УДК 004.4

АНАЛИЗ ФУНКЦИОНАЛА ТЕХНОЛОГИЙ WEBSOCKET И LONG POLLING ПРИ ИХ ИСПОЛЬЗОВАНИИ ДЛЯ РАЗРАБОТКИ ЧАТ-ПРИЛОЖЕНИЙ

Ильичев В.Ю., Иванов Н.В.

ФГАОУ ВО «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)», Москва, e-mail: patrol8@yandex.ru

На современном этапе развития технологий чат-приложения получили огромную популярность благодаря ряду их неоспоримых достоинств при организации обмена информацией, таких как: удобство использования, групповое общение, обеспечение конфиденциальности, доступность, возможность обмена любым мультимедийным контентом. В отличие от традиционных форм коммуникации, чат-приложения позволяют вести диалоги в реальном времени, что значительно ускоряет обмен информацией. Целью исследования является выбор наилучшей технологии из двух популярных: WebSocket и Long Polling – для создания чат-приложений различного назначения. Разработанная методика выбора технологии основана на эксперименте, состоящем из определения основной метрики работы чат-приложений, созданных с использованием данных технологий, – времени доставки сообщения от отправителя к получателю. Для реализации процедур отправки и получения чат-сообщений выбраны программные средства языка Python (в частности, фреймворк Django), позволившие создать чат-приложения, основанные на разных принципах функционирования. В пределах разработки методики составлены листинги программ, реализующие представленные технологии, в том числе организацию выбранной структуры баз данных и логических принципов работы. Проведено нагрузочное тестирование приложений с варьированием количества устанавливаемых подключений. С помощью средств языка Python также реализована визуализация результатов тестирования, позволившая выполнить анализ полученных результатов. Сделан вывод, в каких случаях целесообразнее и эффективнее использовать WebSocket, а в каких – Long Polling. Результаты данной работы могут оказать существенную помощь разработчикам чат-приложений разнообразной направленности.

Ключевые слова: чат-приложение, WebSocket, Long Polling, язык Python, нагрузочное тестирование, время доставки сообщения

ANALYSIS OF FUNCTIONALITY OF WEBSOCKET AND LONG POLLING TECHNOLOGIES WHEN THEY ARE USED FOR DEVELOPMENT OF CHAT APPLICATIONS

Ilichev V.Yu., Ivanov N.V.

Bauman Moscow State Technical University, Moscow, e-mail: patrol8@yandex.ru

At the present stage of technology development, chat applications have gained immense popularity due to number of indisputable advantages in organizing the exchange of information: usability, group communication, ensuring confidentiality, accessibility, ability to exchange multimedia content. Unlike traditional forms of communication, chat applications allow real-time dialogue, which significantly speeds up the exchange of information. The purpose of the study is to choose the best technology from two popular ones: WebSocket and Long Polling, for creating chat applications for various purposes. The developed technology selection methodology is based on an experiment consisting of determining main metric of chat applications created using these technologies - the time of delivery of a message from sender to recipient. To implement procedures for sending and receiving chat messages, Python software tools (in particular, Django framework) were selected, which made it possible to create chat applications based on different principles of functioning. Within the development of methodology, listings of programs are compiled that implement presented technologies, including the organization of selected database structure and logical principles of operation. Load testing of applications was carried out with a variation in the number of connections to be established. With help of Python language tools, visualization of test results is also implemented, which made it possible to analyze the results obtained. It was concluded in which cases it is more expedient and more efficient to use WebSocket, and in which Long Polling. The results of this work can provide significant assistance to developers of chat applications of various orientations.

Keywords: chat application, WebSocket, Long Polling, Python language, message delivery time, load testing

Введение

В современном мире чат-приложения стали неотъемлемой частью общения. Они не только упрощают взаимодействие между людьми, но и открывают новые горизонты для бизнеса, образования и социальных связей [1].

Чат-приложения находят применение в самых различных сферах: в бизнесе они служат для оперативного обмена информацией и координации командной работы (например, с помощью Slack и Microsoft Teams); в образовании – для организации дистанционного обучения (например, с использованием Zoom и Google Classroom); а в социальной сфере – для поддержания связей с друзьями и семьей (например, в приложениях WhatsApp и Viber).

Исходя из этого, в отрасли веб-разработки чат-приложения стали неотъемлемой частью множества серьезных онлайн-проектов, и это объясняется несколькими ключевыми факторами. Во-первых, чат-приложения предоставляют возможность мгновенной обратной связи, позволяя пользователям задавать вопросы, получать поддержку и обмениваться информацией без задержек. Это особенно важно для онлайн-магазинов, сервисов поддержки клиентов и образовательных платформ.

Во-вторых, интеграция чат-приложений в веб-проекты способствует созданию более персонализированного опыта. Пользователи могут общаться с представителями компании или друг с другом, что создает ощущение сообщества и вовлеченности.

Кроме того, чат-приложения могут служить мощным инструментом для сбора данных о поведении пользователей. Анализ взаимодействий в чате позволяет компаниям лучше понимать потребности своей аудитории и адаптировать свои предложения под эти потребности. При этом возможна автоматизация процессов с помощью чат-ботов.

Наконец, с учетом роста популярности мобильных устройств, чат-приложения становятся важным элементом мобильных версий сайтов и приложений.

Таким образом, интеграция чат-приложений в веб-разработку не только отвечает требованиям пользователей к скорости и удобству общения, но и открывает новые возможности для бизнеса в плане взаимодействия с клиентами и анализа их поведения. В результате они становятся важным инструментом для повышения конкурентоспособности онлайн-проектов в условиях быстро меняющегося цифрового ландшафта.

Различные отрасли предъявляют к чат-приложениям большое количество всевозможных требований, поэтому веб-разработчики постоянно сталкиваются с проблемой выбора того или иного подхода. Среди таких подходов особой популярностью пользуются технологии WebSocket [2] и Long Polling [3].

WebSocket – это технология, обеспечивающая полноценную двустороннюю, асинхронную связь между клиентом и сервером в режиме реального времени, что позволяет обмениваться данными в обе стороны без необходимости постоянной отправки HTTP-запросов.

Long Polling – это метод, при котором клиент отправляет HTTP-запрос на сервер и держит соединение открытым до тех пор, пока не получит ответ или пока не истечет тайм-аут.

Каждое из двух рассмотренных решений имеет как достоинства, так и недостатки, которые всегда зависят от конкретной реализации.

Целью работы является сравнительный анализ двух описанных подходов к реализации чат-приложений путем разработки двух рабочих вариантов чат-приложений и проведения нагрузочного тестирования.

Материалы и методы исследования

В данной работе в качестве показателя для выбора технологии для создания чат-приложений выбрана одна из важнейших характеристик (метрик) – время, за которое сообщение доходит до получателя [4].

В качестве программных средств разработки чат-приложений были выбраны язык программирования Python [5] и наиболее используемый веб-фреймворк Django для него [6].

Реализация Django-приложения состоит из трех этапов: инициализация проекта и используемых библиотек, реализация структуры базы данных, реализация логики.

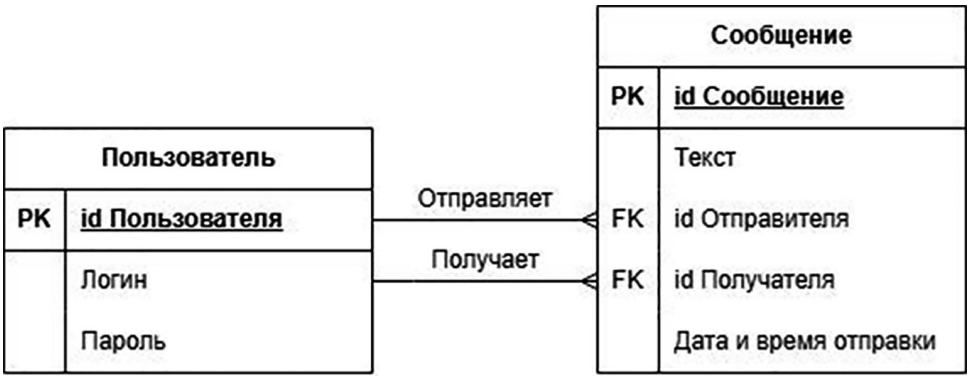

Рис. 1. Схема базы данных чат-приложения
Источник: составлено авторами

Вначале рассмотрим процедуру создания Long Polling чата. Первый описанный этап является одинаковым для всех Django-приложений, поэтому он не заслуживает особого внимания.

На рисунке 1 приведена схема, отображающая минимальную конфигурацию базы данных чат-приложения, обеспечивающую стабильную работу разрабатываемой системы.

Для реализации чата достаточно двух основных сущностей: «Пользователь» и «Сообщение». Пользователь имеет возможность авторизоваться и выбрать получателя сообщения. Получатель имеет доступ ко всем отправленным ему сообщениям. В качестве системы управления базой данных (СУБД) используется PostgreSQL [7].

После реализации базы данных можно приступить к реализации логики. Long Polling предполагает использование протокола HTTP, то есть основных запросов GET, POST и т.д. [8]. Вначале был реализован запрос создания сообщения. Данный запрос является стандартным POST, без каких-либо особенностей. Далее реализован запрос GET для получения новых сообщений. Именно он будет иметь непосредственное отношение к Long Polling. Разделим его реализацию на несколько этапов:

1) определение получателя из запроса;

2) определение последнего полученного сообщения, чтобы можно было узнать, какие сообщения являются новыми;

3) ожидание новых данных. На данном этапе происходит основная часть Long Polling соединения: на протяжении 20 секунд (стандартное значение) запрос ожидает новых сообщений. Если же новых сообщений не найдено, соединение прерывается. В проектной реализации Long Polling при отсутствии новых сообщений соединение необходимо инициализировать заново, но в рамках тестирования авторы старались избежать возникновения такой ситуации.

После реализации основных функций приложения производилось тестирование системы. В качестве инструмента для тестирования выбрана библиотека Locust [9]. Она предоставляет удобный веб-интерфейс для нагрузочного тестирования и способна дать необходимые для сравнения двух чат-приложений метрики. Locust тест представляет собой так называемые задачи, которые вызываются со случайной задержкой (в данном случае 0,5–1,5 с) во время работы теста. Также Locust предоставляет возможность выбора количества подключений к серверу и темпа прироста подключений (по умолчанию составляет 1 в секунду).

Для имитации Long Polling соединения была создана задача, в которой пользователь отправляет сообщение другому пользователю. Все запросы вызываются асинхронно с помощью средств библиотеки Gevent [10, 11], так как при отправке HTTP запрос блокирует свой поток.

Метрикой является время от отправки сообщения до получения его пользователем.

Далее рассмотрим процедуру реализации WebSocket чата. Первые два этапа создания Websocket чат-приложения полностью идентичны созданию Long Polling чат-приложения. Стоит лишь добавить, что для реализации Websocket в Django предусмотрена библиотека Django Channels, которая и является основой разрабатываемого приложения.

Для чистоты сравнения конфигурацию базы данных оставим такой же, как и в первом приложении.

WebSocket – это технология, предполагающая реализацию нескольких основных асинхронных методов. Первый из них – connect запрос, отвечающий за создание соединения между пользователями. Далее следует запрос disconnect, разрывающий соединение. И, наконец, два основных запроса – WebSocket – receive и send, совершающих главную работу. Они работают в следующем порядке:

1) отправка сообщения (send);

2) принятие и сохранение сообщения в базе данных (receive);

3) отправка информации о новом сообщении получателю (receive).

Также стоит отметить, что в приложение был интегрирован брокер сообщений – база данных Redis. Это позволило осуществить асинхронный прием сообщений, что разгружает веб-сервер и улучшает общую производительность. Данное решение является стандартным для WebSocket чат-приложения.

Для тестирования была создана Locust задача, в которой пользователь отправляет сообщение, а адресат принимает его. Так же как и в первом приложении, при тестировании фиксировалось время от отправки до получения сообщения.

Результаты исследования и их обсуждение

Для каждого из двух созданных чат-приложений проведено тестирование в 3 этапа по 5 повторений. Каждый этап характеризуется разным количеством подключений: 1, 20, 100 – и временем проведения: 5 минут для каждого повторения.

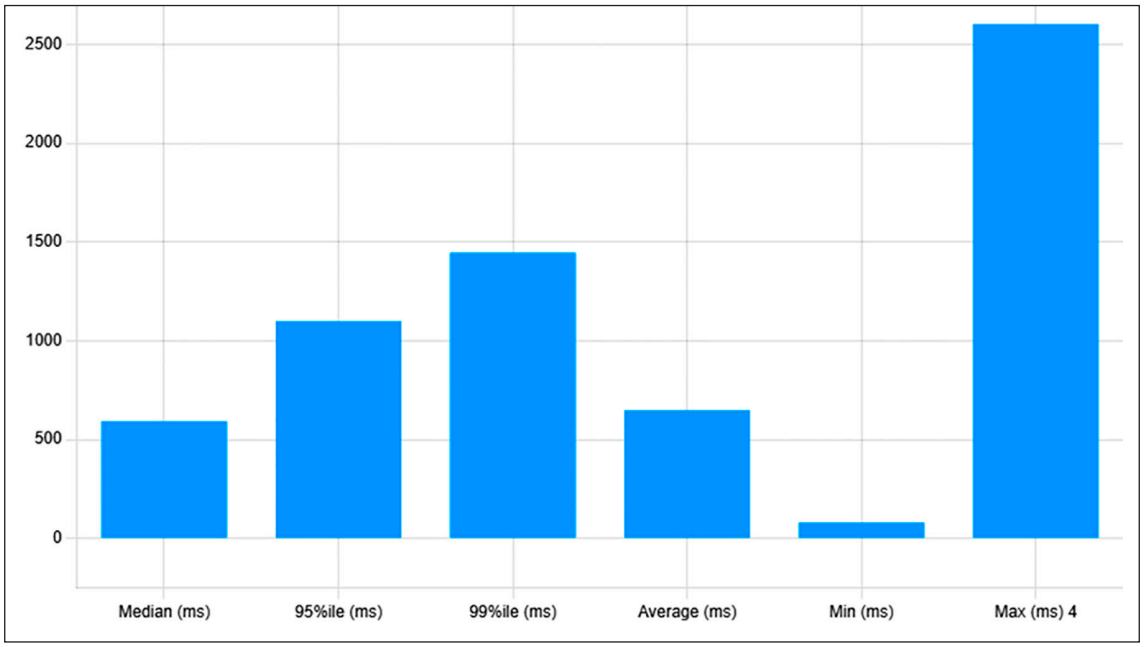

Рис. 2. Результаты метрик при тестировании Long Polling чат-приложения с одним подключением
Источник: составлено авторами

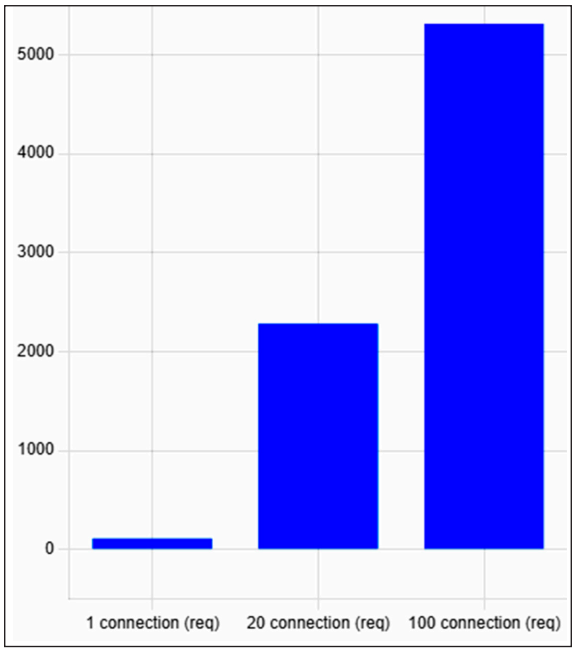

Рис. 3. Количество обработанных запросов Long Polling в зависимости от числа подключений
Источник: составлено авторами

Затем для каждого этапа вычислены значения метрик времени доставки сообщения (median – медианной, 95%ile – 95-процентильной, 99%ile – 99-процентильной, average – средней, min – минимальной, max – максимальной).

Результаты полученных значений данных метрик при тестировании Long Polling чат-приложения с одним подключением представлены на рисунке 2.

Аналогичное тестирование было проведено для Long Polling чат-приложения с 20 и 100 подключениями. Полученное количество обработанных запросов Long Polling в зависимости от числа подключений приведено на рисунке 3.

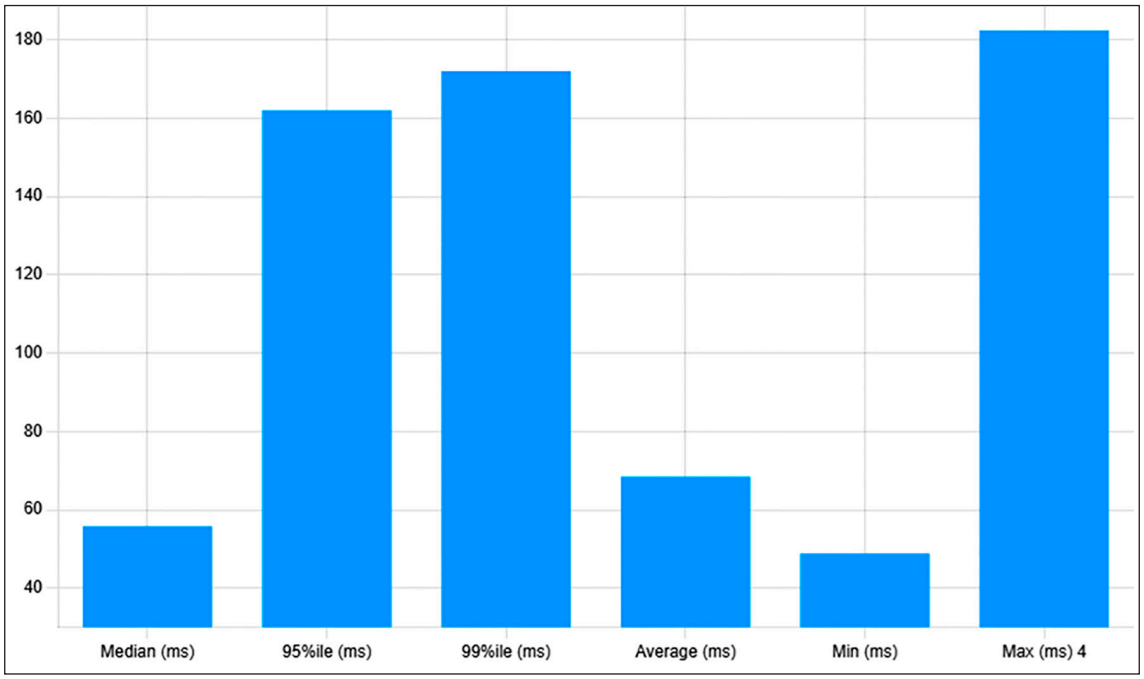

Рис. 4. Результаты метрик при тестировании WebSocket чат-приложения с одним подключением
Источник: составлено авторами

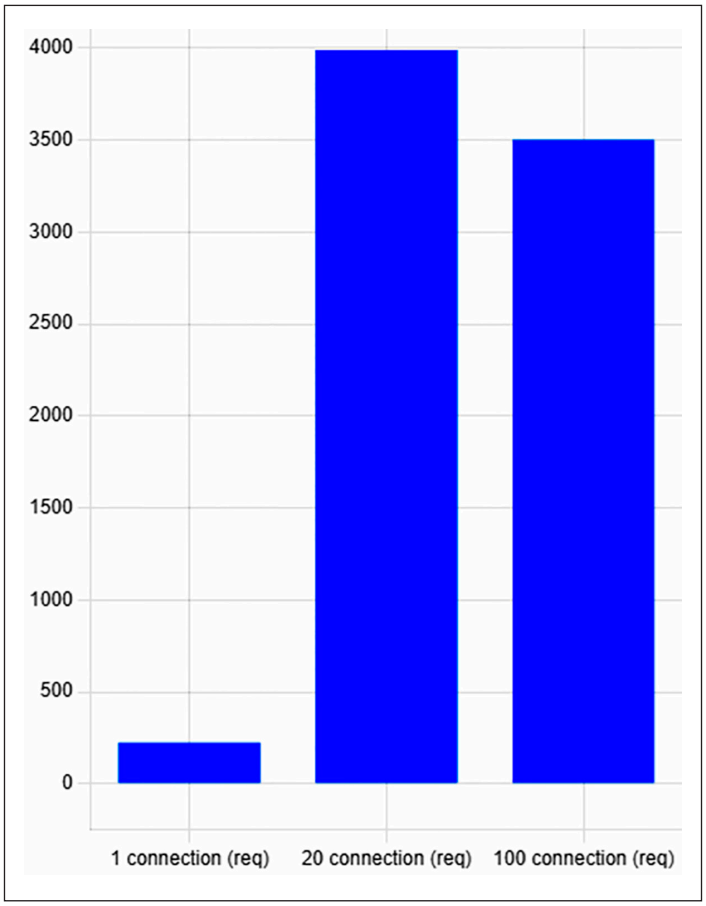

Рис. 5. Количество обработанных запросов WebSocket в зависимости от числа подключений
Источник: составлено авторами

По результатам тестирования Long Polling чат-приложений можно сделать следующий вывод: время доставки сообщения получателю и количество обработанных запросов имеют прямую пропорциональную зависимость от числа подключений.

Результаты полученных значений метрик при тестировании WebSocket чат-приложения с одним подключением представлены на рисунке 4.

Аналогичное тестирование было проведено для WebSocket чат-приложения с 20 и 100 подключениями. Полученное количество обработанных запросов WebSocket в зависимости от числа подключений приведено на рисунке 5.

Заключение

Сравним результаты тестирования двух реализованных технологий.

При увеличении числа подключений медианное и среднее время доставки сообщения Long Polling чат-приложения показывает резкий рост. Время WebSocket чат-приложения показывает устойчиво небольшой рост.

При увеличении числа подключений Long Polling чат-приложения показывает устойчивый рост числа обрабатываемых запросов, но WebSocket чат-приложение способно обрабатывать большее количество запросов.

По результатам тестирования WebSocket можно сделать следующее заключение: до определенного значения числа подключений количество обрабатываемых запросов прямо пропорционально числу подключений, но при дальнейшем его увеличении производительность падает. Отсюда следует вывод: для удержания большого количества WebSocket соединений необходимы соответствующие серверные мощности.

Таким образом, цель данного исследования, заключающаяся в сравнительном анализе двух подходов к реализации чат-приложений, является достигнутой. Разработаны два вида чат-приложений и проведено их нагрузочное тестирование, в результате которого получены графики зависимости времени доставки сообщений и количества выполненных запросов от числа подключений. По результатам сравнения результатов можно сказать, что более эффективным подходом является WebSocket, но при отсутствии достаточных серверных мощностей и небольшом числе подключений Long Polling также представляется приемлемым подходом.

Представленная работа может быть полезна как специалистам, реализующим различные технологии при разработке чат-приложений, так и для практического использования в проектной и научной деятельности.

Список литературы

1. Козориз А.В. Проблема персонализации в маркетинговых решениях современных компаний: роль чат-ботов // Экономика: вчера, сегодня, завтра. 2019. Т. 9, № 10-1. С. 649-656. DOI: 10.34670/AR.2020.91.10.073.

2. Хабаров С.П., Голубев К.С. Клиент-серверная экспертная система на основе технологии WebSocket // Информационные системы и технологии: теория и практика. 2017. С. 115-119. URL: https://spbftu.ru/uploads/the_science/general-science-information/publications/10-IST-part-1-2018.pdf (дата обращения: 20.03.2025).

3. Гридин В.Н., Анисимов В.И., Васильев С.А. Методы повышения производительности современных веб-приложений // Известия ЮФУ. Технические науки. 2020. № 2 (212). С. 193-200. DOI: 10.18522/2311-3103-2020-2-193-200.

4. Ураев Д.А. Метрики для оценки качества чат-бот приложений // Наука, техника и образование. 2019. № 9 (62). С. 36-40. URL: https://3minut.ru/images/PDF/2019/62/NTO-9-62-.pdf (дата обращения: 20.03.2025).

5. Ильичев В.Ю., Юрик Е.А. Разработка программы для нахождения оптимального распределения ресурсов с целью максимизации прибыли // Вектор экономики. 2021. № 5 (59). URL: http://www.vectoreconomy.ru/images/publications/2021/5/mathematicalmethods/Ilichev_Yurik.pdf (дата обращения: 20.03.2025).

6. Мизюков Г.С. Разработка системного программного обеспечения на основе чат-бота ChatGPT // Вестник компьютерных и информационных технологий. 2024. Т. 21, № 4 (238). С. 57-64. DOI: 10.14489/vkit.2024.04.pp.057-064.

7. Шлыков А.А., Антипкин А.С. Сравнительный анализ способов защиты информации в автоматизированных системах с использованием системы управления базами данных PostgreSQL // Инжиниринг и технологии. 2024. Т. 9, № 2. С. 1-3. DOI: 10.21685/2587-7704-2024-9-2-1.

8. Борисов П.Е. HTTP-протокол прикладного уровня (обзор) // Научный аспект. 2024. Т. 22, № 5. С. 2937-2943. URL: https://na-journal.ru/5-2024-informacionnye-tekhnologii/11528-http-protokol-prikladnogo-urovnya-obzor (дата обращения: 20.03.2025).

9. Баранова Е.М., Баранов А.Н., Борзенкова С.Ю., Шепаров И.С., Богданов А.М. Разработка безопасного веб-приложения для организации интернет-мероприятий // Известия Тульского государственного университета. Технические науки. 2024. № 9. С. 334-337. DOI: 10.24412/2071-6168-2024-9-334-335.

10. Ермаков О.А., Брозгунова Н.П. Python – как инструмент для анализа данных // Наука и Образование. 2020. Т. 3, № 4. С. 26. URL: https://opusmgau.ru/index.php/see/article/view/2461/2460. (дата обращения: 20.03.2025).

11. Ilichev V.Y. Development of program for determination of fractal dimensions of images. // International Research Journal. 2021. № 4-1 (106). С. 6-10. DOI: 10.23670/IRJ.2021.106.4.001.