

Е. Н. Яковлев¹✉, А. А. Шарапов¹

Разработка системы аналитики и визуализации данных для веб-платформы управления научными конференциями

¹Сибирский государственный университет геосистем и технологий,
г. Новосибирск, Российская Федерация
e-mail: jakovlev_igor@mail.ru

Аннотация. В статье предложена и обоснована распределенная микросервисная архитектура на базе асинхронной среды FastAPI и реактивного фреймворка Vue.js. Описан механизм внутридоменного сбора телеметрии, опирающийся на алгоритм генерации цифровых отпечатков и полностью исключающий зависимость от cookie-файлов. Детально рассмотрен подход к снижению вычислительной нагрузки на сервер приложений путем делегирования процессов агрегации логов на уровень реляционной СУБД PostgreSQL. Данная методика позволила реализовать мгновенный рендеринг тепловых карт активности и воронок конверсии без ущерба для основного бизнес-ядра. Приведенные метрики тестирования подтверждают высокую отказоустойчивость предложенного решения в условиях интенсивного конкурентного трафика.

Ключевые слова: микросервисная архитектура, фоновая телеметрия, асинхронные интерфейсы, агрегация на уровне СУБД

E. N. Yakovlev¹✉, A. A. Sharapov¹

Development of an analytics and data visualization system for a web-based scientific conference management platform

¹Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
e-mail: jakovlev_igor@mail.ru

Abstract. This article proposes and justifies a distributed microservice architecture based on the asynchronous FastAPI environment and the reactive Vue.js framework. It describes a mechanism for intra-domain telemetry collection that relies on a fingerprinting algorithm and completely eliminates the need for cookies. The approach to reducing the computational load on the application server by delegating log aggregation processes to the PostgreSQL relational DBMS is examined in detail. This method enabled the implementation of instant rendering of activity heatmaps and conversion funnels without compromising the core business logic. The testing metrics presented confirm the high fault tolerance of the proposed solution under conditions of intense competitive traffic.

Keywords: microservices architecture, background telemetry, asynchronous interfaces, aggregation at the DBMS level

Введение

Эволюция цифровых сред академического взаимодействия демонстрирует выраженный дисбаланс между функциональностью платформ приема статей и инструментами оценки их эффективности. Практика проведения научных мероприятий выявляет критическую уязвимость порталов при массовой загрузке отчетных материалов в преддверии дедлайнов. В такие моменты административ-

ному аппарату жизненно важно иметь доступ к объективной картине происходящего для оперативного масштабирования вычислительных мощностей.

Следствием этого становится необходимость проектирования и имплементации глубоко интегрированного, суверенного контура аналитики. Цель настоящего исследования заключается в создании независимой телеметрической подсистемы на основе микросервисной архитектуры и механизма бескуки-идентификации. Такая подсистема должна функционировать в скрытом режиме и обеспечить полный охват аудитории научного портала без снижения производительности его основного бизнес-ядра.

Исследование предметной области

Традиционный ландшафт университетских информационных систем представляет собой преимущественно монолитные решения. Их архитектура изначально не рассчитана на агрессивные флуктуации пользовательского трафика, когда подавляющий объем транзакций, связанных с выгрузкой объемных бинарных файлов (текстов докладов), концентрируется в узком временном коридоре.

Очевидным шагом к мониторингу подобных ситуаций кажется интеграция общедоступных метрических скриптов (коммерческих систем веб-аналитики). Однако глубокий анализ применимости таких решений в образовательном секторе выявляет два непреодолимых препятствия.

Прежде всего, отправка навигационных логов и метаданных исследователей в транснациональные ИТ-компании идёт вразрез с действующими в вузах правилами защиты конфиденциальной информации и политикой цифрового суверенитета. Далее, повсеместное внедрение пользователями клиентских расширений для фильтрации контента сводит на нет любые попытки объективного сбора статистики. Эвристические алгоритмы блокировщиков рекламы с высокой точностью идентифицируют и прерывают HTTP-соединения с доменами известных трекеров. Это искусственно занижает показатели вовлеченности на величину, порой превышающую сорок процентов [1, 2].

Следовательно, использование внешних облачных решений для аналитики научных платформ является нецелесообразным, что формирует четкий технический запрос на разработку децентрализованных, внутридоменных модулей телеметрии, не зависящих от сторонней инфраструктуры.

Методы и технологии

В основу архитектурного каркаса заложен принцип API Composition, предполагающий дробление экосистемы на узкоспециализированные домены [3, 4].

Основное бизнес-ядро, управляющее графами связей между конференциями, секциями и авторами, делегировано программной среде, ориентированной на строгую валидацию транзакций [5]. Параллельно с ним был развернут исследуемый аналитический узел. Для его конструирования использован инструмент FastAPI, чей выбор аргументируется не просто данью современным трендам, а строгой математической целесообразностью [6]. Реализация протокола ASGI позволяет данной части переводить операции ввода-вывода в разрешаю-

щий режим, что показывает возможность сервера поддерживать в оперативной памяти большое количество незавершённых соединений от браузеров без приостановки основного цикла обработки событий [7, 8]. Это, в свою очередь, обеспечивает непрерывность выполнения запросов.

Для обхода клиентских фильтров безопасности скрипт-перехватчик не подключался как внешний ресурс, а был скомпилирован нативно – прямо в ядро фронтенд-приложения Single Page Application. Из-за данного подхода эвристические анализаторы браузеров воспринимают его как часть стандартной логики сайта. Сигнал на отправку лога генерируется после того, как навигационный хук завершил свою работу, что отсекает ложные срабатывания и фиксирует только фактические переходы по страницам.

Главной сложностью подобного скрытого трекинга является необходимость различать пользователей, не нарушив их приватности. По той причине, что подход с применением файлов cookie ведет за собой серьезные юридические риски, от них пришлось отказаться. Идентификация построена на базе алгоритма локальной криптографической маркировки сессий [9].

Для обеспечения правильной работы метрики «Средняя продолжительность визита» и проблем, возникающих из-за длительного простоя открытой вкладки в браузере (вплоть до нескольких недель), в алгоритм был добавлен механизм пассивной инвалидации, который позволяет отсеивать незначимые данные и повышать точность статистики. При каждом действии скрипт сверяет разницу во времени с предыдущим кликом. Если она превышает определенное количество минут, то старый ключ стирается из памяти, а пользователь начинает считаться новым.

Для передачи пакетов на телеметрический узел используется браузерный API асинхронной диспетчеризации. Главное преимущество данного метода заключается в гарантированной доставке пакета в фоновом режиме, что позволяет минимизировать потери данных телеметрии в пограничных сценариях работы веб-приложения. На принимающей стороне происходит парсинг и обогащение логов. HTTP-заголовок, содержащий сигнатуру программного обеспечения клиента, проходит через лексический анализатор, который получает следующие данные: семейство ОС, ядро браузера и форм-фактор устройства, которые на следующем шаге передаются в реляционную таблицу [10].

Самым уязвимым местом аналитической системы является построение дашбордов по массивам из огромного количества записей. Выгрузка миллионов строк из базы данных в оперативную память Python-приложения приводит к отказу в обслуживании. Проблема решена смещением вычислений на уровень базы данных PostgreSQL [11]. Сложные SQL-запросы с использованием обобщенных табличных выражений и фильтров HAVING позволяют ядру СУБД самостоятельно отсеивать «сессии-отказы», в которых отсутствует активность пользователей на странице [12].

Таким образом, был использован следующий технологический стек:

- язык программирования Python, асинхронный фреймворк FastAPI;

- объектно-реляционная система управления базами данных PostgreSQL с логическим разделением контуров на уровне схем;
- JavaScript-фреймворк Vue.js 3 с использованием паттернов реактивного состояния, а также библиотека Chart.js для программного рендеринга дашбордов;
- протоколы межсервисного взаимодействия на базе Axios, маршрутизация пакетов Nginx, а также механизмы криптографической верификации на основе стандарта JSON Web Token.

Результаты

Результатом работы стал ввод в эксплуатацию независимого аналитического контура, интерфейсная часть которого интегрирована в рабочее пространство администратора конференций. Рендеринг визуальных компонентов базируется на аппаратном ускорении HTML5 Canvas, что снижает вероятность замедления при перерисовке сложных полигональных структур (рис. 1).

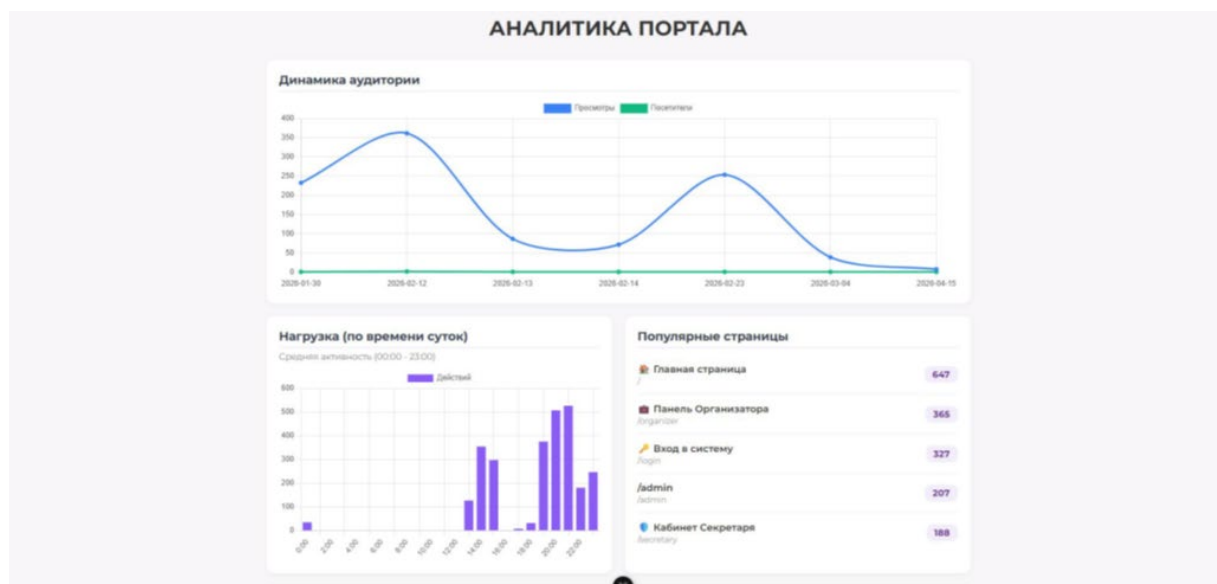


Рис. 1. Панель мониторинга пользовательской активности

Система динамически транслирует унифицированные идентификаторы ресурсов в паттерны, которые облегчают интерпретацию данных. Изменение временных фильтров на клиенте инициализирует пересчет SQL-матриц на сервере, возвращая легковесные JSON-ответы, содержащие итоговые массивы координат для осей графиков.

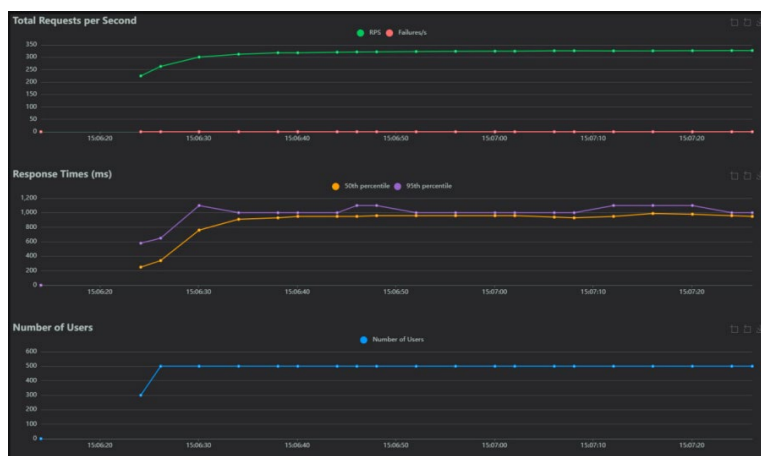
С целью верификации надежности спроектированного решения был инициирован цикл автоматизированного тестирования. Специфика проверок охватывала устойчивость лексического парсера к искаженным или злонамеренно модифицированным метаданным, а также валидацию результатов математической группировки. Результаты инструментального контроля приведены в табл. 1.

Таблица 1. Отчет инструментального анализа покрытия алгоритмов тестами

Компонент микросервиса	Объем кода	Пропуски	Уровень покрытия, %
Слой подключения к БД	22	0	100
Модуль бизнес-логики	69	1	99
Модели реляционных таблиц	17	0	100
Контроллер сценариев	22	0	100

Анализ табл. 1 показывает, что индекс покрытия исполняемой бизнес-логики достигает 99 %. Единственный зафиксированный пропуск относится к системному вызову процесса, инициирующего прослушивание портов операционной системы, что не подлежит проверке в рамках изолированных модульных тестов.

Помимо модульных проверок, для подтверждения заявленных архитектурных преимуществ асинхронного подхода (ASGI) было проведено нагрузочное стресс-тестирование модуля сбора телеметрии с использованием инструментария Locust. Эмулировался сценарий пиковой активности в момент массовой загрузки докладов, предполагающий наличие 500 конкурентных соединений, ежедневно генерирующих POST-запросы с метаданными. Метрики производительности представлены на рис. 2.

**Рис. 2.** Результаты стресс-тестирования

Анализ графиков (рис. 2) показывает высокую пропускную способность сервера, который без проблем держит нагрузку порядка 320-330 запросов в секунду. При этом важный показатель отказов остается на нуле, что показывает отсутствие потерь пакетов данных. Медианное время отклика при максимальной нагрузке стабилизируется в пределах секунды. Однако для пользователя эта задержка незаметна: поскольку браузер отправляет телеметрию в фоне, задержки в работе основной части портала отсутствуют.

Заключение

Итоги разработки подтверждают, что созданная система полностью удовлетворяет требованиям и справляется со всеми задачами. Внедрение продукта на

сервера учебного заведения закрывает вопрос цифрового суверенитета, так как вуз получает полный контроль над метриками. Перенос тяжелой комбинаторики и агрегационных процессов на сторону ядра PostgreSQL доказал свою эффективность, обеспечив быстрое формирование тепловых карт активности даже при экспоненциальном росте базы логов.

Исключение файлов cookie из архитектуры системы позволило устранить правовые риски, связанные с соблюдением строгих нормативных требований в отношении обработки персональных данных. Такой подход снижает нормативную нагрузку и соответствует принципам минимизации сбора персональных данных, закрепленным в действующем законодательстве (в том числе в Федеральном законе № 152). Созданный программный комплекс обладает запасом прочности и не подвержен деградации производительности в критические периоды завершения регистрационных кампаний. Перспективы развития заложенной архитектуры открывают пути для внедрения предиктивных математических моделей. Анализ собранных исторических массивов позволит в будущем проактивно прогнозировать моменты пиковых нагрузок и автоматизировать выделение аппаратных ресурсов серверов.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алексеев С. Н. Методы защиты пользовательских данных в веб-аналитике // Информационная безопасность регионов. – 2022. – № 1. – С. 55–61.
2. Макаров С. В. Анализ влияния эвристических блокировщиков рекламы на достоверность веб-статистики // Вестник компьютерных и информационных технологий. – 2022. – № 2. – С. 18–25.
3. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга. – Санкт-Петербург : Питер, 2019. – 544 с.
4. Баранов И. В. Применение микросервисной архитектуры в распределённых информационных системах // Современные информационные технологии и ИТ-образование. – 2021. – Т. 17, № 3. – С. 601–610.
5. Костин Д. Ю. Использование REST-архитектуры при разработке веб-приложений // Молодой ученый. – 2021. – № 15. – С. 12–18.
6. Маржаков Д. А. Разработка высоконагруженных веб-приложений на Python // Информационные технологии. – 2021. – № 7. – С. 25–31.
7. Жуков Е. А. Асинхронное программирование в Python для веб-сервисов // Программные продукты и системы. – 2021. – № 4. – С. 715–722.
8. Гусев П. А. Анализ производительности асинхронных API-серверов // Вестник современных исследований. – 2023. – № 6. – С. 73–79.
9. Куракин А. С. Разработка системы сбора цифровых отпечатков браузера (Browser Fingerprinting) для идентификации пользователей // Программная инженерия. – 2020. – Т. 11, № 8. – С. 488–496.
10. Вьюгин В. В. Методы обработки и анализа больших данных в веб-системах // Научно-технический вестник информационных технологий, механики и оптики. – 2020. – Т. 20, № 6. – С. 912–919.
11. Гаврилов А. В. Методы оптимизации SQL-запросов в PostgreSQL // Системный администратор. – 2022. – № 5. – С. 44–49.
12. Моргунов Е. PostgreSQL. Основы оптимизации SQL-запросов. – Москва : ДМК Пресс, 2021. – 256 с.

© Е. Н. Яковлев, А. А. Шаранов, 2026