МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ДВИЖКА ИГРОВОГО САЙТА ДЛЯ КОМАНДНЫХ ИГР

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Засовина Максима Владимировича

Научный руководитель
к. ф.-м. н., доцент _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2026

ВВЕДЕНИЕ

Актуальность темы. Командные викторины и игровые сессии в дистанционном формате стали регулярной практикой в образовательной и корпоративной среде. Такие форматы позволяют поддерживать вовлеченность, развивать навыки командного взаимодействия и структурировать групповую работу. Вместе с тем в большинстве случаев проведение игры организуется через набор разрозненных сервисов: видеосвязь, чат, электронные таблицы для подсчета баллов, отдельные заметки ведущего. При таком подходе значительная часть внимания уходит на техническое сопровождение процесса, а не на работу с участниками и содержанием заданий.

Для организатора основной сложностью становится необходимость одновременно управлять несколькими контурными задачами: контролировать состояние раунда, отслеживать очередность команд, фиксировать результаты, проверять корректность действий и синхронизировать информацию для всех подключенных пользователей. При отсутствии единой системы возрастают риски операционных ошибок: потери актуального состояния, рассогласования счета между участниками и неочевидных конфликтов в сценарии матча.

Проект, реализованный в рамках выпускной квалификационной работы, направлен на решение указанных проблем за счет разработки веб-движка командной игры с централизованной серверной логикой, ролевым доступом и управляемым жизненным циклом игровой сессии.

Цель бакалаврской работы: разработать движок игрового сайта для проведения онлайн-командных игр с поддержкой ролей пользователей, сессионной логики и устойчивой синхронизации состояния партии.

Для достижения цели поставлены следующие задачи:

1. Формализовать функциональные требования в процессе коммуникации с инициатором проекта в формате «Стартап как диплом».
2. Провести анализ предметной области и рассмотреть существующие платформенные решения.
3. Обосновать архитектурный подход и выбор технологического стека.
4. Спроектировать структуру данных для хранения пользователей, игровых сессий, контентных сущностей и результатов.
5. Реализовать ключевые функциональные модули приложения: регистрацию и авторизацию, создание и ведение партии, управление карточками

вопросов и ролями.

6. Реализовать серверную валидацию действий по роли пользователя и текущей фазе игры.
7. Выполнить развертывание системы на удаленном сервере и проверить работоспособность основных пользовательских маршрутов.

Краткая характеристика материалов исследования. Материалы исследования включают научные и технические источники по геймификации, разработке веб-сервисов и архитектуре программных систем, а также практическую базу проекта: исходный код, модель данных, API-маршруты и результаты проверки после развертывания на VPS.

Объектом исследования являются процессы проведения онлайн-командных викторин в учебных и прикладных форматах. **Предметом исследования** выступают методы и средства программной реализации веб-движка командной игры, включая проектирование модели данных, API, ролевого доступа и серверной логики управления сессией.

Методы и подходы, использованные в работе. В работе применялся итеративный подход: формализация требований, проектирование, реализация, проверка пользовательских сценариев и корректировка. Для анализа использовались сравнительная оценка существующих решений и сценарный подход, при котором каждый пользовательский шаг связывался с конкретным действием системы. Для описания поведения продукта применялись UML-диаграммы.

Практическая значимость. Практическая значимость работы состоит в создании функционирующего MVP веб-движка, применимого для проведения командных онлайн-сессий в учебных и прикладных сценариях. Реализованная архитектурная основа обеспечивает возможность последующего развития продукта без пересборки базовой модели взаимодействия.

Структура и объем ВКР. Выпускная квалификационная работа включает введение, два основных раздела, заключение, список использованных источников и приложения с ключевыми листингами реализации. Первый раздел посвящен анализу предметной области, постановке задачи, оценке существующих решений и обоснованию технологического выбора. Второй раздел содержит описание проектирования и реализации программной части, структуры данных, API и эксплуатационного контура.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Анализ предметной области.

Предметная область разрабатываемой системы связана с проведением командных онлайн-викторин, где участвуют пользователи с разными ролями и требуется управление ходом партии. Базовая проблема заключается в том, что в типовом сценарии ведущий одновременно проводит игру и вручную выполняет технические операции: считает баллы, отслеживает очередность, фиксирует события раунда и синхронизирует состояние между участниками. Такая организация ухудшает управляемость процесса, повышает риск ошибок и снижает вовлеченность команды.

По итогам анализа сформулированы ключевые требования к системе: централизованное состояние игры, роли с явными ограничениями полномочий, серверная валидация игровых действий, поддержка полного жизненного цикла сессии и независимый контентный контур. Разрабатываемое приложение рассматривается как движок, в котором пользователь создает партию, выбирает режим, подключает участников, проводит раунды и завершает игру с фиксацией итогов.

Обзор существующих решений.

Проведенная оценка распространенных сервисов викторин и универсальных форм показала, что они удобны для быстрого запуска тестовых сценариев, однако не закрывают задачу полноформатной управляемой сессии. Ограничения проявляются в недостаточной гибкости правил, слабом контроле серверной логики, зависимости от ограничений внешней платформы и ограниченных возможностях адаптации административного контура под конкретный процесс проведения игры.

Выбор технологий.

В качестве клиентской части выбран стек Next.js, React, TypeScript и Tailwind CSS. Для серверной части выбран Node.js и Express, для хранения данных используется PostgreSQL. Инфраструктурный контур реализован на Docker, Docker Compose и Nginx с развертыванием на VPS. Такой набор дает быстрый цикл разработки MVP, понятное разделение ответственности между слоями и возможность дальнейшего масштабирования.

Архитектура приложения. Система построена по клиент-серверной модели. Frontend обеспечивает пользовательское взаимодействие и визуализацию состояния партии, backend выполняет бизнес-логику и проверки доступа, ба-

за данных отвечает за устойчивое хранение сессионных и исторических данных. Важным элементом архитектуры является проксирующий слой Next.js API между клиентом и Express-сервисом, который централизует работу с cookie, обработкой 401-ответов и обновлением access-токена.

Реализация программных модулей.

Во втором разделе работы реализованы модули авторизации, управления игровыми сессиями, выполнения раундов, контентного администрирования и просмотра результатов. На этапе проектирования определены роли ADMIN, HOST и OBSERVER. Ролевая модель реализована как в интерфейсной логике, так и на уровне backend-валидации, что исключает несанкционированные изменения состояния при ручной отправке запросов.

Клиентская часть включает ключевые страницы: регистрацию и вход, создание сессии, экран активной игры, личный кабинет и административную панель. Пользовательский поток организован как последовательность этапов: вход, создание/подключение к партии, проведение раундов, фиксация итогов. Для ведущего в интерфейсе собрана вся критичная информация о ходе игры, счете и событиях раунда, что снижает необходимость переключения между внешними инструментами.

Реализация игрового цикла и фазовой модели. Важной частью практической реализации стала формализация игровых фаз и переходов между ними. На серверной стороне каждая сессия проходит через последовательность состояний: создание, ожидание участников, активный раунд, фиксация результата, завершение. Для каждого перехода задан набор предусловий. Например, действие изменения счета допустимо только в активной фазе и только для роли, имеющей право управления раундом. Аналогично, завершение сессии возможно только после прохождения заданного числа игровых шагов или явной команды от ведущего.

Такой подход позволил уйти от неявной клиентской логики, когда корректность шага определяется только интерфейсом. В реализованной системе клиент лишь инициирует действие, а окончательное решение о его применении принимает сервер. Это обеспечивает единообразное поведение для всех участников и уменьшает вероятность рассогласования состояния при нестабильном соединении или повторной отправке запроса.

Реализация контентного модуля и административного контура. Отдельный модуль посвящен управлению контентом игры. Категории и карточки вопросов хранятся в отдельных сущностях и редактируются через административный интерфейс. В практической реализации это дает два эффекта: во-первых, игровой контент может развиваться независимо от кода движка; во-вторых, администратор получает возможность быстро формировать тематические наборы без участия разработчика.

В административном контуре реализованы операции управления пользователями и ролями, операции по контенту и доступ к истории сессий. Права на эти операции проверяются сервером для каждой точки входа. Это особенно важно для сценариев, где интерфейс может быть открыт в нескольких вкладках или запросы выполняются вручную: даже в таких случаях критичные изменения будут отклонены без соответствующей роли.

Безопасность и управление доступом. Аутентификация реализована на основе JWT-модели с access- и refresh-токенами. Access-токен используется для доступа к защищенным API-маршрутам, refresh-токен применяется для продления пользовательской сессии. При ответе 401 выполняется попытка обновления access-токена и повтор исходного запроса. На сервере действует обязательная проверка роли пользователя и допустимости действия в текущей фазе партии.

Запросы к базе данных выполняются в параметризованном формате, пароли хранятся в виде хэшей. Для сценариев, связанных с восстановлением доступа, реализован служебный токенный контур, а при смене пароля выполняется ревокация активных refresh-токенов.

Практическая реализация API и обработки запросов. API-слой построен по функциональному принципу и включает контуры авторизации, игровых операций, административных действий и работы с историей. На практике это позволило упростить маршрутизацию запросов и повысить предсказуемость обработки ошибок. Для пользователя и клиента важно, что система возвращает согласованные коды ответа и структурированные сообщения, что упрощает реакцию интерфейса на успешные и ошибочные сценарии.

Дополнительно реализован проксирующий слой на стороне Next.js, который отвечает за передачу токенов, обновление access-сессии и повтор запросов после refresh. Благодаря этому клиентские компоненты остаются проще, так как не содержат дублирующую инфраструктурную логику для каждого запроса.

Проектирование модели данных и API. Информационная модель PostgreSQL включает сущности users, refresh_tokens, password_reset_tokens, games, game_results, question_categories и question_cards. Ключевое решение состоит в хранении актуального состояния партии в поле JSONB таблицы games и в отдельном хранении итогов в game_results. Такой подход позволяет одновременно поддерживать оперативный игровой цикл и историю завершенных матчей.

REST API структурирован по функциональным контурам: авторизация, игровые сессии, контент и администрирование. Серверно-управляемая модель игрового цикла обеспечивает единый источник истины: клиент инициирует действие, сервер проверяет роль, фазу и корректность параметров, после чего обновляет состояние и возвращает актуальный снимок партии.

Обеспечение целостности данных. На уровне базы данных реализованы внешние ключи и индексы для ключевых связей между сущностями. На уровне приложения целостность дополнительно поддерживается правилами бизнес-логики. Это означает, что даже при корректном SQL-запросе операция не будет выполнена, если она нарушает логику текущей фазы партии или ролевой контур доступа. Сочетание этих уровней проверки позволяет уменьшить риск ошибок, связанных с неконсистентным состоянием.

Разделение оперативного состояния (games.state) и итогов (game_results) упростило практические сценарии сопровождения. Для текущей партии система работает с целостным JSONB-снимком, а для отчетности и истории использует отдельный плоский набор итоговых записей. Это снижает нагрузку на выборки административного интерфейса и упрощает построение аналитических представлений в следующих версиях продукта.

Обработка отказных и пограничных сценариев. В ходе реализации отдельно прорабатывались сценарии, которые в реальной эксплуатации возникают чаще всего: повторный клик по действию, истекший токен, попытка роли OBSERVER выполнить управляющую операцию, отправка запроса в неподходящей фазе игры, а также повторное подключение участника в активной сессии. Для каждого из этих случаев реализована предсказуемая реакция системы: отказ с корректным кодом, сохранение консистентного состояния и возможность повторной синхронизации клиента.

Практическая ценность такой проработки заключается в снижении ко-

личества трудно воспроизводимых ошибок на этапе эксплуатации. Даже если пользователь выполняет действие в неподходящий момент, состояние партии не повреждается, а интерфейс получает достаточно информации для корректного отображения причины отказа.

Тестирование и развертывание. После реализации выполнен прогон ключевых маршрутов: регистрация, вход, создание сессии, подключение участников, проведение раундов, завершение партии и просмотр результатов. Дополнительно проверялись негативные сценарии: недопустимые действия роли OBSERVER, невалидный токен, попытка шага вне допустимой фазы. Во всех случаях сервер возвращал корректные коды ошибок и сохранял согласованность состояния.

Проект развернут на VPS в контейнерном контуре Ubuntu, Docker и Nginx. После публикации подтверждена работоспособность ключевых пользовательских сценариев и воспроизводимость окружения между локальным и удаленным запуском.

Результаты эксплуатационной проверки. Эксплуатационная проверка включала последовательные прогоны пользовательских сценариев в удаленном контуре: вход пользователей с разными ролями, создание нескольких игровых сессий, выполнение игровых действий в разных фазах, завершение партий и просмотр истории. Дополнительно оценивалась устойчивость после перезапуска сервисов в контейнерной среде. После рестарта backend и базы данных приложение восстанавливало рабочее состояние без ручной переинициализации, что подтвердило корректность контейнерного контура.

Отдельно проверялась воспроизводимость конфигурации между локальной разработкой и VPS. Использование Docker Compose и переменных окружения позволило сохранить идентичный принцип запуска в обоих вариантах. Это упростило сопровождение проекта и снизило вероятность ошибок, связанных с различиями окружений.

Инженерные результаты второго раздела. По результатам практической части получена рабочая реализация движка с четким разграничением уровней ответственности: интерфейс, API-прокси, серверная логика и хранение данных. Реализованный контур уже покрывает основные пользовательские сценарии и обеспечивает надежный фундамент для дальнейшего развития. В технологическом плане система готова к следующему этапу развития: переходу

от polling к WebSocket-синхронизации и расширению аналитических модулей.

Практическая значимость и результаты. По итогам работы получен рабочий MVP движка командной игры, в котором техническая нагрузка ведущего перенесена в систему. Ведущий не выполняет ручной подсчет и ручную синхронизацию состояния между участниками, а управляет партией через единый интерфейс. Полученный результат подтверждает достижение цели работы и формирует основу для дальнейшего развития: переход к WebSocket-синхронизации, расширение аналитики и развитие модерационного контура.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта был разработан движок игрового сайта для командных игр. Работа выполнена в логике «Стартап как диплом»: акцент делался не только на создании демонстрационного прототипа, но и на формировании технологической основы, которую можно развивать как самостоятельный продукт. В ходе работы решена прикладная задача автоматизации проведения онлайн-сессий, где ведущему требуется одновременно управлять сценарием партии, фиксацией результатов и взаимодействием участников.

Поставленная цель достигнута: разработано и реализовано клиент-серверное веб-приложение с ролевой моделью доступа, управляемым жизненным циклом игровой сессии, механизмами авторизации и контуром администрирования контента.

Практический результат включает:

- реализацию сценариев регистрации, входа и управления сессией;
- поддержку ролей ADMIN, HOST, OBSERVER;
- реализацию игрового цикла с обновлением состояния и подсчетом очков;
- реализацию структуры данных для пользователей, партий, результатов и карточек;
- развертывание системы на удаленном сервере и проверку базовых пользовательских маршрутов.

Разработанное решение закрывает ключевую проблему предметной области: техническая часть проведения командной викторины переносится в систему, что позволяет организатору сосредоточиться на содержательной стороне взаимодействия с участниками.

Полученный MVP пригоден для практического применения и дальнейшего развития. Перспективные направления включают переход к WebSocket-синхронизации, расширение аналитического блока, развитие модерационных механизмов и углубление автоматизированного тестирования.