсів мають першорядне значення у формуванні користувацького досвіду і підвищенні SEO. Оскільки цифровий ландшафт продовжує розвиватися, глибоке розуміння цих методів та їх ефективне застосування залишатиметься наріжним каменем успішної веб-розробки.

Окрім традиційних методів оптимізації веб-застосунків, для підвищення продуктивності можна використовувати й інші сучасні методи:

1. Архітектура мікросервісів[1]: Декомпозиція великого застосунку на менші, слабко пов'язані між собою сервіси може підвищити масштабованість і сприяти більш ефективному використанню ресурсів.
2. Рендеринг на стороні сервера: Для деяких додатків попередній рендеринг сторінок на сервері може пришвидшити початковий час завантаження сторінки та покращити сприйняття продуктивності для користувачів.
3. HTTP/2: Впровадження[2] HTTP/2 може підвищити продуктивність завдяки таким функціям, як стиснення заголовків, визначення пріоритетів і мультиплексування.

Моніторинг продуктивності внутрішніх веб-застосунків має вирішальне значення для виявлення вузьких місць і їх оперативного усунення. Такі інструменти, як New Relic[3], Datadog[4] та AWS CloudWatch[5] забезпечують моніторинг та діагностику продуктивності в режимі реального часу. Ці інструменти можуть запропонувати цінну інформацію про продуктивність API, допомагаючи розробникам виявляти та вирішувати проблеми з продуктивністю.

Продуктивність внутрішніх веб-застосунків має значний вплив на показники залучення користувачів. Повільні API можуть призвести до вищого показника відмов, нижчої активності користувачів і меншої кількості конверсій. І навпаки, оптимізовані внутрішні API можуть покращити утримання та залучення користувачів, що призведе до підвищення коефіцієнта конверсії.

Хоча основні веб-показники Google Core Web Vitals в першу чергу зосереджені на продуктивності фронтенду, продуктивність бекенду відіграє важливу роль у показнику Largest Contentful Paint (LCP), одному з трьох основних показників Core Web Vitals. Ефективні внутрішні API забезпечують швидкі відповіді сервера, які сприяють швидшому LCP, що позитивно впливає на SEO.

Підсумовуючи, у сфері веб-розробки оптимізація продуктивності бек-енд веб-застосунків є такою ж важливою, як і фронтенд-оптимізація. Вона відіграє важливу роль у покращенні користувацького досвіду, покращенні SEO та загальному успіху веб-застосунки.

У міру того, як Інтернет розвивається в напрямку більш інтерактивних і складних додатків, важливість ефективних бекенд API буде продовжувати зростати. Тому веб-розробники та компанії повинні приділяти першочергову увагу оптимізації продуктивності внутрішніх веб-інтерфейсів API, щоб залишатися конкурентоспроможними в цифровому середовищі.

Список літературних джерел

1. *Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media. (Discusses the concept of Microservices Architecture)*
2. *Belshe, M., Peon, R., and Thomson, M. (2015). Hypertext Transfer Protocol Version 2 (HTTP/2). IETF. (Describes the HTTP/2 protocol and its features)*
3. *New Relic, Inc. (2021). New Relic: Full-Stack Observability.*
4. *Datadog, Inc. (2021). Datadog: Cloud Monitoring as a Service.*
5. *Amazon Web Services, Inc. (2021). Amazon CloudWatch: Monitor Resources and Applications.*