УДК 681.327.1/13

Архитектурные подходы и технологические решения в создании инновационных веб-приложений для голосовых и текстовых чатов. Анализ, перспективы и реализация

Васильков Сергей Константинович — студент, бакалавр, кафедра «Информационные и вычислительные системы». E-mail: z2013zz@bk.ru

Забродин Андрей Владимирович — кандидат исторических наук, доцент, кафедра «Информационные и вычислительные системы». E-mail: zabrodin@pgups.ru

Петербургский государственный университет путей сообщения Императора Александра I, Россия, Санкт-Петербург

Для цитирования: Васильков С. К., Забродин А. В. Архитектурные подходы и технологические решения в создании инновационных веб-приложений для голосовых и текстовых чатов. Анализ, перспективы и реализация // Интеллектуальные технологии на транспорте. 2024. № 2 (38). С. 49–56. DOI: 10.20295/2413-2527-2024-238-49-56

Аннотация. *В статье на примере реализованного решения подробно рассматривается технологии, используемые в инновационном сервисе, реализующим голосовые и текстовые чаты в виде веб-приложения. При этом основное внимание уделяется анализу архитектурных и технологических подходов, используемых при разработке веб-приложений для голосовых и текстовых чатов. Достаточно подробно анализируются функциональные возможности и масштабируемость таких систем. Объектом для анализа служит сервис, с рабочим названием «Сервис с голосовым и текстовым чатами в контексте импортозамещения», имеющий рабочее название Ruscord, который несмотря на схожесть по функциональности и пользовательскому опыту с ведущими платформами общения, такими как Discord, Telegram, «ВКонтакте», реализован с учетом специфических требований и новейших технологий в области веб-разработки.*

В статье описываются ключевые компоненты системы, включая серверную логику, клиентские приложения и инфраструктуру данных. Особое внимание уделено вопросам интеграции современных технологий в реальные продукты, включая использование облачных сервисов, микросервисной архитектуры и современных фреймвоков и протоколов.

В рамках данного исследования анализируются не только технические аспекты, но и вопросы удобства использования, доступности и включенности, что позволяет создавать более эффективные и удовлетворяющие потребности пользователей сервисы.

Ключевые слова: *архитектурные подходы, технологические решения, веб-приложения, голосовые чаты, текстовые чаты, WebRTC, Socket, React, инновационные технологии, платформы общения.*

Введение

В условиях бурного развития цифровых технологий и возрастающих требований к коммуникационным инструментам, разработка веб-приложений для голосовых и текстовых чатов становится предметом особого внимания. Эти сервисы, представляющие собой сложные многоуровневые системы. Они включают в себя множество компонентов, каждый из которых требует тщательного анализа и интеграции современных технологий [1].

Статья структурирована следующим образом: в начале рассматривается общая модель системы, описывающая и перечисляющая ее основные компоненты. Далее каждый компонент будет рассмотрен в отдельности с точки зрения его функциональности, технических требований и взаимодействия с другими элементами системы. Отдельная часть посвящена механизмам соединения этих компонентов для обеспечения надежной и эффективной работы всего приложения. Особое внимание уделено обсуждению пользовательского интерфейса как ключевого аспекта, определяющего удобство и функциональность конечного продукта.

Таким образом, статья предоставит читателю полное представление о том, как строится современное веб-приложение для голосовых и текстовых чатов, акцентируя внимание на наиболее значимых аспектах и решениях, которые могут быть использованы для создания эффективных и безопасных коммуникационных решений.

Архитектура приложения

Архитектура компонентов реализованной системы представляет собой комплексный набор организационных принципов и структурных решений, ориентированных на обеспечение эффективного функционирования приложения. Эта архитектура включает в себя две ключевые составляющие: архитектуру компонентов сервера (рис. 1) и архитектуру компонентов клиента (рис. 2).

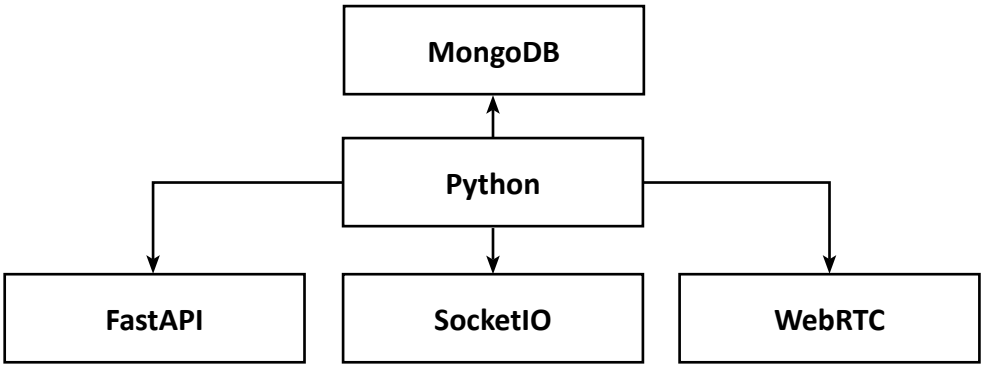

Рис. 1. Архитектура компонентов сервера

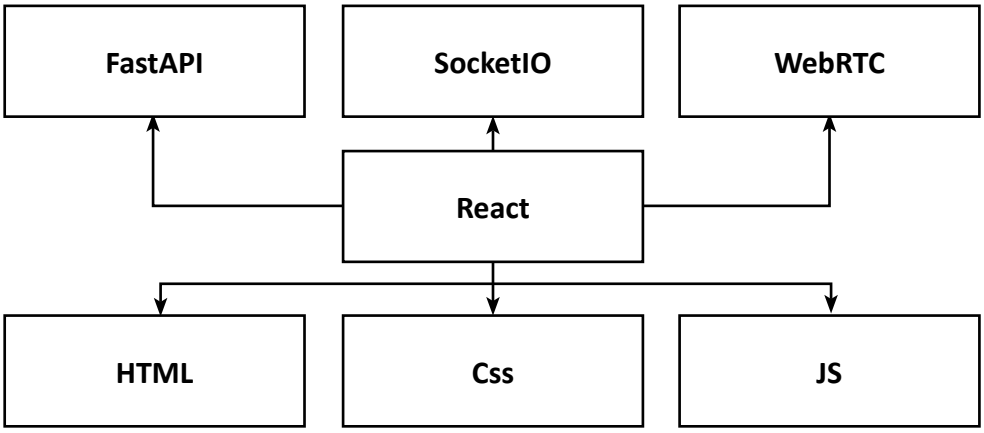

Рис. 2. Архитектура компонентов клиента

В представленной на рис. 1 и 2 архитектурах каждый из компонентов представляет собой определенную абстракцию, обеспечивающую конкретную функциональность и взаимодействие в рамках системы.

Python. Язык программирования, который служит основой для всего серверного взаимодействия и логики приложения, представляет собой ключевой элемент, определяющий функциональность и возможности системы в целом [2].

FastAPI. Современный, высокопроизводительный фреймворк для создания API с Python 3.7+, основанный на стандартных типах Python для валидации данных и аннотаций типов. FastAPI упрощает разработку и предоставляет автоматическую документацию в формате Swagger — инструмента, который позволяет автоматически создавать интерактивную документацию для вашего API.

SocketIO. Программная библиотека, позволяющая реализовывать веб-сокеты, то есть двусторонние и непрерывные каналы связи между клиентом и сервером. Этот механизм обеспечивает возможность серверу и клиентам обмениваться сообщениями в режиме реального времени.

WebRTC. Технология, позволяющая организовывать передачу потоковых данных (аудио-, видеофайлов) между браузерами без необходимости использования посредников [3]. В контексте реализованного сервиса WebRTC может быть использовано для реализации функционала видео- и аудиосвязи в приложении.

React. JavaScript-библиотека для разработки пользовательских интерфейсов. Она позволяет строить веб-интерфейсы, которые могут эффективно обновляться и управлять своим состоянием при взаимодействии с пользователем и сервером [4].

HTML, CSS, JS. Технологии, используемые для создания и стилизации веб-страниц на клиентской стороне. HTML определяет структуру содержимого страницы, CSS отвечает за ее визуальное оформление, а JavaScript управляет поведением страницы и обеспечивает асинхронное взаимодействие с сервером.

MongoDB. Современная, документоориентированная база данных, которая использует формат JSON-подобных документов со схемами. Она позволяет разработчикам строить масштабируемые приложения с гибкими запросами и индексацией. MongoDB предлагает мощные возможности агрегации и транзакций, поддерживая разнообразные операции данных. Это делает ее идеальной для приложений, требующих высокой производительности и легкой интеграции с большим количеством данных. MongoDB также поддерживает репликацию и шардинг, обеспечивая надежность и доступность данных в распределенных системах.

Алгоритм взаимодействия компонентов

В этой части работы рассматривается детальное взаимодействие между ключевыми компонентами системы, основываясь на диаграммах последовательности, описывается последовательность операций и обмен данных между серверными и клиентскими частями приложения. Это позволит лучше понять, как компоненты взаимодействуют друг с другом для предоставления необходимой функциональности пользователю. Рис. 3 наглядно иллюстрирует взаимодействие между ключевыми компонентами системы.

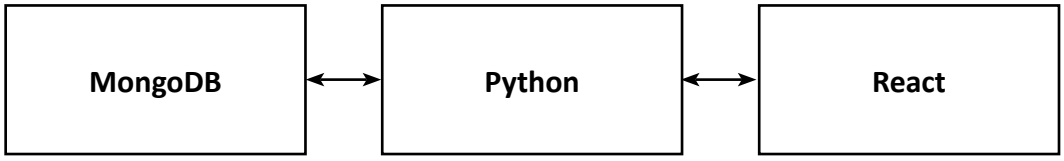

Рис. 3. Взаимодействие основных компонентов системы

Клиент и сервер непрерывно обмениваются данными через сокеты, позволяя приложению работать в режиме реального времени, что идеально подходит, например, для чатов, игр и совместной работы в интернете.

Алгоритм взаимодействия:

1. Python использует MongoDB для хранения и управления данными через библиотеку pymongo, которая представляет собой драйвер для работы с MongoDB. Это позволяет Python-приложениям легко взаимодействовать с базой данных, выполнять запросы, обновления, удаление и агрегацию данных [5]. MongoDB поддерживает асинхронное взаимодействие с базой данных, что совместимо с асинхронной природой FastAPI, улучшая производительность приложений, работающих в реальном времени, и позволяет связывать одну базу с несколькими серверами.

2. Сервер на Python использует FastAPI для обработки HTTP-запросов и может отправлять или получать данные в реальном времени через SocketIO [6].

3. WebRTC используется для прямой передачи потоковых данных между пользователями (например, для видеозвонков) [7].

4. React на клиентской стороне общается с сервером через HTTP и веб-сокеты для получения данных и управления состоянием интерфейса.

5. HTML, CSS и JavaScript используются для построения и оформления пользовательского интерфейса, который отображается в веб-браузере.

Клиент и сервер непрерывно обмениваются данными через сокеты, позволяя приложению работать в режиме реального времени, что идеально подходит, например, для чатов, игр и совместной работы в интернете.

Рассмотрим примеры реакций сервера на отдельные действия пользователя при взаимодействии с приложением.

На диаграмме взаимодействия (рис. 4) показан процесс отправки сообщений.

Алгоритм обмена сообщениями в реальном времени через веб-приложение включает следующие шаги:

1. Пользователь входит на сайт, авторизуется и открывается Socket-соединение.

2. Пользователь пишет сообщение и нажимает кнопку «отправить».

3. На сервер FastAPI приходит POST-запрос с сообщением в теле запроса [8].

4. Сервер проверяет состояние авторизации и записывает сообщение в БД.

5. Сервер отправляет сообщение в сокет клиента, находя его Socket ID по User ID.

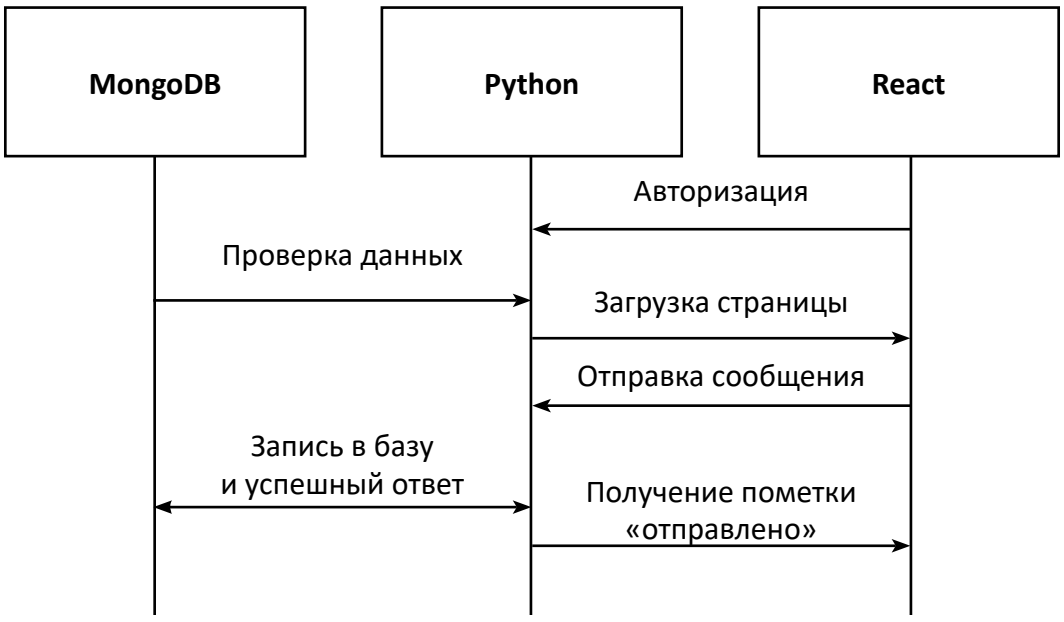

Рис. 4. Диаграмма взаимодействий при отправке сообщения

На диаграмме взаимодействия (рис. 5) продемонстрирован процесс добавления в друзья между двумя клиентами.

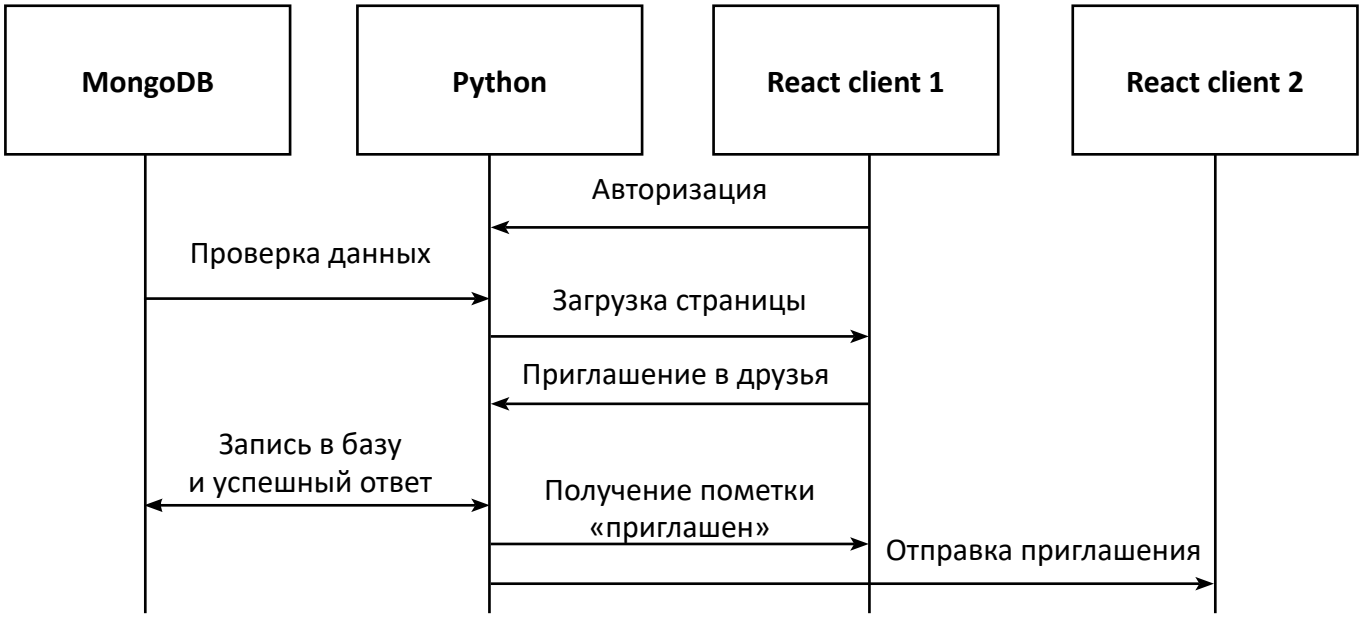

Рис. 5. Диаграмма взаимодействия при приглашении в друзья

Алгоритм включает следующие шаги:

1. Пользователь входит на сайт, авторизуется и открывается Socket-соединение.

2. Пользователь открывает список друзей и заходит во вкладку приглашения друзей.

3. Сервер присылает список пользователей в соответствии с введенным запросом пользователя.

4. Пользователь приглашает другого пользователя, нажимая на специальную кнопку в веб-интерфейсе.

5. На сервер FastAPI приходит POST-запрос с ID приглашенного пользователя в теле запроса.

6. Сервер проверяет состояние авторизации и записывает приглашение в друзья в БД [9].

7. Сервер отправляет сообщение в сокет клиента, находя его Socket ID по User ID, чтобы сменить кнопку приглашения на галочку. Так человек понимает, что он успешно пригласил другого пользователя.

8. Приглашение доходит до приглашенного человека через Socket ID, если он в сети.

Одним из ключевых преимуществ реализованной программы является ее способность функционировать в мультисерверной среде. Это означает, что система поддерживает одновременную работу нескольких бэкэнд-серверов, которые совместно используют одну базу данных MongoDB. Такая архитектура обеспечивает высокую масштабируемость и отказоустойчивость приложения (рис. 6).

Благодаря встроенным в код модулям синхронизации взаимодействие backend- и frontend-серверов происходит без сбоев и задержек. Эти модули отвечают за согласованность данных и предотвращение конфликтов, что особенно важно в многосерверной среде. Использование механизмов распределенного кэширования позволяет минимизировать задержки при доступе к данным. Таким образом, независимо от количества подключенных backend-серверов, данные в базе данных остаются согласованными, а пользователи получают стабильный и бесперебойный доступ к функционалу приложения. Следует отметить также, что данное решение не только подчеркивает гибкость и адаптивность системы, но и демонстрирует ее способность к оперативной реакции на изменяющиеся условия.

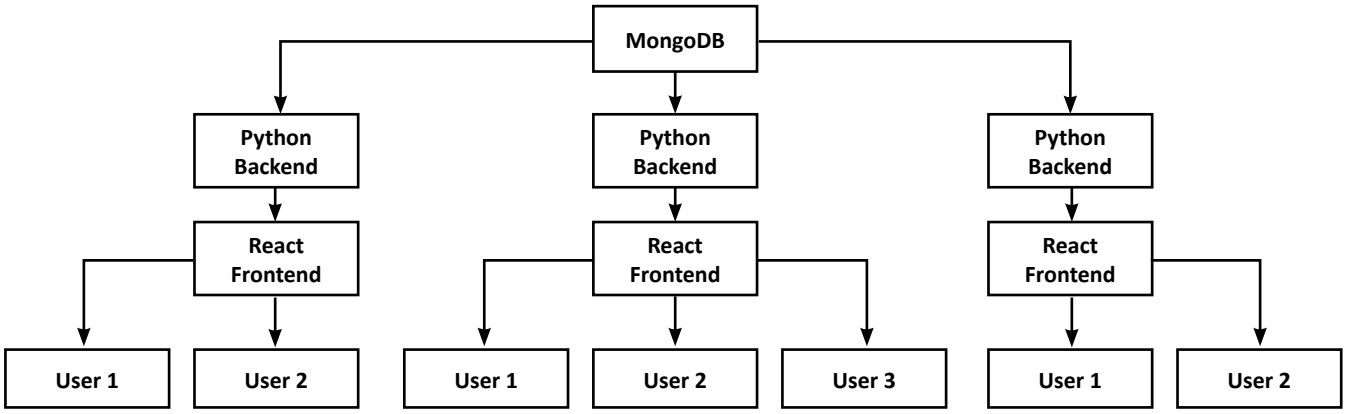

Рис. 6. Диаграмма мультисерверной реализации

В результате программа способна поддерживать высокие скорости обработки данных и стабильную работу даже при значительном увеличении количества пользователей и объема обрабатываемой информации.

Подводя итог вышесказанному: главная ценность проекта состоит в реализованном алгоритме взаимодействия между различными программными компонентами системы для обеспечения функциональности приложения. Именно правильная реализация и оптимизация взаимодействия обеспечивают высокую производительность, стабильность и отзывчивость системы, что является ключевым фактором в удовлетворении потребностей пользователей и достижении поставленных целей проекта.

Заключение

Следует отметить, что разработка инновационного веб-сервиса для голосовых и текстовых чатов требует комплексного подхода к выбору технологий и архитектурных решений. Проанализированный в статье проект демонстрирует, как современные технологические решения, такие как Python, FastAPI, SocketIO, WebRTC, React, а также MongoDB, могут быть эффективно интегрированы для создания масштабируемой, удобной и функционально удобной платформы. Это позволяет не только достичь высокого уровня пользовательского опыта, аналогичного ведущим мировым платформам, но и учитывать специфические требования и контексты использования, включая аспекты импортозамещения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Автоматизация, безопасность, онлайн-знакомства: для чего еще используют голосовые технологии в 2021 году. URL: https://habr.com/ru/articles/558774/ (дата обращения: 10.04.2024).

2. Федоров Д. Ю. Программирование на языке высокого уровня Python: учебное пособие для прикладного бакалавриата / 2-е изд., перераб. и доп. М.: Юрайт, 2019. 161 с. URL: https://urait.ru/bcode/437489 (дата обращения: 26.04.2024).

3. Streaming protocols and ultra-low latency including #webrtc. URL: https://webrtcbydralex.com/index.php/2018/05/15/streaming-protocols-and-ultra-low-latency-including-webrtc/ (дата обращения: 20.04.2024).

4. Top-10 Best Voice Chat APIs for Mobile and Web Apps. URL: https://habr.com/ru/articles/538150/ (дата обращения: 10.04.2024).

5. Рамальо Л. Совершенный Python. Пер. с англ. / СПб.: Питер, 2017. 800 с.

6. Как создать многопользовательский чат с помощью WebSocket. URL: https://dzen.ru/a/Y-AXCcMKtSLcYSJx (дата обращения: 10.04.2024).

7. Просто о WebRTC. URL: https://forasoft.github.io/webrtc-in-plain-russian/ (дата обращения: 10.04.2024).

8. Лутц М. Программирование на Python. Пер. с англ. / СПб.: Символ-Плюс, 2011. Том I, 4-е издание. 992 с.

9. Лутц М. Программирование на Python. Пер. с англ. / СПб.: Символ-Плюс, 2011. Том II, 4-е издание. 992 с.

Дата поступления: 28.05.2024
Решение о публикации: 17.06.2024

Architectural Approaches and Technological Solutions in Creating Innovative Web Applications for Voice and Text Chats. Analysis, Prospects, and Implementation

Sergey K. Vasilkov — Student, Bachelor, Department of Information and Computing System.
E-mail: z2013zz@bk.ru

Andrey V. Zabrodin — Candidate of Historical Sciences, Associate Professor.
E-mail: zabrodin@pgups.ru

Emperor Alexander I Petersburg State Transport University, Saint Petersburg, Russia

For citation: Vasilkov S. K., Zabrodin A. V. Architectural Approaches and Technological Solutions in Creating Innovative Web Applications for Voice and Text Chats. Analysis, Prospects, and Implementation // Intelligent technologies on transport. 2024. No. 2 (38). P. 49–56. (In Russian). DOI: 10.20295/2413-2527-2024-238-49-56

Abstract. *The article provides a detailed examination of the technologies used in an innovative service that implements voice and text chats as a web application, using an already implemented solution as an example. The main focus is on analyzing the architectural and technological solutions employed in the development of innovative web applications for voice and text chats. The article extensively discusses the functional capabilities, scalability of such systems. The service, provisionally named "Voice and Text Chat Service in the Context of Import Substitution", serves as the object of analysis. Despite its functional similarities and user experience with leading communication platforms such as Discord, Telegram, and VKontakte, it is developed with specific requirements and the latest technologies in web development in mind.*

The article meticulously reviews the key components of the system, including server logic, client applications, and data infrastructure. Special attention is given to the integration of modern technologies into real products, including the use of cloud services, microservices architecture, and contemporary frameworks and protocols.

The analysis covers not only technical aspects but also issues of usability, accessibility, and inclusivity, enabling the creation of more effective services that meet user needs. The article also highlights potential directions for further development of such platforms and presents examples of best practices in this field.

Keywords: *architectural approaches, technological solutions, web applications, voice chats, text chats, WebRTC, Socket, React, innovative technologies, communication platforms.*

REFERENCES

1. Avtomatizaciya, bezopasnost', onlajn-znakomstva: dlya chego eshche ispol'zuyut golosovye tekhnologii v 2021 godu. URL: https://habr.com/ru/articles/558774/ (data obrashcheniya: 10.04.2024). (In Russian)

2. Fedorov D. YU. Programmirovanie na yazyke vysokogo urovnya Python: uchebnoe posobie dlya prikladnogo bakalavriata / 2-e izd., pererab. i dop. M.: YUrajt, 2019. 161 s. URL: https://urait.ru/bcode/437489 (data obrashcheniya: 26.04.2024). (In Russian)

3. Streaming protocols and ultra-low latency including #webrtc. URL: https://webrtcbydralex.com/index.php/2018/05/15/streaming-protocols-and-ultra-low-latency-including-webrtc/ (дата обращения: 20.04.2024).

4. Top-10 Best Voice Chat APIs for Mobile and Web Apps. URL: https://habr.com/ru/articles/538150/ (data obrashcheniya: 10.04.2024).

5. Ramal'o L. Sovershennyj Python. Per. s angl. / SPb.: Piter, 2017. 800 s. (In Russian)

6. Kak sozdat' mnogopol'zovatel'skij chat s pomoshch'yu WebSocket. URL: https://dzen.ru/a/Y-AXCcMKtSLcYSJx (data obrashcheniya: 10.04.2024). (In Russian)

7. Prosto o WebRTC. URL: https://forasoft.github.io/webrtc-in-plain-russian/ (data obrashcheniya: 10.04.2024). (In Russian)

8. Lutc M. Programmirovanie na Python. Per. s angl. / SPb.: Simvol-Plyus, 2011. Tom I, 4-e izdanie. 992 s. (In Russian)

9. Lutc M. Programmirovanie na Python. Per. s angl. / SPb.: Simvol-Plyus, 2011. Tom II, 4-e izdanie. 992 s. (In Russian)

Received: 28.05.2024
Accepted: 17.06.2024