УДК 004.9

# Определение узких мест серверного веб-приложения и их рефакторинг в рамках разработки Конструктора ОП

**Коряков С. А.** (Университет ИТМО), **Говоров А.И.** (Университет ИТМО)
**Научный руководитель – кандидат технических наук, доцент Хлопотов. М. В.**
(Университет ИТМО)

**Введение.** Разработка веб-приложений – это комплексный процесс, включающий в разработку множество архитектурных элементов системы. В зависимости от сложности и набора подобных элементов возрастает и общее количество узких мест у приложения – чем больше информации обменивается между клиентом и сервером, а также внутри этих приложений, тем больше вероятность обнаружить низкоэффективные программные модули и решения [1]. На примере разработки Конструктора ОП представлен обзор методов определения узких мест системы и методики улучшения производительности проблемных элементов серверного приложения.

**Основная часть.** Серверная часть веб приложения взаимодействует с множеством других элементов архитектуры системы. Обнаружение «узких мест» в таком случае зачастую осуществляется анализом проблемных взаимодействующих с сервером компонентов – зачастую первичное выявление низкопроизводительных элементов системы производится в ходе нагрузочного тестирования и пользовательского взаимодействия приложением - во втором случае не все проблемы могут касаться серверной части напрямую. Первичное определение узких мест тесно завязано на комплексном исследовании всей системы, зачастую такие элементы выявляются при реальном взаимодействии с разработкой путем анализа нагрузки наиболее используемых частей веб-приложения.

Конструктор ОП — это веб-приложение, обеспечивающее сотрудников университета инструментарием для взаимодействия с рабочими программами дисциплин, ГИА и практик, учебными планами и образовательными программами. Серверная часть приложения написана на языке Python с использованием библиотек Django и Django REST и СУБД PostgreSQL. При анализе производительности сервиса были выявлены узкие места проекта, относящиеся к серверной части приложения – это загрузка страниц учебных планов, компетенций внутри РПД и матриц компетенций. Также проблемными являются модули загрузки файлов учебных планов и матриц компетенций.

В рамках подобного исследования на наличие мест, влияющих на производительность продукта, использовались следующие подходы, локализующие исследуемый элемент и выявляющие характер уменьшения скорости работы системы:

1) Измерение и анализ запросов, уходящих на сервер, выявление TTFB [2]
2) Определение мест блокировки основного потока серверного приложения на момент простоя I/O [3]
3) Выявление количественных и качественных характеристик, связанных с взаимодействием с другими архитектурными элементами системы (скорость передача данных, количество запросов к БД)
4) Анализ использования лишних сущностей в рамках реализации программного элемента

Каждый критерий позволяет более точно определить проблемные элементы системы, выявить находятся ли они на стороне сервера и определить причину возникновения такого узкого места. Зачастую, локализация происходит поэтапно, следуя списку, описанному выше.

Ход анализа проблемных мест в конструкторе ОП происходил следующим образом:

1) На стороне клиента, замерялась скорость отправки запроса – все описанные элементы системы имели долгое время ответа от сервера – более 10 секунд. Проверялось, что проблема находилась именно на стороне сервера по параметру TTFB
2) Через режим отладки выявлялось время выполнения и количество всех внутренних методов внутри представления, отвечающего за запрос.

3) С помощью сторонних библиотек анализировалось количество запросов к БД и скорость их выполнения.
4) Определены лишние классы и поля таблиц, запрашиваемые внутри API-запросов.

Такое исследование помогло выявить конкретные места приложения, где происходила потеря скорости обработки запроса – ввиду состава учебного плана с использованием рекурсивной системы модулей, получение его элементов, находящихся на нижнем уровне иерархии сопровождалось многочисленными однотипными вызовами рекурсии и запросами к Базе данных. Количество запросов многократно увеличивалось из-за необходимости динамического подсчета трудоемкости т.к. при запросе каждого модуля необходимо было получать все его нижестоящие элементы – связано это со способом хранения трудоемкости в сервисе. При запросе рабочих программ дисциплин в составе УП запрашивалась цепочка лишней информации, связанной с компетенциями для рабочих программ путем выполнения многократных запросов. В местах загрузки файлов проблемной зоной является вызов внутренних обработчиков библиотеки для преобразования excel-файлов.

На примере модуля отображения учебного плана для устранения причин низкой скорости его загрузки были использованы следующие программные и архитектурные решения:
1) Использование select_related и prefetch_related для сокращения количества запросов в рамках типовой проблемы N+1 [4]
2) Замена вызова программной рекурсии технологией CTE для БД
3) Перенос нагрузки вычисления трудоёмкости при реализации модуля с записью в БД
4) Удаление лишних полей, запрашивающие дополнительные данные

Таким образом, количество запросов к базе данных для отдельного учебного плана сократилось на 80-90%, а скорость загрузки для типовых объектов стала составлять не более чем 3-7 сек.

**Выводы.** В рамках работы были описаны методы обнаружение мест, негативно влияющих на производительность серверной части веб-приложения на примере Конструктора ОП. Были выявлены подобные места, локализованы проблемы, связанные с производительностью и на примере отдельного модуля системы описан метод разрешения проблемы. После реализации подобных средств были рассчитаны относительные характеристики, описывающие прибавку к скорости загрузки компонента системы.

**Список использованных источников**:

1. Юлия Рагунович. Узкие места производительности веб-приложений, или куда смотреть, чтобы ничего не пропустить: презентация доклада на XXXIV Международной конференция по вопросам качества программного обеспечения. // CUSTIS. Заказные информсистемы. URL: https://shorturl.at/mnsv4 (дата обращения: 05.02.2024).

2. Jeremy Wagner, Barry Pollard. Time to First Byte (TTFB) // Web.Dev. URL: https://web.dev/articles/ttfb (дата обращения: 05.02.2024).

3. Paul Hazell. Improving Python Performance in an I/O-Bound Application // Better Programming. URL: https://betterprogramming.pub/improving-python-performance-in-an-i-o-bound-application-redis-cluster-interactions-9e3288bca0e8

4. Adam Johnson. Django and the N+1 Queries Problem // Scout APM. URL: https://scoutapm.com/blog/django-and-the-n1-queries-problem