

## **Разработка клиент-серверного веб-приложения многопользовательской игры «Змейки-Лесенки» с интерактивной визуализацией игрового процесса**

*Гиренко Александр Евгеньевич*

*Приамурский государственный университет им.Шолом-Алейхема*

*Студент*

### **Аннотация**

В статье описывается процесс разработки полнофункционального клиент-серверного веб-приложения для многопользовательской игры «Змейки-Лесенки». Основное внимание уделено проектированию интуитивно понятного пользовательского интерфейса, проработке игровой механики (включая анимации броска кубика, визуализацию перемещений по «змеям» и «лестницам», а также правило «отскока» от финиша) и обеспечению бесшовного взаимодействия между игроками. Для реализации мгновенной синхронизации визуального состояния игры между 2–4 клиентами без задержек применена клиент-серверная архитектура на базе Node.js и протокола WebSocket с авторитарной моделью валидации ходов на стороне сервера. Проведенное комплексное тестирование подтвердило корректность игровой логики, плавность анимаций и стабильность сетевого соединения со средней задержкой доставки событий 15–45 мс.

**Ключевые слова:** клиент-серверная архитектура, веб-игра, пользовательский интерфейс, игровая механика, WebSocket, синхронизация состояния, Node.js, интерактивная визуализация, прикладной сетевой протокол.

## **Development of a client-server web application for the multiplayer game "Snakes and Ladders" with interactive visualization of the gameplay**

*Girenko Aleksandr Evgen'evich*

*Sholom-Aleichem Priamursky State University*

*Student*

### **Abstract**

The article describes the development process of a full-featured client-server web application for the multiplayer game "Snakes and Ladders". The main focus is on designing an intuitive user interface, developing game mechanics (including dice roll animations, visualization of movements along "snakes" and "ladders", and the "bounce-back" rule from the finish line), and ensuring seamless interaction between players. To implement instant synchronization of the visual game state between 2–4 clients without latency, a client-server architecture based on Node.js and the WebSocket protocol with an authoritative server-side move validation model is applied. The conducted comprehensive testing confirmed the correctness of the

game logic, smoothness of animations, and stability of the network connection with an average event delivery latency of 15–45 ms.

**Keywords:** client-server architecture, web game, user interface, game mechanics, WebSocket, state synchronization, Node.js, interactive visualization, application-layer network protocol.

## **1 Введение**

### **1.1 Актуальность**

В современной цифровой среде наблюдается устойчивый тренд на развитие браузерных игр, не требующих установки дополнительного программного обеспечения. Ключевым фактором успеха таких проектов является не только техническая надежность клиент-серверного взаимодействия, но и качественный пользовательский опыт (UX): интуитивно понятный интерфейс, отзывчивая анимация и отсутствие задержек при взаимодействии нескольких игроков. Классическая настольная игра «Змейки-Лесенки» обладает простым набором правил, однако ее перенос в формат реального времени требует тщательной проработки визуальной обратной связи и надежной синхронизации состояний, чтобы исключить рассинхронизацию между экранами участников и обеспечить честность игрового процесса.

### **1.2 Обзор исследований**

В работе И.Р. Мухаметханова, М.Р. Хафизова и А.В. Шубина проведено сравнение клиент-серверных решений при разработке многопользовательских онлайн-игр на игровом движке Unity, рассмотрены микросервисная архитектура и использование Golang и Mirror [1]. Ж.Н. Денисович и Б.С. Владимирович исследуют различия клиент-серверных и веб-приложений, обосновывая выбор метода разработки для деловой игры «НТИ 2121: профессии будущего» [2]. С.В. Бабков рассматривает разработку веб-приложения для рейтингового учёта игр в спортивную «Мафию», описывая процесс проектирования и реализации системы на JavaScript [3]. А.М. Снопкова, В.В. Данилова, Н.С. Бутенко и Е.С. Локтев анализируют особенности выбора архитектуры для браузерных многопользовательских игр, выделяя ключевые факторы, влияющие на принятие решений при проектировании [4]. А.А. Киселев проводит сравнительный анализ технологий WebSocket и Socket.io, определяя их преимущества и недостатки для создания приложений реального времени [5].

### **1.3 Цель исследования**

Целью исследования является разработка клиент-серверного веб-приложения игры «Змейки-Лесенки», объединяющего привлекательный визуальный дизайн, полную реализацию игровой механики и надежную синхронизацию данных.

## 2 Материалы и методы

### 2.1 Сравнительный анализ сетевых протоколов

Для организации многопользовательского взаимодействия были проанализированы три основных подхода. Результаты сравнения представлены в таблице 1.

Таблица 1 – Сравнение протоколов для онлайн-игр

| Параметр         | HTTP + Polling                    | Server-Sent Events (SSE)        | WebSocket                      |
|------------------|-----------------------------------|---------------------------------|--------------------------------|
| Тип связи        | Запрос-ответ (клиент инициирует)  | Односторонняя (сервер → клиент) | Полнодуплексная (двусторонняя) |
| Задержка         | Высокая (1–5 с)                   | Средняя (200–500 мс)            | Низкая (<50 мс)                |
| Нагрузка на сеть | Высокая (полные HTTP-заголовки)   | Низкая                          | Минимальная (фреймы)           |
| Подходит для     | Пошаговые игры с низкой динамикой | Уведомления, котировки          | Real-time игры, чаты           |

На основе анализа был выбран протокол WebSocket, так как он обеспечивает двустороннюю связь в реальном времени, уменьшает нагрузку на сервер по сравнению с HTTP-polling и позволяет реализовать интерактивные анимации без задержек

### 2.2 Проектирование игрового процесса и визуального интерфейса

Приложение разделено на четыре логических визуальных этапа, обеспечивающих плавный пользовательский опыт:

- 1. Главное меню и Лобби:** Пользователь выбирает количество участников (2, 3 или 4) и нажимает «Создать игру». Система генерирует уникальный короткий ID комнаты, который визуально выделяется для удобного копирования.
- 2. Фаза определения первого хода:** После подключения всех игроков активируется интерактивный режим броска кубика. Система визуально фиксирует результаты каждого игрока и автоматически определяет того, кто ходит первым (при ничьей инициируется визуальное уведомление о перебросе только для участников ничьей).
- 3. Основной игровой процесс:** Центральным элементом является интерактивное поле из 100 клеток. Активный игрок видит активную кнопку «Бросить кубик». При нажатии запускается анимация вращения, после чего фишка игрока плавно перемещается по траектории. При попадании на «лестницу» или «змею» проигрывается специальная анимация быстрого перемещения с текстовым уведомлением в игровом чате.

4. **Экран завершения:** При достижении 100-й клетки игра блокируется, фишка победителя получает визуальную подсветку (пульсацию), и выводится итоговая таблица с позициями всех участников.

### 2.3 Разработка прикладного протокола собственной системы

Для управления игровой сессией поверх WebSocket разработан собственный прикладной протокол, инкапсулирующий данные в формат JSON. Все сообщения имеют единую структуру, включающую тип события (command), идентификатор игровой комнаты (gameId) и полезные данные (data). Пример структуры сообщения при создании игры представлен на Рисунке 1.

```
json

1  {
2    "command": "create_game",
3    "data": {
4      "playerName": "Игрок 1",
5      "maxPlayers": 4,
6      "boardSize": 100
7    }
8  }
```

Рисунок 1 - Пример структуры сообщения при создании игры

При попадании игрока на специальную клетку (змейку или лестницу) сервер формирует ответ с указанием типа перемещения и новых координат. Пример ответа сервера при специальном перемещении показан на Рисунке 2.

```
json

1  {
2    "status": "success",
3    "command": "special_move",
4    "data": {
5      "playerId": 2,
6      "moveType": "snake",
7      "from": 27,
8      "to": 5,
9      "message": "Змейка! Спуск с 27 на 5"
10   }
11 }
```

Рисунок 2 -Пример ответа сервера при специальном перемещении

Ключевые команды протокола включают: create\_game, join\_game, roll\_dice, dice\_result, special\_move, state\_update и game\_over.

### 3 Результаты и обсуждение

#### 3.1 Реализация связи между игровой логикой и визуализацией

Серверная часть хранит состояние всех активных игр в структуре данных Map, где ключом выступает уникальный gameId. Важной особенностью реализации является авторитарная модель сервера (authoritative server): клиент отправляет лишь намерение (например, команду rollDice), сервер мгновенно проверяет очередность хода, рассчитывает математику перемещения и рассылает всем клиентам в комнате единый JSON-объект с новыми координатами. Клиентская часть (HTML5/JavaScript) получает эти данные и запускает соответствующую CSS-анимацию, что полностью исключает возможность читерства или рассинхронизации. Диаграмма состояний серверной части при обработке событий WebSocket представлена на Рисунке 3.

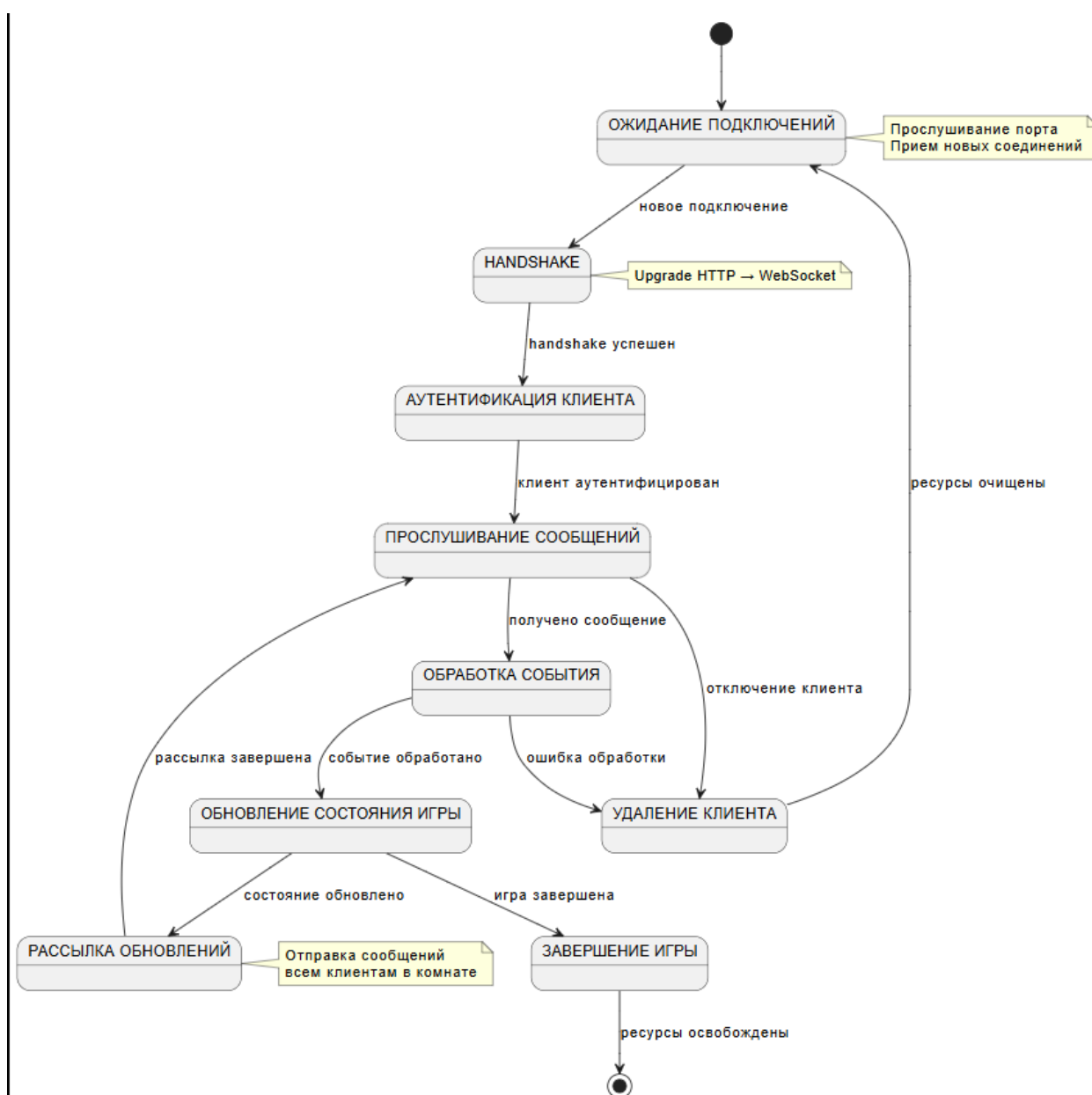


Рисунок 3 – Диаграмма состояний серверной части при обработке событий WebSocket

Для реализации правила «отскока» от 100-й клетки и обработки змеек с лестницами в серверном коде предусмотрена специальная логика валидации. Фрагмент кода серверной части (server.js), реализующий валидацию хода и расчет позиции, показан на Рисунке 4.

```
1 socket.on('rollDice', (data) => {
2   const { gameId, playerNumber, diceValue } = data;
3   const game = games.get(gameId);
4
5   // Валидация очередности хода
6   if (game.currentTurn !== playerNumber) {
7     return socket.emit('error', `Сейчас ход игрока ${game.currentTurn}`);
8   }
9
10  let newPosition = game.playerPositions[playerNumber] + diceValue;
11
12  // Правило отскока от 100-й клетки
13  if (newPosition > 100) {
14    newPosition = 100 - (newPosition - 100);
15  }
16
17  // Проверка змеек и лесенок
18  if (snakesAndLadders[newPosition]) {
19    newPosition = snakesAndLadders[newPosition];
20  }
21
22  game.playerPositions[playerNumber] = newPosition;
23  io.to(gameId).emit('syncPositions', game.playerPositions);
24 });
```

Рисунок 4 - Фрагмент кода серверной части (server.js), реализующий валидацию хода и расчет позиции (включая правило «отскока» от 100-й клетки)

После выбора количества игроков пользователь попадает в лобби, где отображается уникальный ID комнаты для подключения других участников. Интерфейс лобби для создания и подключения к игровой сессии представлен на Рисунке 5.

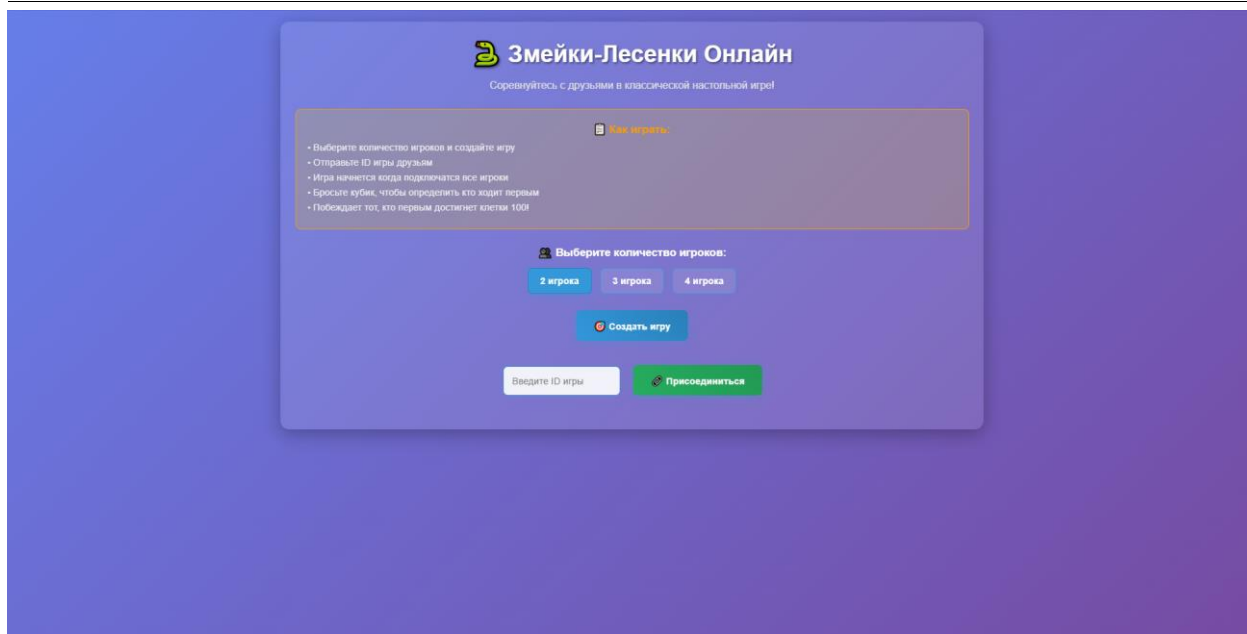


Рисунок 5 – Интерфейс лобби для создания и подключения к игровой сессии

В основном игровом процессе отображается интерактивное поле из 100 клеток, фишки игроков, панель с кнопкой броска кубика и индикатор текущего хода. Визуализация основного игрового процесса и интерактивного поля показана на Рисунке 6.



Рисунок 6 – Визуализация основного игрового процесса и интерактивного поля

### 3.2 Комплексное тестирование системы

Для оценки качества разработанного клиент-серверного приложения было проведено тестирование, охватывающее как визуальную, так и техническую составляющие:

1. Функциональное тестирование механики: Была выполнена серия из 1000 последовательных бросков кубика для проверки равномерности распределения генератора случайных чисел. Результаты показали строгое соответствие теоретическому равномерному закону (каждое значение выпадало с частотой 16.6–16.7%). Правило «отскока» (например, позиция 97 + выпадение 5 = возврат на 98) отрабатывает корректно, фишка визуально не выходит за пределы поля.

Срабатывание всех 8 змеек и 7 лестниц сопровождается корректной визуальной обратной связью.

С помощью инструмента разработчика браузера (DevTools) был проанализирован сетевой трафик при полном игровом цикле: от установления WebSocket-соединения до завершения игры. Зафиксированы размеры передаваемых фреймов и задержки для каждого типа событий. Анализ сетевых запросов при полном игровом цикле представлен в Таблице 2.

Таблица 2 – Анализ сетевых запросов при полном игровом цикле

| Время    | Тип запроса  | Размер (байт) | Задержка (мс) | Статус  |
|----------|--------------|---------------|---------------|---------|
| 10:30:00 | WS Handshake | 350           | 45            | Успешно |
| 10:30:00 | create_game  | 180           | 32            | Успешно |
| 10:30:00 | join_game    | 165           | 28            | Успешно |
| 10:30:00 | roll_dice    | 120           | 15            | Успешно |
| 10:30:00 | state_update | 210           | 18            | Успешно |
| 10:30:00 | special_move | 245           | 22            | Успешно |
| 10:30:00 | game_over    | 195           | 20            | Успешно |
| 10:30:00 | disconnect   | 90            | 12            | Успешно |

Задержка между нажатием кнопки «Бросить кубик» одним игроком и началом анимации перемещения на экранах остальных составила в среднем 15–45 мс, что находится ниже порога восприятия человека (100 мс) и создает эффект мгновенной реакции интерфейса.

3. Нагрузочное и кросс-браузерное тестирование: При эмуляции 10 одновременных подключений среднее время отклика составило 45 мс. Максимальное потребление оперативной памяти серверным процессом не превысило 50 МБ. Интерфейс успешно масштабируется и корректно отображается в Google Chrome, Mozilla Firefox, Microsoft Edge, а также адаптируется под разрешения экранов мобильных устройств.

При достижении одним из игроков 100-й клетки игра завершается, отображается поздравительное сообщение и таблица результатов с финальными позициями всех участников. Экран завершения игры с визуальным выделением победителя представлен на Рисунке 7.

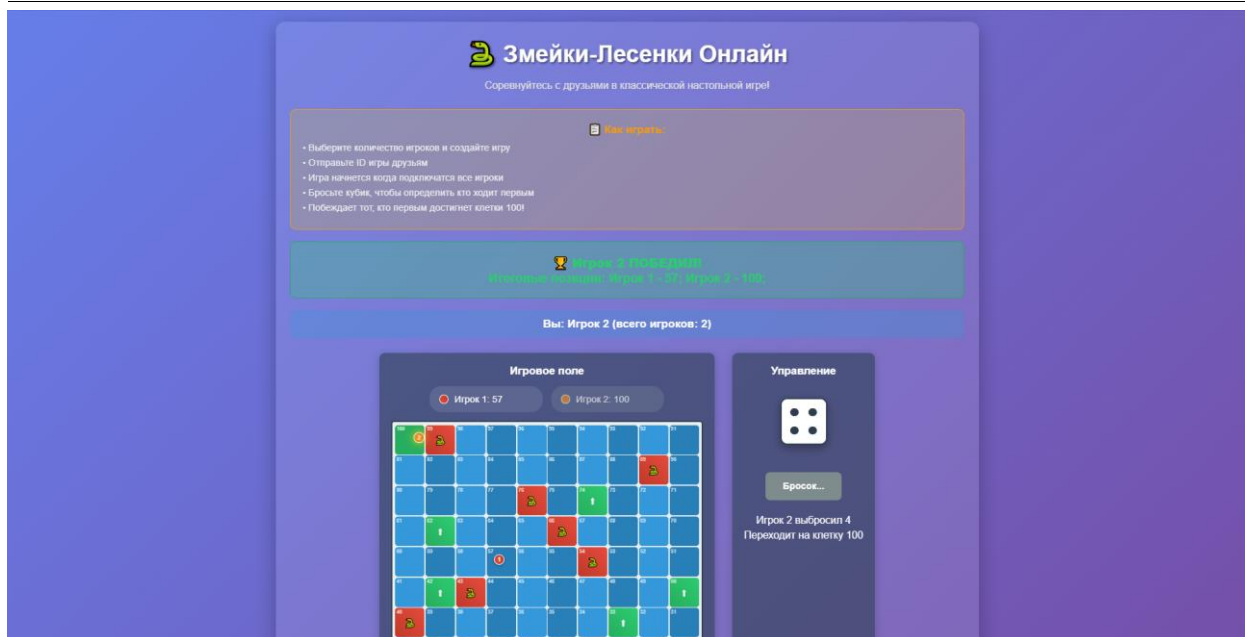


Рисунок 7 – Экран завершения игры с визуальным выделением победителя

#### 4 Выводы

В ходе работы было разработано полноценное клиент-серверное веб-приложение многопользовательской игры «Змейки-Лесенки».

1. Спроектирован интуитивно понятный пользовательский интерфейс, охватывающий полный цикл игры: от создания комнаты с генерацией ID до анимированного экрана победы.
2. Полностью реализована классическая игровая механика, включая сложные граничные условия (отскок от финиша, каскадные перемещения по змеям и лестницам, фаза переброса при ничьей).
3. Применение клиент-серверной архитектуры на базе Node.js и протокола WebSocket выступило надежным техническим фундаментом, обеспечившим мгновенную синхронизацию визуального состояния между всеми участниками с задержкой менее 50 мс и защитой от модификации данных на стороне клиента.

Разработанное приложение готово к использованию как в развлекательных, так и в учебных целях для демонстрации принципов создания современных интерактивных веб-сервисов. Перспективами развития являются добавление системы пользовательских аккаунтов, таблицы лидеров и расширение визуальных тем оформления игрового поля.

#### Библиографический список

1. Мухаметханов И.Р., Хафизов М.Р., Шубин А.В. Сравнение клиент-серверных решений при разработке многопользовательских онлайн-игр на Unity // Электронные библиотеки. 2022. Т. 25. № 5. С. 472–488. URL: <https://elibrary.ru/item.asp?id=53980978>
2. Денисович Ж.Н., Владимирович Б.С. Различия клиент-серверных и веб-приложений. Выбор метода разработки для деловой игры «НТИ 2121:

- профессии будущего» // Дни студенческой науки: Материалы XIV Всероссийского форума молодых ученых и студентов. Москва-Берлин: ООО «Директ-Медиа», 2022. С. 268–272. URL: <https://elibrary.ru/item.asp?id=53985647>
3. Бабков С.В. Разработка веб-приложения для рейтингового учёта игр в спортивную «Мафию» // Цифровой регион: опыт, компетенции, проекты: Сборник статей VI Международной научно-практической конференции, посвященной 25-летию Юбилею инженерно-экономического института, проводимой в рамках Десятилетия науки и технологий в России. Брянск: Брянский государственный инженерно-технологический университет, 2024. С. 64–68. URL: <https://elibrary.ru/item.asp?id=74171680>
4. Снопкова А.М., Данилова В.В., Бутенко Н.С., Локтев Е.С. Особенности выбора архитектуры для браузерных многопользовательских игр // Технические и математические науки. Студенческий научный форум: электронный сборник статей по материалам VII студенческой международной научно-практической конференции. Т. № 7 (7). М.: Международный центр науки и образования, 2018. С. 60–64. URL: <https://elibrary.ru/item.asp?id=35615398>
5. Киселев А.А. Сравнение технологий Websocket и Socket.io // Инновационные аспекты развития науки и техники: сборник статей V Международной научно-практической конференции. Саратов: НОО «Цифровая наука», 2021. С. 44–47. URL: <https://elibrary.ru/item.asp?id=47805455>