– проводить точный расчёт заработной платы с учётом всех необходимых параметров;

– автоматизировать процесс расчёта налогов и других отчислений;

– сохранять и анализировать данные о расчётах.

Программа представляет собой комплексное решение, способное значительно упростить и ускорить процессы управления персоналом и расчёта заработной платы на предприятии. Дальнейшее развитие программного продукта может включать расширение функциональности и адаптацию под специфические требования различных отраслей промышленности.

Список литературы

1. Яковлев А. В. Управление производством: планирование и диспетчеризация. М.: ООО «1С-Паблишинг», 2018. 219 с. ISBN 978-5-9677-2707-8

2. Разработка информационного обеспечения расчета плановых калькуляций выпускаемой продукции (на примере АО «ККЖБМИ») / Н. Ф. Телешева, И. В. Филимоненко, Е. И. Высотенко, Д. И. Ярещенко. Красноярск, 2018. 110 с.

3. Курдин А. Р., Байков М. Ю., Чеблаков Г. С., Пахомова И. В. Влияние глобальных вызовов на российский рынок нефти и нефтепродуктов // Научный журнал НИУ ИТМО. Серия: Экономика и экологический менеджмент. 2020. №1. С. 158-169.

4. Парушина Н. В., Лытиева Н. А., Ершова И. Г. Региональное управление экономикой: монография. Воронеж: Научная книга, 2010. 210 с.

5. Лытиева Н. А. Управление системными изменениями // Вестник ОрелГИЭТ. 2008. №4. С. 72-83.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СИНХРОННЫХ И АСИНХРОННЫХ ВЕБ-СЕРВЕРОВ

Журавлёв Д. В., Букреев Д. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: dmitriy.bukreev@mel-su.ru

Научный руководитель: Букреев Д. А.

Введение

В современное время развитие веб-технологий сопровождается ростом требований к производительности серверной части, устойчивости к высокому трафику и способности обрабатывать большое количество одновременных параллельных соединений. Традиционные веб-серверы используют синхронную модель запросов, при которой каждый запрос выполняется последовательно или в отдельных потоках. Такая модель является предсказуемой, при высоком числе одновременных соединений возникает падение производительности.

Современные же веб-приложения выставляют новые требования – поддержку WebSocket-соединений, асинхронных операций и обработку большого числа запросов. Эти условия способствуют развитию событийно-ориентированным, неблокируемым серверам, которые работаю на базе, асинхронных моделей ввода-вывода.

В экосистеме Python синхронная модель реализуется WSG, среди которых наиболее распространенной является сервер Gunicorn, который демонстрирует высокую стабильность и надежность при использовании многопроцессной модели обработки запросов, что делает его стандартным решением для классических веб-приложений.

Альтернативой синхронной архитектуре стал асинхронный стек ASGI, который обеспечивает неблокирующую обработку соединений. Один из наиболее изученных представителей является Uvicorn, показывая, что ASGI-серверы демонстрируют существенный прорыв в скорости обработки большого числа параллельных запросов благодаря событийной модели и отсутствию блокировок. Дополнительным элементом серверной архитектуры является Nginx – высокопроизводительный обратный прокси-сервер. Благодаря его модели работы Nginx способен эффективно обслуживать десятки тысяч одновременных соединений с минимальными задержками. Таким образом задача эпохи высоконагруженных веб-систем заключается в сравнительном анализе синхронных и асинхронных моделей обработки запросов, оценке их производительности и архитектуры, выбор оптимального решения для конкретного класса приложений.

Цель исследования – провести систематический сравнительный анализ синхронных и асинхронных веб-серверов, выявить их архитектурные особенности, показатели производительности и области применения, при проектировании современных веб-систем.

Материал и методы исследования

Исследование заключается в сравнительном анализе современных синхронных и асинхронных веб-серверов, применяемых в современной разработке веб-приложений.

В качестве материалов исследования рассматривались, три серверных решения, применяющиеся в современной веб-разработке: синхронный сервер Gunicorn, асинхронный сервер Uvicorn и обратный прокси-сервер Nginx. Для анализа были использованы данные о их времени отклика, возможной пропускной способности, их устойчивости к нагрузке и так же особенности архитектуры и модели обработки запросов.

Методы исследования включают в себя аналитическое изучение принципов работы синхронной и асинхронной обработки запросов, а также сравнение серверов по ключевым показателям производительности.

Дополнительно было произведено тестирование под нагрузкой, которое позволило оценить работу серверов в условиях нагрузки реального трафика.

Результаты исследования и их обсуждение

Проведенное исследование было направлено на сравнительный анализ поведения синхронных и асинхронных веб-серверов в условиях различной нагрузки, а также на оценку влияния прокси-уровня Nginx на общую производительность системы. Результаты были получены с опорой на данные экспериментальных и обзорных работ, которые посвящены архитектуре высоконагруженных приложений, особенностям асинхронной обработки и сравнительному анализу веб-серверов [1-3].

1. Поведение синхронных веб-серверов при увеличении нагрузки

В ходе проведенного исследования было установлено, что синхронные веб-серверы, использующие модель блокирующей обработки запросов, характерную для классического WSGI-подхода, демонстрируют стабильные и предсказуемые показатели производительности при сравнительно небольшом количестве одновременных клиентских соединений. В условиях низкой и умеренной нагрузки такие серверы способны обеспечивать приемлемое среднее время отклика, при этом пропускная способность системы возрастает практически линейно по мере увеличения числа активных пользователей вплоть до определенного порогового значения. Однако при дальнейшем росте уровня параллелизма начинают проявляться характерные ограничения блокирующей архитектуры. Среднее время отклика заметно увеличивается, что негативно сказывается на общей производительности системы, а рост пропускной способности замедляется и со временем выходит на плато. Это связано с тем, что при достижении максимального количества доступных рабочих потоков сервер перестает эффективно обрабатывать новые запросы, вынужденно переводя их в очередь ожидания.

Наиболее выраженные недостатки синхронной модели обработки запросов проявляются в сценариях, где наблюдается существенное увеличение времени удержания соединений. Подобные ситуации характерны для веб-приложений, выполняющих длительные операции ввода-вывода, включая обращения к базам данных, взаимодействие с удаленными внешними сервисами, работу с файловыми системами и сетевыми ресурсами. В рамках блокирующей модели каждый такой запрос занимает отдельный рабочий поток на весь период выполнения операции, включая время ожидания завершения ввода-вывода.

В результате сервер оказывается вынужден последовательно обрабатывать запросы в рамках ограниченного пула потоков, что приводит к формированию очередей и росту времени ожидания для клиентов. Данное обстоятельство напрямую отражается на качестве пользовательского опыта, выражаясь в увеличении задержек, снижении отзывчивости системы и потенциальных тайм-аутах соединений [2].

Таким образом, можно сделать вывод, что синхронные серверные решения, основанные на традиционной архитектурной модели «один поток – один запрос», обладают ограниченными возможностями масштабирования в условиях высокой конкурентности. При увеличении числа одновременных операций такая архитектура приводит к нерациональному использованию вычислительных ресурсов, перерасходу памяти и снижению общей эффективности задействования аппаратных возможностей сервера, что делает ее менее подходящей для современных высоконагруженных веб-приложений.

2. Эффективность асинхронных серверов и событийно-ориентированных архитектур

В процессе тестирования асинхронных серверных конфигураций были получены результаты, демонстрирующие принципиально иное поведение системы при увеличении числа одновременных клиентских соединений по сравнению с синхронными решениями. Асинхронные серверы, использующие событийно-ориентированную архитектуру и неблокирующие операции ввода-вывода, показали способность сохранять низкий уровень задержек на протяжении значительно более широкого диапазона нагрузок. По мере роста числа подключенных клиентов среднее время отклика увеличивалось более плавно и равномерно, не демонстрируя резких скачков, характерных для блокирующих моделей обработки запросов. Пропускная способность асинхронного сервера при этом продолжала возрастать вплоть до уровней нагрузки, при которых синхронные серверы уже испытывали выраженную деградацию производительности, проявляющуюся в росте очередей запросов и увеличении времени ожидания отклика [5]. Данный эффект объясняется тем, что асинхронная модель не требует закрепления отдельного потока выполнения за каждым запросом на весь период его обработки, что позволяет более эффективно использовать вычислительные ресурсы сервера.

На основании полученных результатов, а также анализа особенностей асинхронной обработки запросов в веб-приложениях, можно сделать вывод о том, что переход к неблокирующей архитектуре позволяет существенно сократить простои, связанные с ожиданием завершения операций ввода-вывода. Взаимодействие с внешними ресурсами, такими как базы данных, сетевые сервисы и файловые системы, перестает приводить к блокировке рабочих потоков, что способствует повышению общей утилизации серверных ресурсов без необходимости пропорционального увеличения вычислительной мощности или объема оперативной памяти. Особенно отчетливо преимущества асинхронных серверов проявляются в сценариях с дли-

тельно удерживаемыми соединениями, например при использовании протоколов WebSocket, реализации стриминговых механизмов передачи данных или работе с серверными событиями. В подобных условиях сервер способен приостанавливать обработку конкретного соединения и переключаться на выполнение других задач, возвращаясь к нему только при возникновении соответствующего события, вместо блокировки потока на все время существования соединения. Данный подход обеспечивает более равномерное распределение нагрузки между доступными ресурсами и позволяет обслуживать значительно большее количество клиентов при фиксированной аппаратной конфигурации сервера. Именно поэтому событийно-ориентированная и асинхронная модели обработки запросов лежат в основе большинства современных архитектур высоконагруженных и масштабируемых веб-приложений, ориентированных на работу в условиях высокой конкурентности и интенсивного взаимодействия с внешними ресурсами.

3. Влияние использования Nginx как прокси-уровня

Отдельное внимание в рамках исследования было уделено анализу влияния обратного прокси-сервера Nginx, используемого в качестве внешнего прокси уровня, размещенного перед серверами приложений, на общую производительность и устойчивость системы. Полученные экспериментальные данные подтверждают, что включение Nginx в архитектуру веб-приложения позволяет существенно повысить стабильность функционирования системы и увеличить ее совокупную пропускную способность. Использование Nginx способствует эффективной разгрузке серверов приложений за счет переноса на прокси-уровень обработки статического контента, выполнения терминации TLS-соединений, а также первичной маршрутизации и балансировки входящих HTTP-запросов. Благодаря событийно-ориентированной архитектуре и неблокирующей модели обработки соединений, Nginx способен обслуживать десятки тысяч одновременных подключений при относительно низком потреблении вычислительных ресурсов, что является критически важным фактором для высоконагруженных систем, особенно в условиях кратковременных или непредсказуемых всплесков трафика [4]. В случае использования синхронных серверов приложений внедрение Nginx оказывает выраженный положительный эффект, позволяя сглаживать пиковые нагрузки и сокращать количество прямых соединений, поступающих непосредственно на сервер приложений. За счет этого снижается вероятность исчерпания пула рабочих потоков, уменьшается глубина очередей запросов и повышается устойчивость системы к резким изменениям интенсивности нагрузки. В результате наблюдается снижение среднего времени отклика и более плавный, контролируемый рост задержек по мере увеличения числа клиентов.

В конфигурациях, основанных на асинхронных серверах приложений, применение Nginx дополнительно усиливает уже присущие им преимущества событийной архитектуры. Комбинация неблокирующего прокси-уровня и асинхронного сервера приложений формирует согласованную многоуровневую архитектуру, в которой каждый компонент эффективно обрабатывает свою часть нагрузки без взаимной блокировки ресурсов. Такая интеграция обеспечивает максимальные показатели пропускной способности и высокую устойчивость к нагрузкам, сохраняя при этом низкий и предсказуемый уровень задержек даже при значительном увеличении числа одновременных соединений. Таким образом, результаты исследования показывают, что использование Nginx в качестве обратного прокси-сервера является важным архитектурным элементом при проектировании современных высокопроизводительных и масштабируемых веб-приложений. Особенно эффективной данная схема оказывается при сочетании Nginx с асинхронными серверами приложений, где достигается оптимальный баланс между производительностью, устойчивостью и рациональным использованием аппаратных ресурсов.

Заключение

В ходе анализа полученных результатов исследования, подтверждается, что синхронные веб-сервера имеют свою практическую значимость в задачах с относительно простой моделью загрузки: небольшое число одновременных пользователей, отсутствие долговременных соединений. В таких условиях блокирующая модель не доходит до пика своих ограничений и может работать быстро и стабильно. Преимущества такой модели проявляются в простоте конфигурации и предсказуемости поведения. Было выявлено, что асинхронные веб-сервера и событийно-ориентированные архитектуры демонстрируют более устойчивое состояние в высоконагруженных системах. При большем количестве одновременных соединений, так же при наличии долгоживущих запросов и при значительной части времени, которая затрачивается на взаимодействие с внешними ресурсами, поэтому переход с блокирующей модели на неблокирующую модель обеспечивает более высокую производительность, пропускную способность и меньшие задержки. Подтверждается, что внедрение Nginx в качестве прокси-уровня, является хорошим и эффективным решением, для повышения производительности и надежности всей системы, не зависимо от типа сервера, который будет использоваться на уровне приложений. Но стоит учесть, что наиболее заметный эффект наблюдается при сочетании Nginx с асинхронными серверами, где общая со-

бытийная модель прослеживается на всем пути обработки запросов.

Список литературы

1. Амиров С. Н. Особенности разработки высоконагруженных систем // International Journal of Open Information Technologies. 2020. №8. URL: https://cyberleninka.ru/article/n/osobennosti-razrabotki-vysokonagruzhennyh-sistem (дата обращения: 11.12.2025).

2. Букреев Д. А., Луц С. М. Особенности разработки web-платформ автоматизированного взаимодействия с клиентом // Современные проблемы геометрического моделирования и информационные технологии: материалы II Межрегиональной научно-практической конференции преподавателей и студентов, посвященной 60-летию образования Мелитопольской школы прикладной геометрии (Мелитополь, 28 мая 2024 года). Мелитополь: Мелитопольский государственный университет, 2024. С. 76-85. EDN: BVUZYR.

3. Журавлев Д. В., Букреев Д. А. Современное состояние реактивных технологий frontend разработки // Современные проблемы геометрического моделирования и информационные технологии: материалы II Межрегиональной научно-практической конференции преподавателей и студентов, посвященной 60-летию образования Мелитопольской школы прикладной геометрии (Мелитополь, 28 мая 2024 года). Мелитополь: Мелитопольский государственный университет, 2024. С. 152-157. EDN: EGZAEX.

4. Латыпов Э. Ф. Сравнительный анализ работы веб-серверов apache и nginx // Экономика и социум. 2017. №6-2 (37). URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-raboty-veb-serverov-apache-i-nginx (дата обращения: 11.12.2025).

5. Хомякова А. А. Возможности программного обеспечения «веб-серверов» Apache и MS IIS по противодействию деструктивным информационным кибернетическим воздействиям // Новые информационные технологии в автоматизированных системах. 2019. № 22.

АНАЛИЗ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ РАЗРАБОТКИ И ПРОЕКТИРОВАНИЯ ВЕБ-РЕСУРСА

Задорожный А. Я., Букреев Д. А.

ФГБОУ ВО «Мелитопольский государственный университет», Мелитополь, e-mail: dmitriy.bukreev@mel-su.ru

Научный руководитель: Букреев Д. А.

Введение

Стремительное развитие цифровых технологий привело к существенному усложнению задач, связанных с проектированием и разработкой веб-ресурсов. Если ранее веб-сайт выполнял преимущественно информационную функцию и строился на основе минимального набора технологий, то сегодня он представляет собой полноценную программную систему, обеспечивающую интерактивное взаимодействие, обработку данных, интеграцию со сторонними сервисами и адаптацию под различные пользовательские устройства. Рост пользовательских ожиданий в отношении удобства, скорости загрузки и безопасности существенно влияет на выбор инструментов и технологий, а сама веб-разработка становится междисциплинарным направлением, объединяющим программирование, проектирование интерфейсов, архитектуру приложений и инфраструктурные решения.

Современный веб-ресурс должен одновременно удовлетворять целому ряду требований: обеспечивать понятную навигацию и визуальную целостность интерфейса, корректно отображаться на различных устройствах, быстро реагировать на действия пользователя и гарантировать надёжную защиту данных. Достижение такого сочетания характеристик возможно только при грамотном выборе технологического стека и архитектурных подходов. Анализ современных технологий разработки позволяет выявить ключевые тенденции отрасли, связанные с переходом к одностраничным приложениям, широким распространением реактивных фреймворков, использованием гибких архитектурных подходов и развитием инструментов автоматизации. Значимую роль играет и этап проектирования веб-ресурса, включающий UX/UI-аналитику, создание прототипов, оценку пользовательских сценариев и выбор архитектурного решения. Именно сочетание технологической и проектной составляющих определяет качество конечного продукта.

Цель исследования – анализ современных технологий разработки и проектирования веб-ресурсов, выявление их особенностей, преимуществ и ограничений, а также формирование рекомендаций по выбору технологического стека в зависимости от задач веб-проекта.

Материалы и методы исследования

Материалом исследования послужил комплекс современных технологий, применяемых при проектировании и разработке веб-ресурсов, а также нормативные и методические документы, описывающие требования к веб-приложениям с точки зрения функциональности, удобства использования, производительности и безопасности. В обзор были включены как классические технологии, лежащие в основе веба (HTML5, CSS3, стандарты ECMAScript), так и современные фреймворки фронтенд- и бэкенд-разработки, архитектурные подходы к построению веб-приложений, инструменты автоматизации и средства обеспечения качества. Дополнительно учитывался практический опыт реализации веб-ресурсов образовательного и информационного профиля, что позволило сопоставить теоретические положения с реальными сценариями разработки.

Методологическая основа исследования опирается на сочетание теоретического анализа и сравнительного подхода. Дополнительно использовались элементы системного подхода и структурного анализа: технологии рассматривались не изолированно, а как части единого технологического стека, включающего этапы проектирования, реализации, тестирования и сопровождения веб-ресурса. Это позволило оценить совместимость инструментов между собой, а также определить типичные конфигу-