DOI 10.37882/2223-2966.2024.3-2.06

РАЗРАБОТКА АДАПТИВНОГО ПО ДЛЯ ЭФФЕКТИВНОГО ОБНАРУЖЕНИЯ АНОМАЛИЙ ВЕБ-СЕРВЕРА НА ОСНОВЕ ЭРГОНОМИЧЕСКОГО АНАЛИЗА

DEVELOPMENT OF ADAPTIVE SOFTWARE FOR EFFECTIVE DETECTION OF WEB SERVER ANOMALIES BASED ON ERGONOMIC ANALYSIS

B. Goryachkin
M. Torzhkov

Summary. *Problem statement*. In manual monitoring systems, the human factor plays a significant role in the stage of problem detection and decision-making, increasing the risks of overlooking critical changes in the system due to human errors or lack of attention.
Goal. It is necessary to develop an alert system for monitoring the web server, optimally adapted to identify the most common problems.
Result. The authors conducted an analysis, set limit values for alerts and experimentally substantiated their effectiveness. Particular attention is paid to assessing the effectiveness of the system in comparison with human monitoring. In addition, a recommended order for alerts has been established, which allows you to most effectively identify problems in accordance with established principles.
Practical significance. These results can be used to optimize monitoring systems and timely respond to problems in the operation of the web server.

Keywords: monitoring, alert, metrics, ergonomics, CPU, RAM, optimization.

Горячкин Борис Сергеевич
кандидат технических наук, доцент;
Московский государственный технический университет им. Н.Э. Баумана
bsgor@mail.ru
Торжков Максим Сергеевич
Московский государственный технический университет им. Н.Э. Баумана
makstorzhkov@yandex.ru

Аннотация. *Постановка проблемы*. В ручных системах мониторинга высокую роль играет человеческий фактор на этапе обнаружения проблем и принятия решения, что повышает риски пропуска критических изменений в системе из-за человеческих ошибок или невнимательности.
Цель. Необходимо разработать систему оповещений для мониторинга веб-сервера, оптимально адаптированной для выявления самых распространенных проблем.
Результаты. Авторы провели анализ, установили граничные значения для алертов и экспериментально обосновали их эффективность. Особое внимание уделено оценке эффективности системы по сравнению с человеческим мониторингом. Помимо этого, установлен рекомендованный порядок прихода алертов, который позволяет наиболее эффективно выявлять проблемы в соответствии с установленными принципами.
Практическая значимость. Эти результаты могут быть использованы для оптимизации систем мониторинга и своевременного реагирования на проблемы в работе веб-сервера.

Ключевые слова: мониторинг, алерт, метрики, эргономика, CPU, оперативная память, оптимизация.

Введение

Современные системы мониторинга веб-серверов состоят из различных компонентов, обеспечивающих комплексный анализ и контроль за работой веб-сервера, приложений, и связанных с ними ресурсов, таких как инструменты отслеживания ошибок, логирование, визуализация и отчетность.

Но в таких системах высокую роль играет человеческий фактор на этапе обнаружения проблем и принятия решения. Это повышает риски пропуска критических изменений в системе из-за человеческих ошибок или невнимательности.

Одним из возможных решений являются интеграции с системами оповещения. Системы оповещения используются во многих продвинутых системах мониторинга веб-серверов. Ее задача — предупреждать администраторов или ответственных лиц о возможных проблемах, аномалиях или событиях, требующих внимания, исключая нужду в ручном анализе метрик и графиков внутри системы мониторинга. В рамках данной статья описана разработка ПО для системы оповещений на основе эргономического анализа. Анализ будет заключатся в оценке того, насколько эффективнее данное ПО будет работать в отличие от человека.

Обоснование актуальности проблемы

Система оповещения может использоваться для мониторинга различных параметров, таких как загрузка ЦП, использование памяти, доступность сети, логические ошибки, а также для отслеживания событий безопасности. В целом, она является неотъемлемой частью ИТ-инфраструктуры, способствуя более эффективной и стабильной работе системы.

Но все эти параметры требует определенного понимания того, какими они должны быть для оптимальной работы системы. Далее мы проанализируем каждый параметр и определим его допустимые значения.

А. Загрузка CPU

ЦП выполняет всю обработку, вычисления и логику. Для надежной и эффективной работы серверу требуется один или несколько быстрых и мощных процессоров, способных обрабатывать и выполнять несколько задач одновременно. Потребление ЦП в высоконагруженных серверах может достигать 75 % в пике [2]. При этом загрузка процессора должна составлять максимум 80 процентов. В противном случае сервер начинает работать медленнее, поскольку ставит в очередь большинство задач, поэтому ответ на запросы занимает больше времени.

Загрузка ЦП, которая постоянно превышает 80 %, должна быть исследована и исправлена. Производительность сервера часто снижается, когда загрузка ЦП достигает ~80–90 %, и становится более выраженной, когда загрузка приближается к 100 %. Загрузка ЦП при обслуживании одного запроса незначительна, но при масштабировании, возникающем во время максимального трафика, пиковая нагрузка может привести к перегрузке сервера [3].

Б. Потребление оперативной памяти

Оперативная память (RAM) обеспечивает временное хранилище данных, обрабатываемых ЦП, и обеспечивает лучшую и быструю производительность чтения/записи, чем жесткий диск. Достаточный объем памяти снижает необходимость частого доступа сервера к медленной памяти жесткого диска.

Бывают случаи, когда пользователи сообщают о странном поведении системы всякий раз, когда использование памяти достигает 100 %. Другими словами, всякий раз, когда объем физической памяти исчерпывается запущенными процессами, хост начинает демонстрировать очень высокий уровень использования % sy, что влияет на производительность реальных выполняемых задач, а также на интерактивное реагирование [4]. Учитывая, что сервер, отвечающий на веб-запросы, с большей вероятностью будет потреблять ОЗУ [5], чем другие ресурсы, желательно обеспечить соответствующую емкость ОЗУ для оптимальной производительности сервера в перегруженных ситуациях.

Необходимо оставлять около 10 процентов памяти, чтобы не исчерпать весь объем во время обычных операций. При отсутствии свободного пространства в оперативной памяти система может начать использовать механизм подкачки, что может привести к потере данных в случае аварийного завершения работы системы [6].

В. Потребление пространства на диске

Общий объем дискового пространства должен представлять собой сумму всех различных данных, которые вы храните на диске. Лучше всего следить за тем, чтобы использование диска никогда не превышало 85 % доступного пространства. Это означает, что на жестком диске всегда должно оставаться минимум 15 процентов свободного места [8], к примеру, для создания резервных копий баз данных и конфигураций сервера.

Самые распространенные проблемы, с которыми можно столкнуться при переполнении диска [7]:

— Обновления не могут быть применены на сервере.
— Веб-сайты не обслуживаются/не загружаются.
— Не происходит резервное копирование данных и конфигураций сервера.
— Базы данных могут быть повреждены и потребовать восстановления из резервной копии.

Г. Время ответа сервиса на запросы

Данный алерт предупреждает о том, что время, необходимое для того, чтобы веб-сервер обработал запрос и вернул ответ, превысило заданный лимит. Это важная метрика для отслеживания производительности и качества обслуживания вашего веб-сервера. Когда такой алерт срабатывает, это может указывать на проблемы, которые могут привести к замедлению или недоступности веб-сервиса для конечных пользователей.

Рекомендации по времени отклика для веб-приложений такие же, как и для всех других приложений. Эти рекомендации остаются неизменными уже 46 лет, поэтому они вряд ли изменятся в зависимости от того, какая технология внедрения будет следующей [9]:

1 секунда: предельное значение для пользователей, чтобы ощущать, что они могут свободно перемещаться по странице, не дожидаясь сервера. Задержка в 0,2–1,0 секунды означает, что пользователи замечают задержку и, таким образом, чувствуют, что сервер «работает» над запросом, а не что команда является прямым следствием действий пользователей. Пример: если сортировку таблицы по выбранному столбцу невозможно выполнить за 0,1 секунды, ее обязательно нужно сделать за 1 секунду, иначе пользователи почувствуют, что пользовательский интерфейс работает вяло и потеряет ощущение «потока» при выполнении. их задача. При задержках более 1 секунды следует указать пользователю, что компьютер работает над проблемой, например, изменив форму курсора.

10 секунд: ограничение, позволяющее пользователям удерживать внимание на задаче. Для всего, что медленнее 10 секунд, необходим индикатор процента выполнения, а также четко обозначенный способ, позволяющий пользователю прервать операцию. Предположим, что пользователям потребуется переориентироваться, когда они вернутся к пользовательскому интерфейсу после задержки более 10 секунд. Задержки длительностью более 10 секунд допустимы только во время естественных перерывов в работе пользователя, например при переключении задач.

В тоже время, используя доступные исследования, Бейли [10] рекомендовал, чтобы компьютер отвечал на запросы пользователей в течение 2 секунд. Мартин и Корл [11] утверждали, что для большинства задач по вводу данных время ответа быстрее 1 секунды не дает преимуществ, и обнаружили линейное снижение производительности при более медленном времени ответа (от 1 до 5 секунд). В задачах решения проблем, которые больше похожи на задачи веб-взаимодействия, они не обнаружили достоверного влияния на производительность вплоть до 5-секундной задержки.

В следствие чего можно утверждать, что время ответа до 5 секунд будет удовлетворительным для пользователя, а время ответа более 10 секунд может сильно влиять на пользовательский опыт взаимодействия с веб-сервисом.

Д. Процент ошибок сервиса

Это важная метрика для мониторинга стабильности и доступности вашего веб-сервиса и качества обработки запросов.

Хотя уровень ошибок HTTP-сервера не имеет четкой корреляции со скоростью приложения, он является важнейшим показателем производительности. Он показывает общее количество внутренних сбоев сервера (также известных как номера HTTP 5xx), которые были переданы потребителям. Если исключение или другая ошибка не обрабатывается должным образом, неисправные приложения будут отправлять эти ошибки [12]. Настройка оповещения о возникновении подобных ошибок — разумный подход.

Процент ошибок сервиса и его доступность тесно связаны, и высокий процент ошибок обычно негативно влияет на доступность сервиса. Если сервис часто генерирует ошибки, это может привести к недоступности сервиса для конечных пользователей. Высокий процент ошибок может сопровождаться сбоями в работе приложения, что в конечном итоге может привести к отказу в обслуживании (Downtime) [13].

Доступность (Availability) и процент ошибок (Error Rate) взаимосвязаны по следующей формуле:

$$Availability = 100\% - ErrorRate \quad (1)$$

где:

Availability — процент времени, в течение которого сервис доступен и работает без ошибок. Обычно измеряется в процентах. Если сервис всегда доступен и не выдает ошибок, его доступность равна 100 %.

Error Rate — это процент времени, в течение которого сервис возвращает ошибку. Опять же, измеряется в процентах. Если сервис всегда возвращает ошибку, его процент ошибок равен 100 %.

Предположим, мы стремимся к значению доступности сервера = 99 %. Тогда значение неудачных запросов должно равняться <1 %.

Е. Объем трафика

Алерт предупреждает о том, что объем входящего трафика значительно увеличился. Это может быть важным сигналом, поскольку резкое увеличение трафика может привести к различным проблемам, таким как перегрузка сервера, замедление производительности или даже отказ в обслуживании. К тому же при переполнении полосы пропускания пользователи не смогут получить доступ к вашей веб-странице, что может сильно повлиять на их лояльность и пользовательский опыт [14].

Трафик, с которым работает веб-сервер, может принимать различные значения. Все зависит от того, с какими задачами сталкивается ваш сервис. К примеру, если ваш сервис занимается хранением картнок, то соответственно данные, которые он получает по сети от одного пользователя, должны равняться размеру картинки (от 4 Кб до 10 Мб). Но если ваш веб сервис работает только со статичными страницами, следовательно объем трафика не должен превышать размера страницы (от 500 Байт до 25 Кб). Помимо этого, также нужно учитывать количество серверов, которое занимается приемом входящих соединений. Это в разы может изменить объемы входящих данных, так как зачастую нагрузка распределяется равномерно между всеми серверами в топологии [15].

Предположим, у нас средний размером страницы 25 КБ и ожидаемой посещаемостью 100 человек в минуту. Если каждый из посетителей посещает в среднем 2 страниц в минуту, нам потребуется минимум 100 x 25 КБ x 2 = 5 МБ полосы пропускания на 1 сервер.

Проектирование

Для проведения исследования мы спроектировали систему с интегрированным модулем оповещений. Ее обобщенная архитектура представлена на рис. 1.

Система оповещений представляет из себя встраиваемый в систему мониторинга код, который отслеживает состояние заданных метрик или построенных на этих метриках логических выражений, и, согласно поставленным правилам, отправляет сообщения пользователю в заданном формате в мессенджер. Такие сообщения называются алертами. Для этой задачи будем использовать веб-сервис «Форум запросов и ответов». В нем реализована система мониторинга с возможностью отправки оповещений в канал в Telegram.

При разработке системы мы полагались на следующие принципы:

1. Так как мы гарантируем обнаружение проблемы, то при ее возникновении хотя бы один алерт должен прийти;
2. Так как наша система является эффективной, мы гарантируем, что алерт должен прийти до выхода из строя сервера и при этом время между возникновением проблемы и приходом алерта должно быть минимальным.
3. Количество алертов должно быть минимальным, но при этом их набор должен в полной мере покрывать выбранные проблемы.

В рамках проектирования системы оповещения нам нужно выполнить несколько задач:

— обозначение набора основных проблем, на обнаружение которых будет настроена наша система оповещений;
— определение набора критических метрик, на основе которых будет происходит отслеживание состояние системы. На основе эксперимента из практической части этот набор может изменится на основании принципа 3, описанного выше.
— построение алертов, для которых необходимо задать пороговые значения для конкретных метрик.

А. Проблемы веб-сервисов

В рамках данной работы мы рассмотрели пять основных проблем, с которыми может столкнуться любой веб-сервис [1]:

— Медленное время отклика
— Поломка в DNS-запросах
— Взаимные блокировки базы данных
— Плохой код (утечки памяти)
— Неожиданно высокий трафик

Б. Метрики веб-сервисов

В данной работе будут рассмотрены следующие метрики:

1. Среднее время ответа на запрос
2. Количество HTTP запросов
3. Общее время использования процессора
4. Количество используемой оперативной памяти
5. Количество используемого дискового пространства
6. Количество полученных/переданных данных по сети

Исследование

Определение порядка прихода алертов в системе мониторинга веб-сервера зависит от приоритетов и критичности событий. Обычно, более критичные события должны вызывать оповещения раньше, чтобы операторы системы могли быстро реагировать на проблемы.

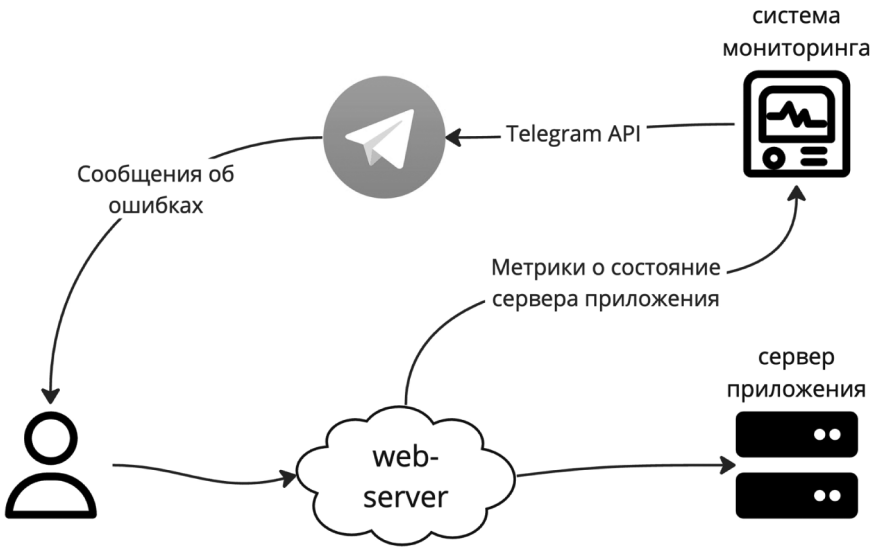

Рис. 1. Графическое представление системы оповещений

Каждый эксперимент будет происходить в два этапа.

— Этап 1. Доведение системы до нерабочего состояния и оценка времени вывода из строя системы. Данный этап необходим для приоритезации проблем по уровням критичности, а также для оценки максимального времени реакции Δ_{max} для каждой проблемы, которое допустимо для нашей системы.
— Этап 2. Определение критичных алертов на основе времени их прихода.

Первоначально для настройки порядка прихода алертов необходимо определить, какие алерты помогут нам отследить конкретные проблемы.

В данных экспериментах мы будем отслеживать время появления инцидентов (время запуска скриптов, отключения сервера) и время поступления сообщения в чат.

Помимо подбора порядка прихода алертов мы сможем протестировать корректность выбора граничных значений для алертов.

Результаты всех экспериментов представлен в табл. 1 и 2.

Анализ результатов

На основе этапа 1 распределим вышеперечисленные проблемы по трем уровням критичности.

- Поломка в DNS-запросах.

В рамках данной проблемы система продолжила функционировать. Но при этом доступ к веб-сервису был ограничен из-за некорректной работы DSN. Это значит, что система не выполняла своих прямых обязанностей — обслуживание клиентов. При этом система начала выходить из строя быстрее всего(Δ_{max}=70 мин=4200 с). Поэтому данной проблеме присвоен первый уровень критичности.

- Плохой код и неожиданно высокий трафик.

В рамках данных проблем системы выходила из строя примерно в одно время и это время достаточно велико, что значит, есть шанс его ручного обнаружения до отключения.

- Медленное время отклика и взаимные блокировки базы данных.

В рамках данных проблем отключения системы не происходило. Но данные проблемы все равно имели непосредственное негативное влияние на ключевые метрики, что соответственно видно из графиков.

Из таблиц видно, что в рамках поставленных проблем два алерта не сработали ни разу — «Критично мало свободного места на диске» и «Получаем большой объем трафика». Это ожидаемое поведение, так как никакая из этих проблем не связана с хранением или получением большого объема данных, поэтому и диск в данных инцидентах не участвует.

Таблица 1.

Итоговая таблица прихода алертов в зависимости от проблемы

Проблема \ Алерт	Начало инцидента	Отключение системы	Критично высокая загрузка CPU	Высокое потребление памяти
Поломка в DNS-запросах	00:46:50	01:57:13	—	—
Неожиданно высокий трафик	02:31:30	03:51:32	02:39:34	02:33:22
Плохой код (утечки памяти)	02:17:17	03:44:23	—	02:21:32
Медленное время отклика	23:44:25	—	—	—
Взаимные блокировки базы данных	01:28:35	—	—	—

Таблица 2.

Итоговая таблица прихода алертов в зависимости от проблемы

Проблема \ Алерт	Критично мало свободного места на диске	Время ответа сервиса превышает установленный порог	Процент ошибок сервиса на хэндлере больше критичного значения	Получаем большой объем трафика
Поломка в DNS-запросах	—	—	00:47:52	—
Неожиданно высокий трафик	—	02:34:10	—	—
Плохой код (утечки памяти)	—	—	02:29:11	—
Медленное время отклика	—	23:45:33	—	—
Взаимные блокировки базы данных	—	—	01:40:27	—

Так как для каждой проблемы есть несколько алертов, которые позволяют их идентифицировать, то нам необходимо вычислить значения среднего времени об-

наружения (MTTD) для определения, какой алерт раньше всех позволяет обнаружить проблему.

Формула для расчета данной метрики (2):

$$MTTD = \sum\left(t_{\text{обнаружено}} - t_{\text{произошло}}\right) / N_{\text{инцидентов}} \quad (2)$$

где $t_{\text{обнаружено}}$ — время обнаружения инцидента, $t_{\text{произошло}}$ — время происхождения инцидента и $N_{\text{индицентов}}$ — количество инцидентов.

Низкий MTTD позволяет команде поддержки быстрее принимать меры по восстановлению системы, минимизируя потенциальные негативные последствия для пользователей и бизнес-процессов.

Помимо этого, согласно принципу 2, так как наша система является эффективной, мы гарантируем, что алерт должен прийти до выхода из строя сервера и при этом время между возникновением проблемы и приходом алерта должно быть минимальным, мы должны определить Δ_{alert}, которое вычисляется по формуле (3):

$$\Delta_{alert} = \min\left(MTTD_{issue}\right) \quad (3)$$

где $\min(MTTD_{issue})$ — минимальное значение MTTD для конкретной проблемы (в строке).

Вычисленные значения MTTD представлены в табл. 3.

Таблица 3.

Значения MTTD

Проблема \ Алерт	Δ_{max}	Критично высокая загрузка CPU	Высокое потребление памяти	Время ответа сервиса превышает установленный порог	Процент ошибок сервиса на хэндлере больше критичного значения
Поломка в DNS-запросах	4200				62
Неожиданно высокий трафик	4800	484	112	160	
Плохой код (утечки памяти)	5220		255		714
Медленное время отклика	Ґ			68	
Взаимные блокировки базы данных	Ґ			552	712

Из таблицы сразу можно сделать вывод, что наша система — эффективнее человека, так как для каждой проблемы $\Delta_{alert} < \Delta_{max}$, что удовлетворяет принципу 2. Так же наша система удовлетворяет первому и третьему принципу: хотя бы один алерт должен прийти для каждой проблемы и данный набор алертов является минимальным и полностью покрывает все проблемы.

Далее определим приоритетность алертов. Как видно из таблицы, для самой приоритетной проблемы — «Поломка в DNS-запросах» — пришел только один алерт «Процент ошибок сервиса на хэндлере больше критичного значения». Следовательно, данный алерт будет самым приоритетным.

Из таблицы видно, что для обнаружения проблемы «Неожиданно высокий трафик» (2 уровень критичности) лучше всего подходит алерт «Высокое потребление памяти», так как он приходит раньше всех (минимальное значение MTTD в строке). К тому же это единственный алерт, который приходит при проблеме утечек памяти (2 уровень критичности). Следовательно, он имеет вторую приоритетность. Следующим должен прийти алерт «Критично высокая загрузка CPU», так как он тоже свидетельствует о проблеме 2 уровень критичности. Последним алертом по приоритетности будет «Время ответа сервиса превышает установленный порог», так как он свидетельствует о проблемах 3 уровня критичности. К тому же может говорить о многих проблемах, что затрудняет поиск проблемы разработчиком.

Аналитическим путем получили граничные значения для алертов, а также экспериментально их обосновали, которые представлены в табл. 4.

Таблица 4.

Граничные значения параметров

Использования процессора, %	80
Использования оперативной памяти, %	90
Использования диска, %	85
Среднего времени ответа сервиса, с	5 (для предупреждающего) 10 (для критичесокго)
Значение получаемого трафика, Мб	5 (для предупреждающего) 15 (для критического)
Неудачные запросы, %	1

Заключение

Таким образом, была спроектирована система оповещений для системы мониторинга веб-сервера, которая максимально эффективно подходит для отслеживания таких проблем, как медленное время отклика, поломка в DNS-запросах, взаимные блокировки базы данных, плохой код с утечками памяти и неожиданно высокий трафик.

Также был установлен рекомендованный порядок прихода алертов, который позволил наиболее эффективно отслеживать проблемы в соответствие с установленными принципами, а, именно:

1. процент ошибок сервиса на хэндлере больше критичного значения;
2. высокое потребление памяти;
3. критично высокая загрузка CPU;
4. время ответа сервиса превышает установленный порог.

ЛИТЕРАТУРА

1. What are the most common web application issues that you might face day-to-day in your application? [Электронный ресурс]. — URL: https://www.atatus.com/ask/what-are-the-most-common-web-application-issues, Дата обращения: 04.12.2023
2. Casalicchio, E. A study on performance measures for auto-scaling CPU-intensive containerized applications. Cluster Comput 22, 995–1006 (2019)
3. Fix an overloaded server [Электронный ресурс]. — URL: https://web.dev/articles/overloaded-server?hl=ru, Дата обращения: 04.12.2023
4. Igor Ljubuncic, Chapter 10 — Fine-tuning the system performance, Problem-Solving in High Performance Computing, Morgan Kaufmann, 2015, Pages 259–275, ISBN 9780128010198
5. Salmanian, Zolfaghar & Izadkhah, Habib & Isazadeh, Ayaz. (2017). Optimizing web server RAM performance using birth–death process queuing system: scalable memory issue. Journal of Supercomputing. 73. 1–18. 10.1007/s11227-017-2081-z.
6. How Much RAM For Web Server [Электронный ресурс]. — URL: https://robots.net/tech/how-much-ram-for-web-server/, Дата обращения: 04.12.2023
7. How Much Free Disk Space Should I Have On My Server? [Электронный ресурс]. — URL:/ https://conetix.com.au/support/how-much-free-disk-space-should-i-have-on-my-server/ Дата обращения: 04.12.2023
8. How to Determine the Correct Size and Type of a Web Server [Электронный ресурс]. — URL:/ https://hostadvice.com/how-to/web-hosting/how-to-determine-the-correct-size-and-type-of-a-web-server/ Дата обращения: 04.12.2023
9. Jakob Nielsen, «Usability Engineering (Interactive Technologies)» — NY.: Morgan Kaufmann, 1993, ISBN 978-0125184069
10. Bailey, R.W. (1982), Human Performance Engineering (1st Edition), Prentice-Hall: Englewood Cliffs, NJ.
11. Martin, G.L. and Corl, K.G. (1986), System response time effects on user productivity, Behaviour and Information Technology, 5(1), 3–13.
12. The Clean Architecture [Электронный ресурс]. — URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html, Дата обращения 14.03.2022
13. Availability in the Cross-Channel User Experience [Электронный ресурс]. — URL: https://www.nngroup.com/articles/available-cross-channel/, Дата обращения: 04.12.2023
14. Jader, Omid & Zeebaree, Subhi & Zebari, Rizgar. (2019). A State Of Art Survey For Web Server Performance Measurement And Load Balancing Mechanisms. International Journal of Scientific & Technology Research. 8. 535–543
15. Fung Po Tso, Simon Jouet, Dimitrios P. Pezaros, Network and server resource management strategies for data centre infrastructures: A survey, Computer Networks, Volume 106, 2016, Pages 209–225, ISSN 1389–1286

Журнал «Современная наука: актуальные проблемы теории и практики»