МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка системы управления платформой AI-Battles»

Студент	А.Д. Козлов (И.О. Фамилия)	(личная подпись)
Руководитель	О.В. Аникина (И.О. Фамилия)	(личная подпись)
Консультанты	М.А. Четаева (И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____» _____ 20___ г.

Тольятти 2019

АННОТАЦИЯ

Тема выпускной квалификационной работы: «Разработка системы управления платформой AI-Battles».

Целью выпускной квалификационной работы является разработка компоненты платформы AI-Battles для организации управления функционалом части сервисов.

Работа состоит из введения, трех глав, заключения, списка литературы и приложения.

Во введении представлена актуальность темы, а также формулировка цели и задач.

В первой главе произведен анализ деятельности, рассматриваются бизнес-процессы, описывается задача на автоматизацию.

Во второй главе представлено проектирование системы с помощью структурного и объектно-ориентированного подходов.

В третьей главе представлена технология разработки и принципы работы системы управления платформой, приведено обоснование целесообразности использования панели управления.

В заключении представлены результаты и выводы о выполненной работе. Результатом выпускной квалификационной работы является система управления платформой AI-Battles. В приложении представлены фрагменты кода.

В работе представлено 2 таблицы, 32 рисунка, список использованной литературы содержит 21 источник. Общий объем выпускной квалификационной работы составляет 51 страницу.

ABSTRACT

The title of the bachelor's thesis is «The control panel for AI-Battles platform». This bachelor's thesis is devoted to developing a system for management of the AI-Battles platform activities.

Structurally the work includes an introduction, three chapters, a conclusion and a list of references and an appendix.

The aim of the work is to develop the AI-Battles platform component for the activity management in core services.

In the first chapter, the analysis of the platform activities is made, business processes are reviewed, and the task of automation is set.

The second chapter describes the design of the control panel with the help of the structural and object-oriented approaches.

The third chapter presents the development technology and principles of the control panel, such as reactivity, a single page approach and state management. Also, the rationale for using the control panel in the AI-Battles platform components is given.

The conclusion presents the results and conclusions on the work performed. The result of the work is the part of AI-Battles platform which automates the platform administrator actions by providing web user interface for interaction with the platform.

The appendix contains snippets of code with detailed comments.

The work presents 2 tables, 32 figures, the list of 21 references. The total volume of the bachelor's thesis is 55 pages.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В 2019 году в Тольяттинском государственном университете, совместно с тольяттинским офисом компании Netcracker Technology была проведена ежегодная городская олимпиада по программированию, в одном из этапов которой студентам предлагалось поучаствовать в соревновании по написанию стратегии для игры в морской бой.

Игровой искусственный интеллект (англ. Game artificial intelligence, сокр. AI) — набор программных методик, которые используются в компьютерных играх для создания иллюзии интеллекта в поведении объектов, управляемых компьютером.

Для проведения соревнования офисом компании Netcracker была разработана и предоставлена платформа «AI Battles», куда участники могли загружать свои программы, реализующие стратегии — ботов, написанных на любом из поддерживаемых платформой распространенных языков программирования. Система выполняла компиляцию и интерпретацию исходного кода, проверяла бота на ошибки и в случае успеха добавляла его в соревнование.

Во время проведения мероприятия организаторам требовался эффективный и удобный способ контролировать процессы в платформе, собирать и анализировать сведения о объектах, воздействовать на поведение системы. Исходя из этого, целесообразно внедрение системы для управления некоторыми бизнес-процессами платформы.

Актуальность выпускной квалификационной работы (ВКР) обусловлена необходимостью разработки системы управления платформой AI-Battles по заказу тольяттинского офиса компании Netcracker.

Объектом исследования является платформа AI игр «AI Battles».

Предметом исследования является автоматизация бизнес-процесса управления объектами платформы.

Целью ВКР является разработка компоненты платформы AI-Battles для организации управления функционалом части сервисов.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить предметную область платформы AI-игр;
- разработать концептуальную модель системы управления;
- разработать логическую модель данных системы управления;
- разработать систему, реализующую пользовательский интерфейс для управления состоянием на сервисах платформы;
- провести тестирование системы;
- обосновать эффективность использования системы.

При написании выпускной квалификационной работы использовались такие методы исследования как: анализ, методы моделирования бизнес-процессов при помощи Case-средств, системный подход. В работе использовались стандарты по моделированию и проектированию программных средств, учебные пособия.

В данной выпускной квалификационной работе рассматриваются вопросы по разработке и реализации системы управления платформой «AI Battles».

Выпускная квалификационная работа состоит из введения, трех глав, заключения, списка использованных источников и приложения.

В первой главе производится анализ бизнес-процессов платформы, ставится задача на автоматизацию.

Во второй главе представлено проектирование системы управления платформой «AI-Battles» с помощью структурного и объектно-ориентированного подходов.

В третьей главе описывается архитектура и технология разработки, а также принципы работы системы по управлению платформой «AI-Battles». Приведено обоснование целесообразности использования системы управления для организации.

В заключении приводятся результаты и выводы о проделанной работе.

Итогом выпускной квалификационной работы является разработанная система управления платформой.

Данная работа выполнялась по заказу тольяттинского офиса компании Netcracker. Результат работы планируется к внедрению к основным компонентам платформы AI-Battles.

Глава 1 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ПЛАТФОРМОЙ AI-BATTLES

1.1 Технико-экономическая характеристика компании

Netcracker Technology — одна из сильнейших компаний в сфере разработки и внедрения программного обеспечения. У предприятия более двух десятилетий богатейшего отраслевого опыта, тысячи специалистов, огромное количество успешных внедрений и высочайший уровень стандартизации в индустрии. Громкие успехи предприятия регулярно отмечаются самыми престижными отраслевыми наградами, а также признанием ведущих аналитиков. В сотрудничестве со своей материнской компанией — японской корпорацией NEC — Netcracker уже несколько лет занимает ведущие позиции на рынке инновационных технологий виртуализации сетей. Офисы компании находятся по всему миру, в том числе в России и странах СНГ.

Тольяттинский офис был открыт в 2008 году, через год начал свою работу учебно-научный центр «ИнфоКом». За время работы УНЦ обучил и выпустил большое количество студентов. Офис регулярно участвует в различных профессиональных мероприятиях и олимпиадах, призёры которых имеют возможность на внеконкурсное поступление в учебный центр компании, обучение в котором ведётся бесплатно. В 2019 году офисом была представлена платформа «AI Battles» для проведения соревнований на основе игрового искусственного интеллекта.

1.2 Концептуальное моделирование предметной области платформы AI-Battles

1.2.1 Моделирование бизнес-процессов предметной области платформы AI-Battles для постановки задачи автоматизированного варианта решения

Для разработки проекта, а именно алгоритмов работы и основных принципов его действия, необходим анализ предметной области. Анализ предметной области – деятельность, направленная на выявление реальных

потребностей заказчика, а также на выяснения смысла высказанных требований. Таким образом, первое, с чего нужно начать разработку системы – это досконально изучить предметную область, в которой предстоит работать.

В данном случае предстоит разработать компонент платформы для управления внутренними сервисами. Задача подразумевает, что доступ к панели управления должны иметь только аутентифицированные пользователи с соответствующими правами. Рассмотрим существующие компоненты платформы:

- Game Service.
- Bot Service.
- Environments.
- SeaBattle.
- Portal Backend.
- Portal Frontend.

Game Service обеспечивает проведение игры, выполнение раундов, Bot Service обеспечивает хранение и обработку ботов для игр. Portal Backend обеспечивает хранение пользователей, транспортировку ботов в ядро платформы, REST API эндпойнты для Portal Frontend.

Environments содержит окружения для компиляции исходного кода разных языков в исполняемые файлы.

Portal Frontend обеспечивает UI для Portal Backend. То, что видит конечный пользователь, желающий участвовать в игре в веб-браузере. С помощью этого веб-интерфейса участник мероприятия проходит авторизацию и аутентификацию, а затем загружает свою стратегию в платформу. В то же время для организатора мероприятия не существует способа быстро и эффективно, без технических знаний, специфичных для платформы AI-Battles просмотреть информацию об участниках и их боях и ботах, поменять настройки платформы, управлять активностями платформы.

Сайт для участников мероприятия представлен на рисунке 1.1.

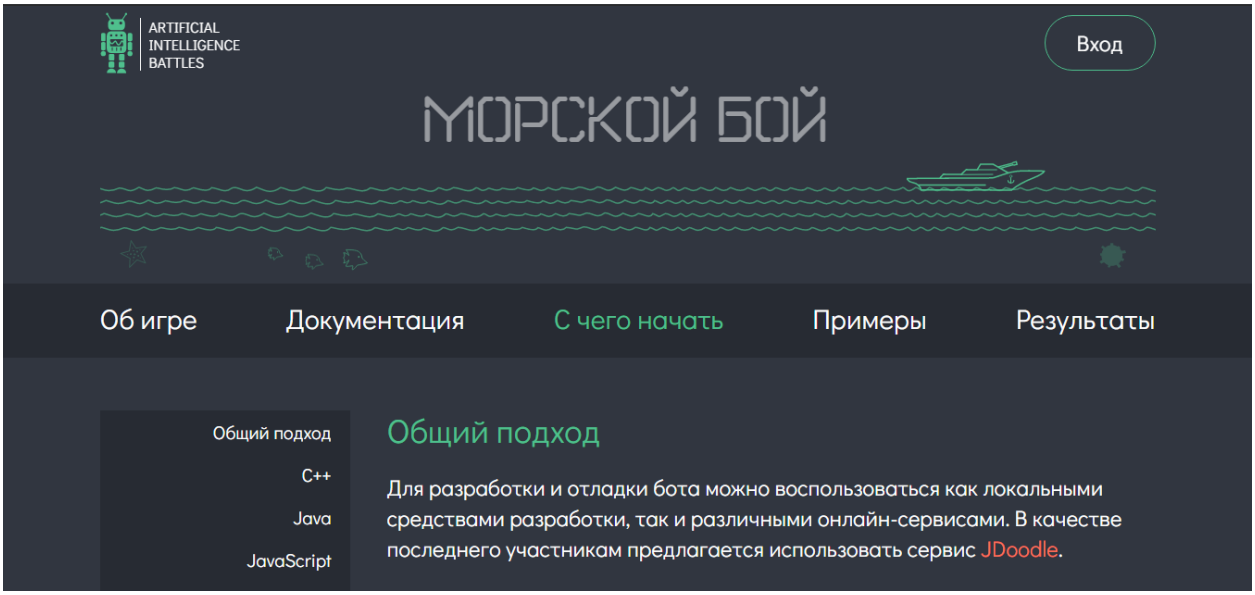

Рисунок 1.1 – Сайт для участников мероприятия

Появляется необходимость в удобном менеджменте процессов, которые происходят в платформе – администрировании.

1.2.2 Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Для анализа работы системы в настоящее время строится функциональная модель «КАК ЕСТЬ». Данная функциональная модель позволяет увидеть, где находятся наиболее слабые места в системе, а также в чем будут состоять преимущества новых бизнес-процессов. Детализация бизнес-процессов позволяет выявить недостатки. Найденные в модели «КАК ЕСТЬ» недостатки должны быть исправлены при создании модели «КАК ДОЛЖНО БЫТЬ» – модели новой организации бизнес-процессов. Модель «КАК ДОЛЖНО БЫТЬ» необходима для оценки последствий внедрения информационной системы и анализа альтернативных путей выполнения работы и документирования того, как система будет функционировать в будущем.

Обследование процессов является обязательным этапом любого проекта создания или развития системы. Построение функциональной модели «КАК ЕСТЬ» позволяет четко зафиксировать какие информационные объекты используются при выполнении функций различного уровня детализации. На основе анализа текущих процессов платформы AI игр была создана следующая

«КАК ЕСТЬ» модель, которая позволяет выделить и систематизировать процессы, протекающие в данной системе при её функционировании (Рисунок 1.2)

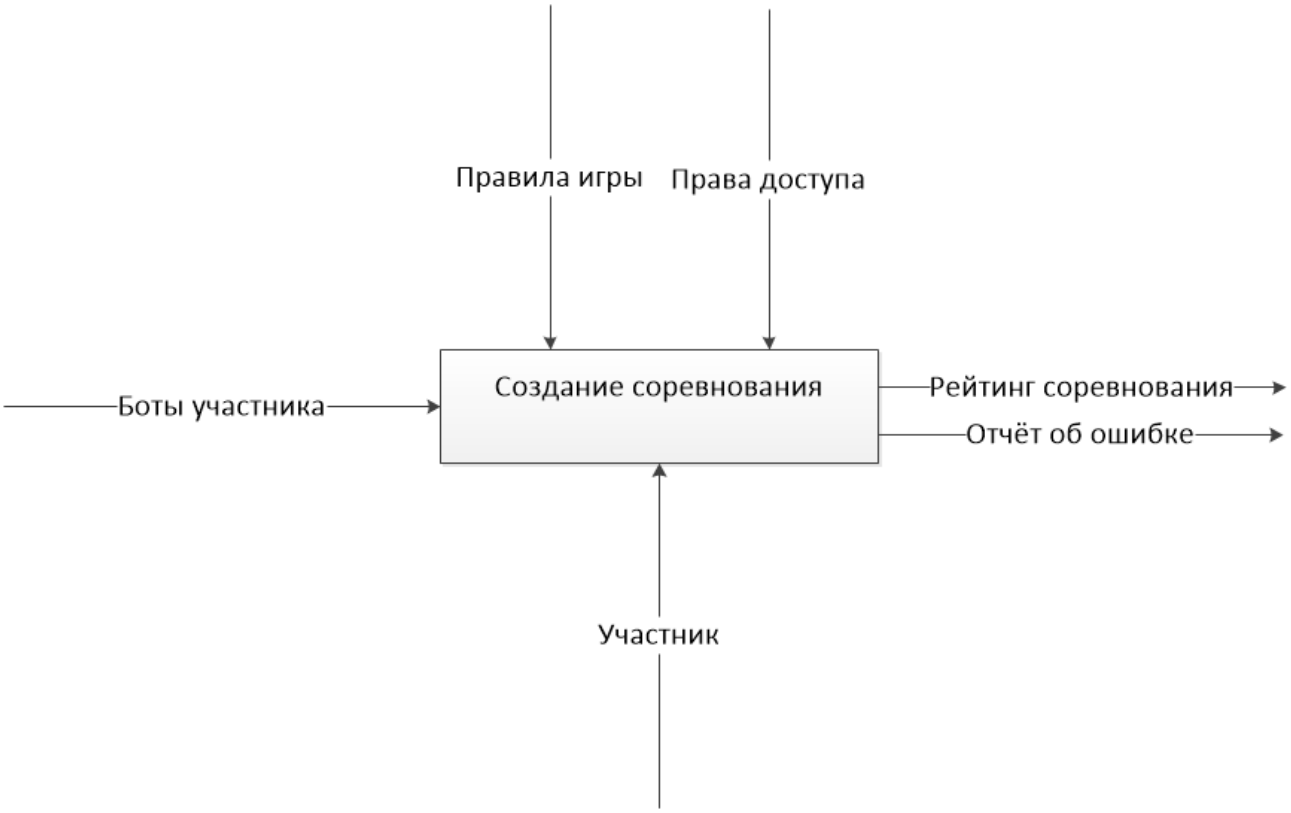

Рисунок 1.2 – Модель «КАК ЕСТЬ»

Для более детального понимания логики бизнес-процессов, протекающих в текущей области разработанная контекстная диаграмма была разбита на четыре следующих процесса:

- добавление бота;
- валидация бота;
- создание сражения;
- расчёт раундов.

Декомпозиция контекстной диаграммы представлена на рисунке 1.3.

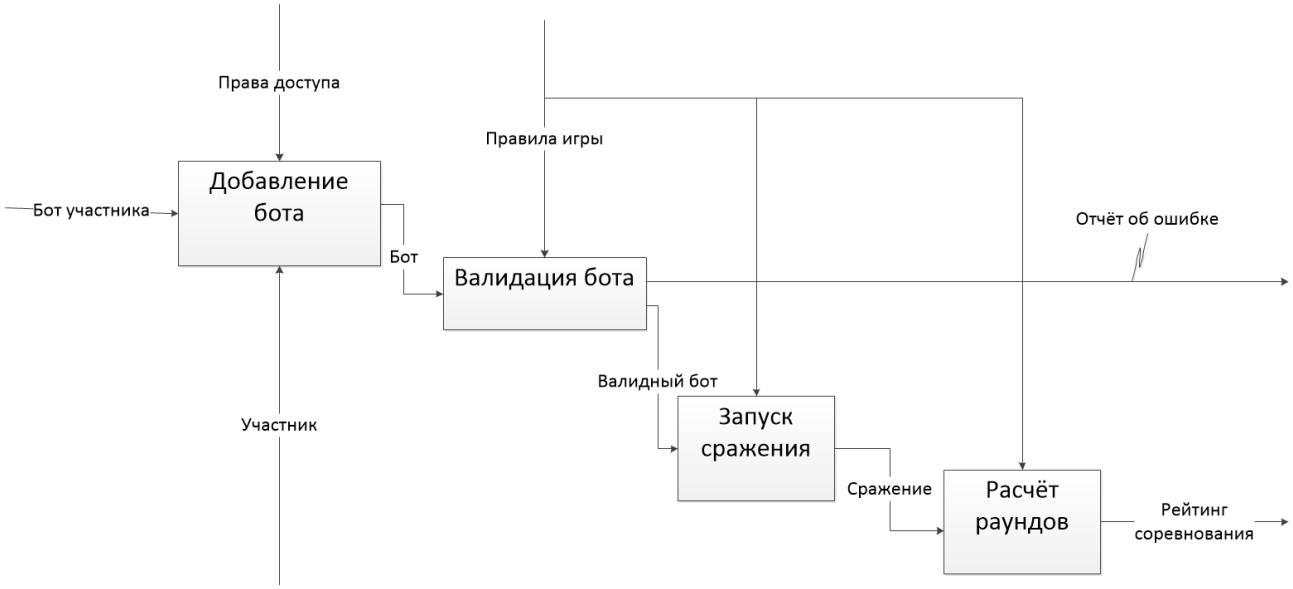

Рисунок 1.3 – Декомпозиция модели «КАК ЕСТЬ»

На рисунке 1.3 показано, что для создания соревнования необходимо, авторизоваться, предоставить ботов, которые пройдут валидацию по правилам игры, если боты валидные с ними начнётся сражение, которое состоит из нескольких раундов, определенных правилами игры, за каждый из которых бот получит очки рейтинга, в зависимости от исхода, по которым и строится финальный рейтинг всего соревнования.

Анализ показал, что в текущей реализации расчёт сражений начинается сразу же после добавления очередного нового валидного бота, исходя из этого, необходимо реализовать дополнительный процесс, запускавший соревнование и соответственно все последующие процессы разбиения ботов на сражения и расчёт раундов в сражениях, по которым формируется рейтинг.

После проведённого анализа процесса «Создание соревнования», сформируем требования к проектируемой системе управления платформой «AI Battles».

1.2.3 Обоснование необходимости автоматизированного варианта решения и формирование требований к новой технологии

Анализ бизнес-процесса «Создание соревнования» показал, что система

обладает недостатками, связанными с отсутствием процессов, регулирующих выполнение бизнес-логики в компонентах платформы.

Также в системе отсутствует удобный и понятный интерфейс для выполнения специфичных для организатора соревнования или лица, исполняющего обязанности обслуживания платформы операций, ниже представлены некоторые из них:

- отсутствие эргономичного, дружественного интерфейса для просмотра и загрузки исходного кода бота участника для анализа его стратегии и чистоты кода;
- отсутствие интерфейса для удаления исходного кода бота участника;
- отсутствие возможности массовой генерации пользователей для случаев запланированных мероприятий с предоставлением участникам заранее подготовленных для них аккаунтов;
- отсутствие возможности запуска соревнования;
- отсутствие интерфейса для смены игры и набора правил;
- отсутствие интерфейса для изменения настроек платформы.

Таким образом, система должна предоставлять логичный, простой и понятный в использовании пользовательский интерфейс с единым стилем и цветовой гаммой, также все поля в формах должны производить проверку введенных данных на корректность.

Система должна разграничивать права доступа пользователей и быть доступной 24 часа в сутки.

Система должна быть удобной в сопровождении и расширении функционала.

После того, как выявлены все недостатки и требования, необходимо произвести постановку задачи на разработку проекта автоматизированной информационной системы. Анализ существующих в открытом доступе разработок, соответствующим сформулированным требованиям не дал результатов по той причине, что платформа разработана и спроектирована с

нуля, а поэтому является уникальной системой, интегрировать в нее типовое существующее решение не представляется возможным, исходя из этого было принято решение о создании собственной системы управления, заточенной под конкретно платформу «AI Battles».

1.3 Постановка задачи на разработку проекта создания/внедрения системы управления платформой AI-Battles

Разрабатываемая система управления платформой AI-игр будет внедрена в основную группу компонентов платформы «AI Battles» для того, чтобы работа организатора соревнований по управлению платформой стала более эффективной и быстрой. Целью создания системы управления является обеспечение простого и удобного менеджмента платформой, а именно система должна реализовывать следующие функции:

- запуска соревнований;
- изменения настроек платформы;
- добавления администраторов;
- просмотра, добавления и массовой генерации пользователей для запланированных мероприятий;
- просмотра, добавления, удаления, поиска и фильтрации ботов по пользователям или играм;
- просмотра информации по сражениям, фильтрация сражений по ботам;
- просмотр и загрузка исходного кода бота участника.

Данный функционал должен значительно сократить количество часов, требуемых на обслуживание системы, а также сократить время обработки и получения оперативных данных для принятия управленческих решений.

Анализ существующих аналогов показал, что подходящие информационные системы не существуют, поэтому для достижения поставленной цели создания системы управления были сформированы основные требования:

- система должна расширить и улучшить качество обслуживания

существующих сервисов;

- диалоговый режим работы с пользователем;

Функциональная структура системы:

- управление ботами;
- управление настройками;
- пользователи;
- сражения;
- авторизация;
- игры.

Требования к качеству:

- надежность;
- практичность;
- сопровождаемость;
- мобильность;
- эффективность.

Распределение функций между человеком и системой:

Со стороны пользователя необходимо ввести необходимую информацию в предлагаемые формы. Со стороны системы необходимо обработать и сохранить информацию согласно структуре.

Все введенные данные должны сперва проверятся на корректность на клиентской стороне.

Требования к безопасности:

- функции административного, процедурного и программно-технического уровня;
- обеспечение целостности;
- функции, реализующие аутентификацию и авторизацию;
- инкапсуляция данных.

Таким образом формализованной постановкой задачи является разработка системы, предоставляющей пользовательский интерфейс для управления состоянием на компонентах платформы.

1.4 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

При создании контекстной диаграммы «КАК ДОЛЖНО БЫТЬ» (TO-BE), изображенной на рисунке 1.4, появляется механизм «Система управления», который отвечает за автоматизацию бизнес-процессов запуска соревнований.

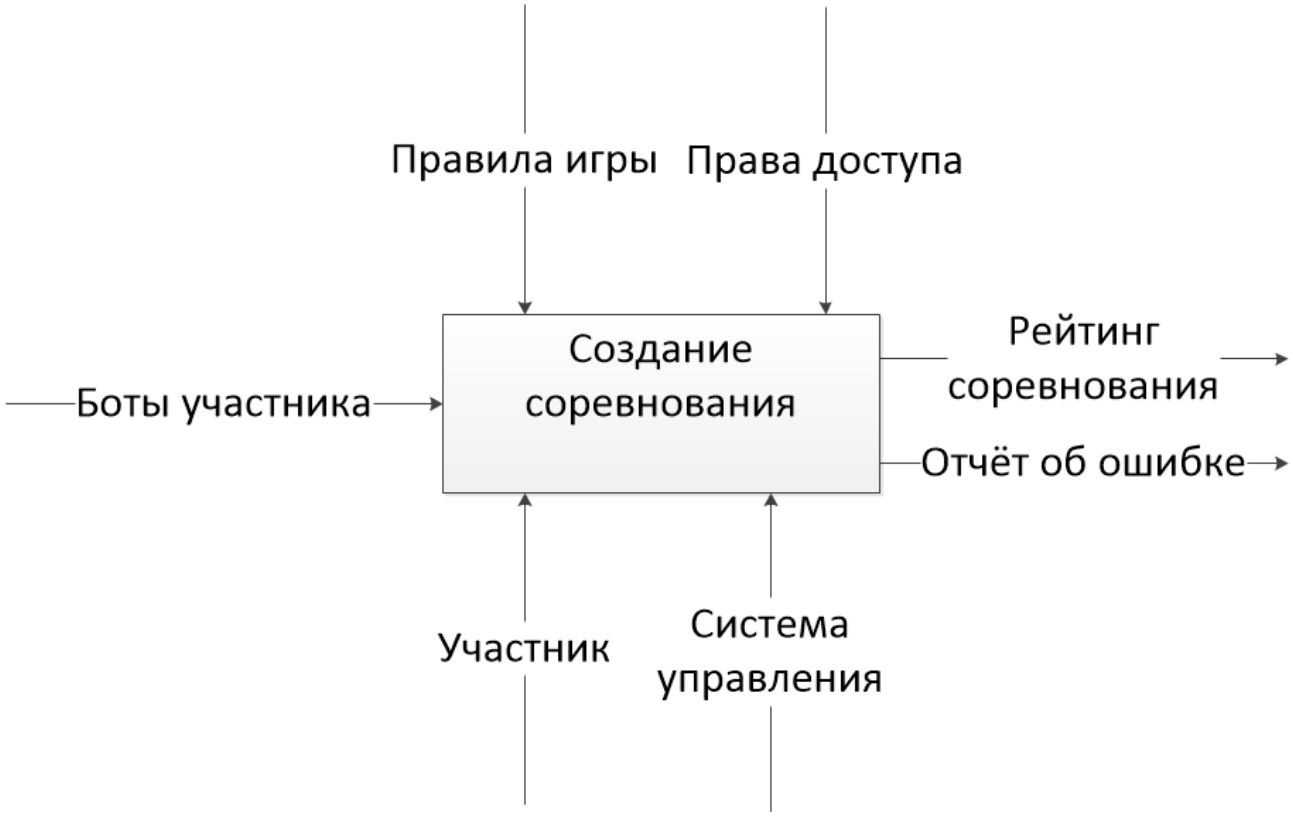

Рисунок 1.4 – Модель «КАК ДОЛЖНО БЫТЬ»

На модели выше появился механизм «Система управления», воздействующий на бизнес-процесс «Создание соревнования» и оказывающий непосредственное влияние на течение процессов в платформе.

Далее следует посмотреть на эти процессы более подробно, для чего проведем декомпозицию этих бизнес-процессов. Результат декомпозиции «Создание соревнования» представлен на рисунке 1.5.

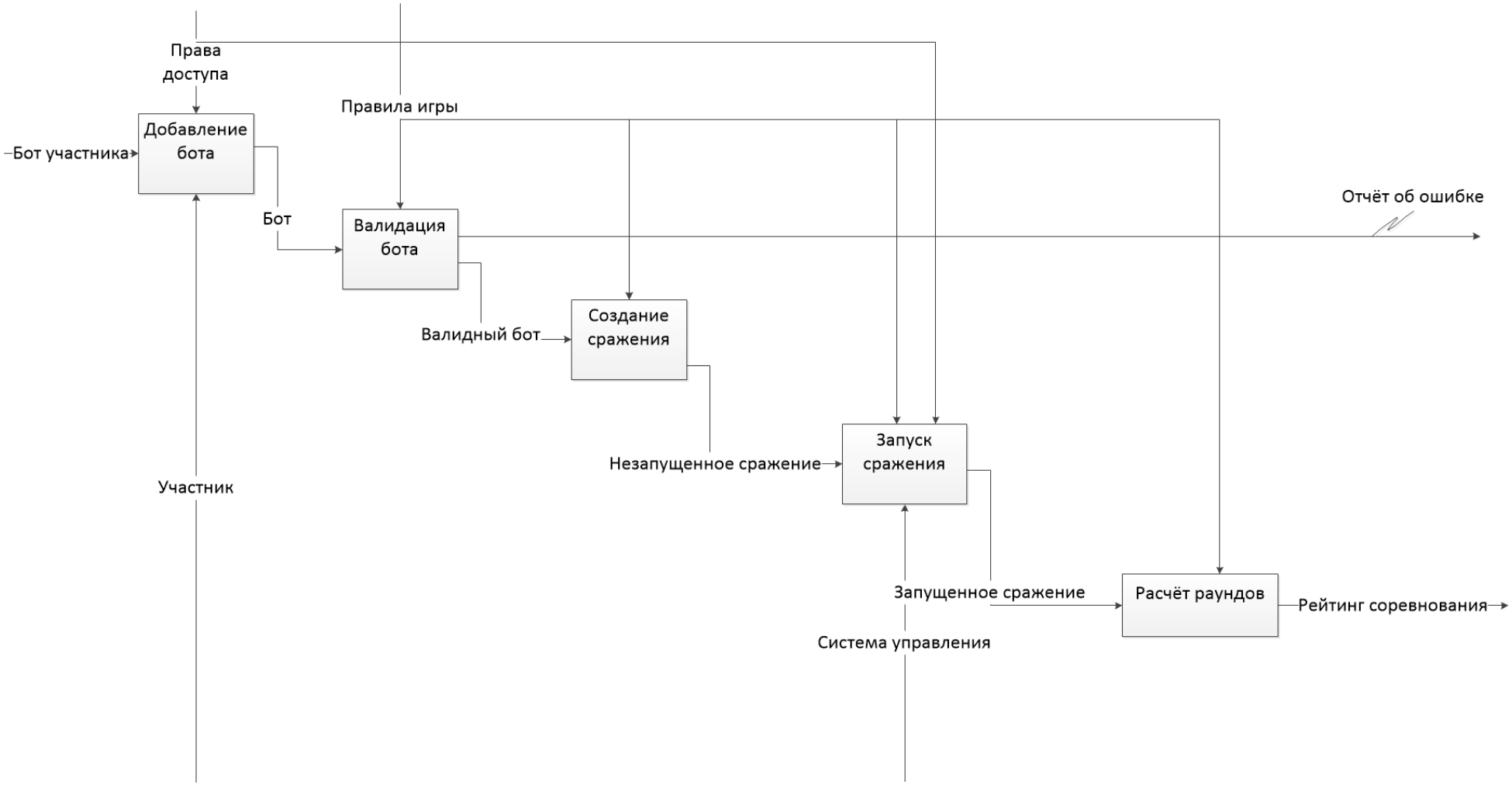

Рисунок 1.5 – Декомпозиция модели «КАК ДОЛЖНО БЫТЬ»

После внедрения механизма «Система управления» запуск расчёта сражений происходит под управлением панели администрирования. Это устраняет недостаток первоначального течения процессов платформы, когда сражения запускались сразу же после добавления ботов, что является нежелательным, неконтролируемым поведением для организаторов мероприятий.

Выводы по главе 1

В первой главе были построены диаграммы основных процессов платформы AI-Battles. Была построена контекстная диаграмма бизнес-процесса вида «КАК ЕСТЬ», а также её декомпозиция. После тщательного анализа существующего течения процессов, происходящих в платформе AI-Battles были определены некоторые улучшения и модификации. Для анализа того, как будут протекать процессы в платформе после улучшения и внедрения системы управления была построена диаграмма вида «КАК ДОЛЖНО БЫТЬ» и её декомпозиция. После анализа была определена цель разработки и поставлены основные задачи, которые послужили основанием для необходимости разработки системы управления платформой AI-Battles.

Глава 2 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ ПЛАТФОРМОЙ AI-BATTLES

2.1 Выбор технологий логического моделирования системы управления платформой AI-Battles

Следующий этап проектирования – это логическое моделирование, представляющее собой проверку функционирования логики на конкретной модели данных. Методология может быть реализована через конкретные технологии и поддерживающие их стандарты, методики.

Одним из таких инструментов, позволяющим обеспечить процесс разработки, спроектировать программу и помочь создать практически готовый к применению продукт, являются CASE-средства.

При применении этой технологии сокращается время разработки и количество ошибок, а также повышается общая эффективность работы. Другими словами, CASE это набор инструментальных средств, которые позволяют моделировать и анализировать предметную область на всех этапах разработки.

Логическая модель системы управления платформой AI-Battles будет описана с помощью диаграммы вариантов использования и диаграммы классов. Логическая и физическая модели системы управления платформой будут построены с помощью языка моделирования UML.

Перейдем к созданию логической модели системы управления.

2.2 Логическая модель системы управления платформой AI-Battles и её описание

При разработке логической модели происходит переход от структурной диаграммы «КАК ДОЛЖНО БЫТЬ» к диаграмме вариантов использования, отражающей функциональный аспект логической модели системы. Также на этом этапе разрабатывается диаграмма классов предметной области, рассматривающая элементный аспект логической модели системы.

Диаграмма вариантов использования представлена на рисунке 2.1.

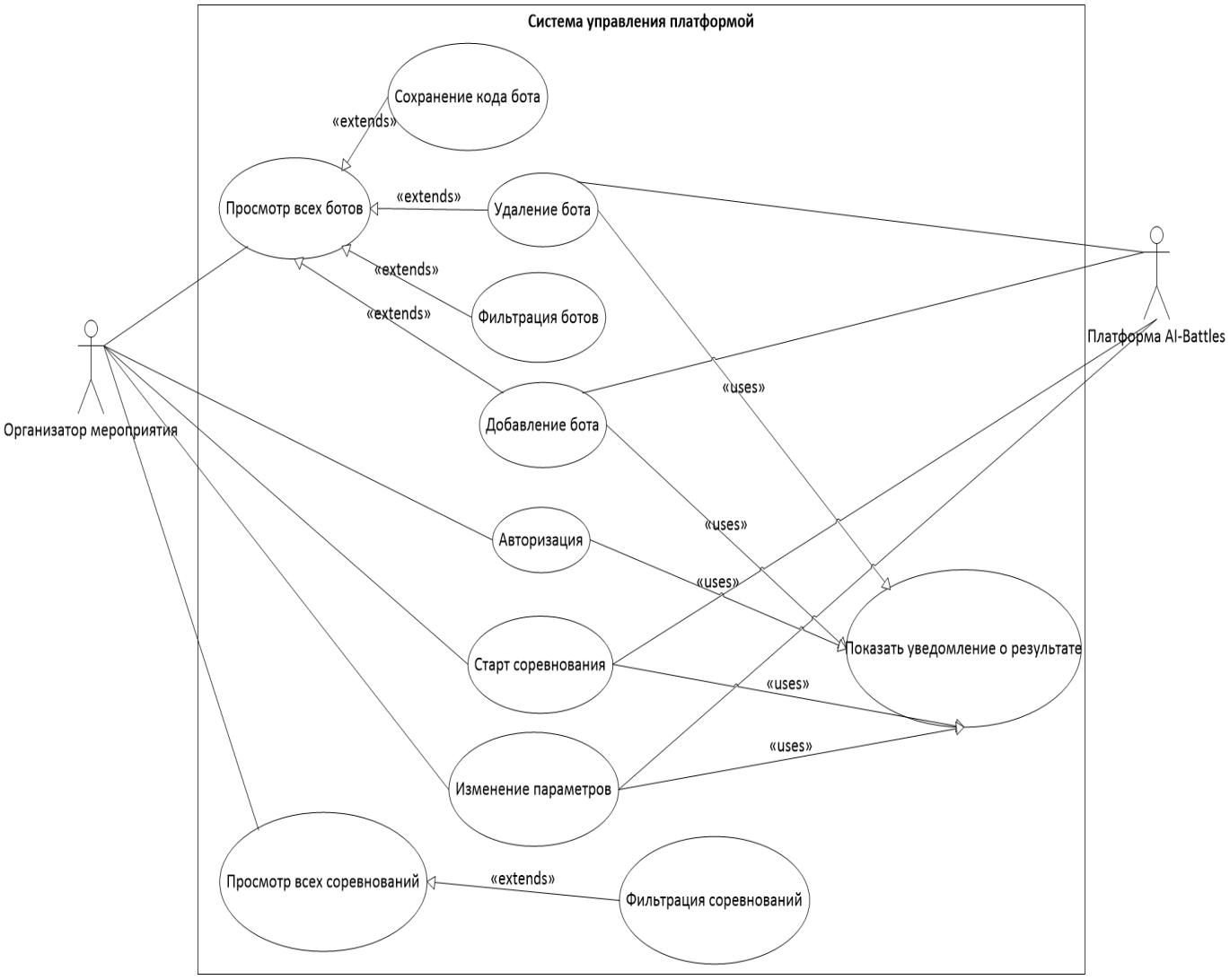

Рисунок 2.1 – Диаграмма вариантов использования системы управления платформой «AI-Battles»

На диаграмме представлены следующие роли: организатор мероприятия - лицо, использующее платформу, которая предоставляет свои функции. Система управления – является посредником между платформой и организатором мероприятия и облегчает использование внутренних возможностей платформы.

Разработанная диаграмма классов показывает атрибуты классов и их взаимосвязь, а также описывает структуру. Данная диаграмма представлена на рисунке 2.2.

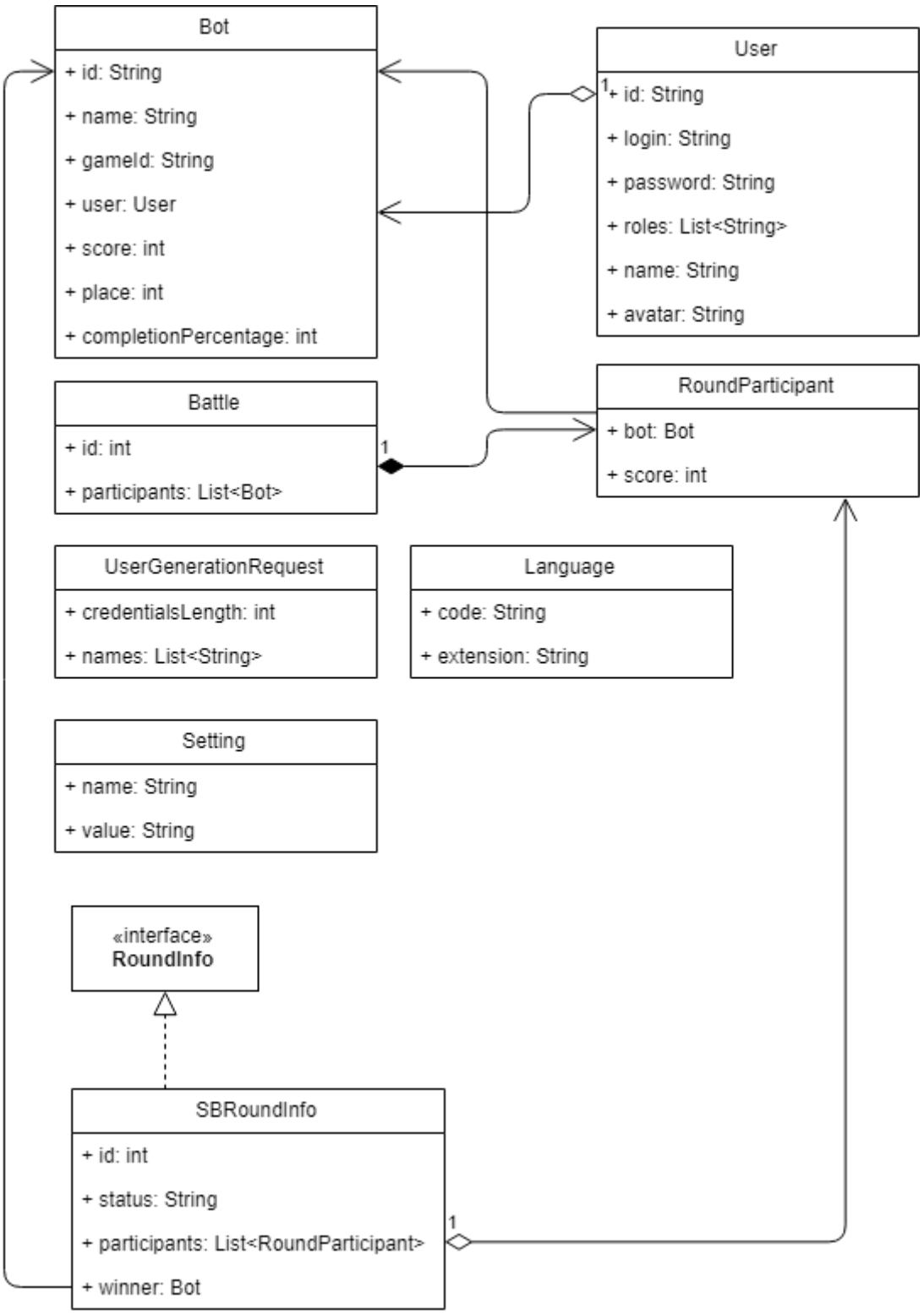

Рисунок 2.2 – Диаграмма классов

В диаграмме были использованы такие связи как: ассоциация, агрегация и композиция. Композиция подразумевает, что объекты зависимого класса не могут существовать без родителя, агрегация подразумевает, что могут.

Ассоциация же подразумевает, что объекты одной сущности связаны с объектами другой сущности так, что можно перемещаться от объектов одного класса к другому.

Так же на рисунке 2.3 изображена диаграмма последовательности процесса генерации пользователей в системе управления платформой «AI-Battles».

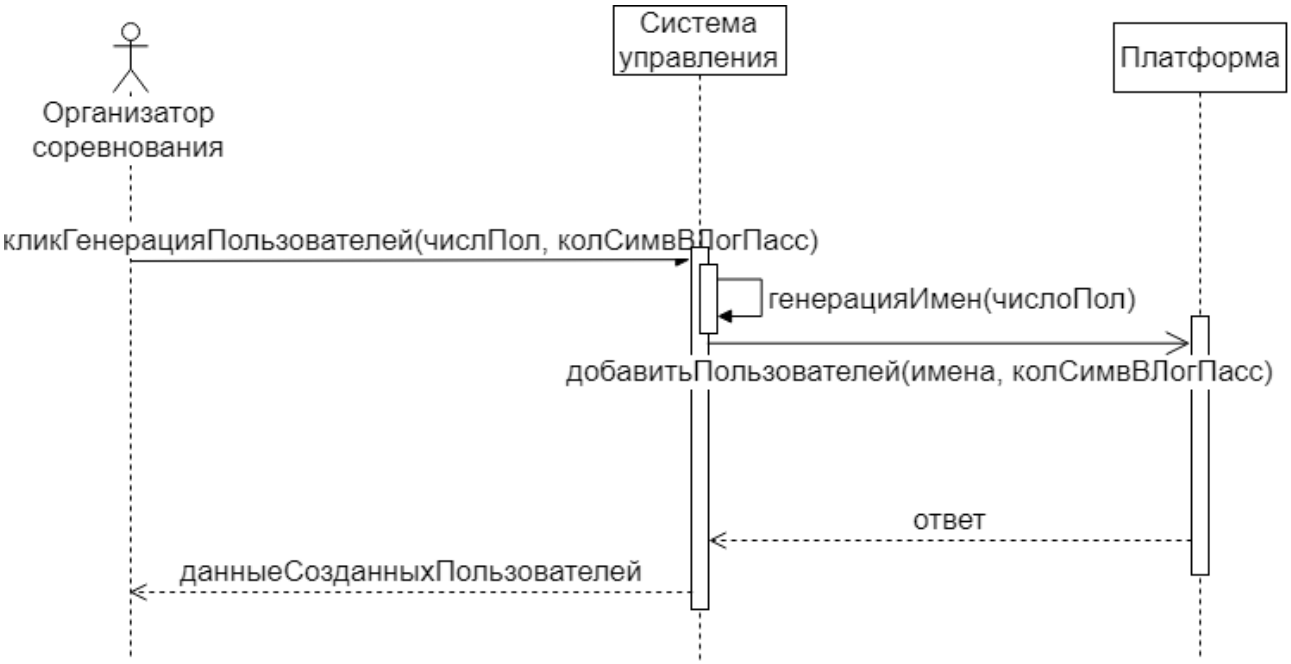

Рисунок 2.3 – Диаграмма последовательности процесса генерации пользователей в системе управления платформой «AI-Battles»

Организатор соревнования заполняет данные о необходимом числе пользователей, количестве символов для логинов и паролей, затем эти данные идут в систему управления, где по количеству пользователей для них генерируются имена по шаблону, затем имена и количество символов для логина и пароля отправляются на платформу, где и генерируются массово пользователи, затем ответ с логинами, паролями, уникальными идентификаторами и именами возвращается через систему управления к организатору соревнования.

Таким образом, с помощью диаграммы последовательности системы

управления платформой, была определена логика функционирования системы и описана последовательность работы.

Далее необходимо построить концептуальную и логическую модель будущей базы данных, используя вышеописанные диаграммы вариантов использования и классов.

2.3 Разработка концептуальной и логической моделей данных системы управления платформой AI-Battles

Концептуальная модель данных описывает основные сущности в проектируемой базе данных и их отношения.

Построение этой модели, изображенной на рисунке 2.4, является важной частью при проектировании системы. Концептуальная модель данных системы управления платформой «AI Battles» имеет шесть сущностей: «Bot», «User», «User Authorities», «Battles», «Participants» и «Settings». После построения концептуальной модели базы данных перейдем к расширенной логической модели, в которой определены атрибуты для сущностей.

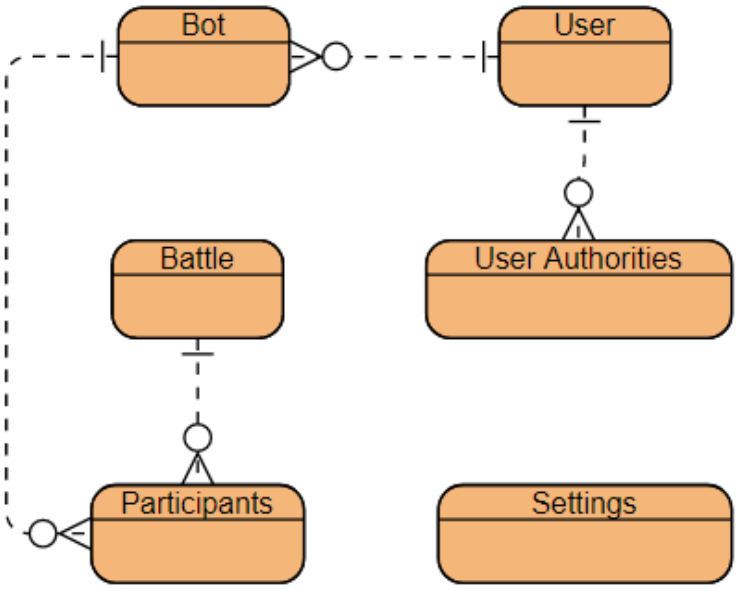

Рисунок 2.4 – Концептуальная модель базы данных системы управления платформой AI-Battles

Логическая модель изображена на рисунке 2.5:

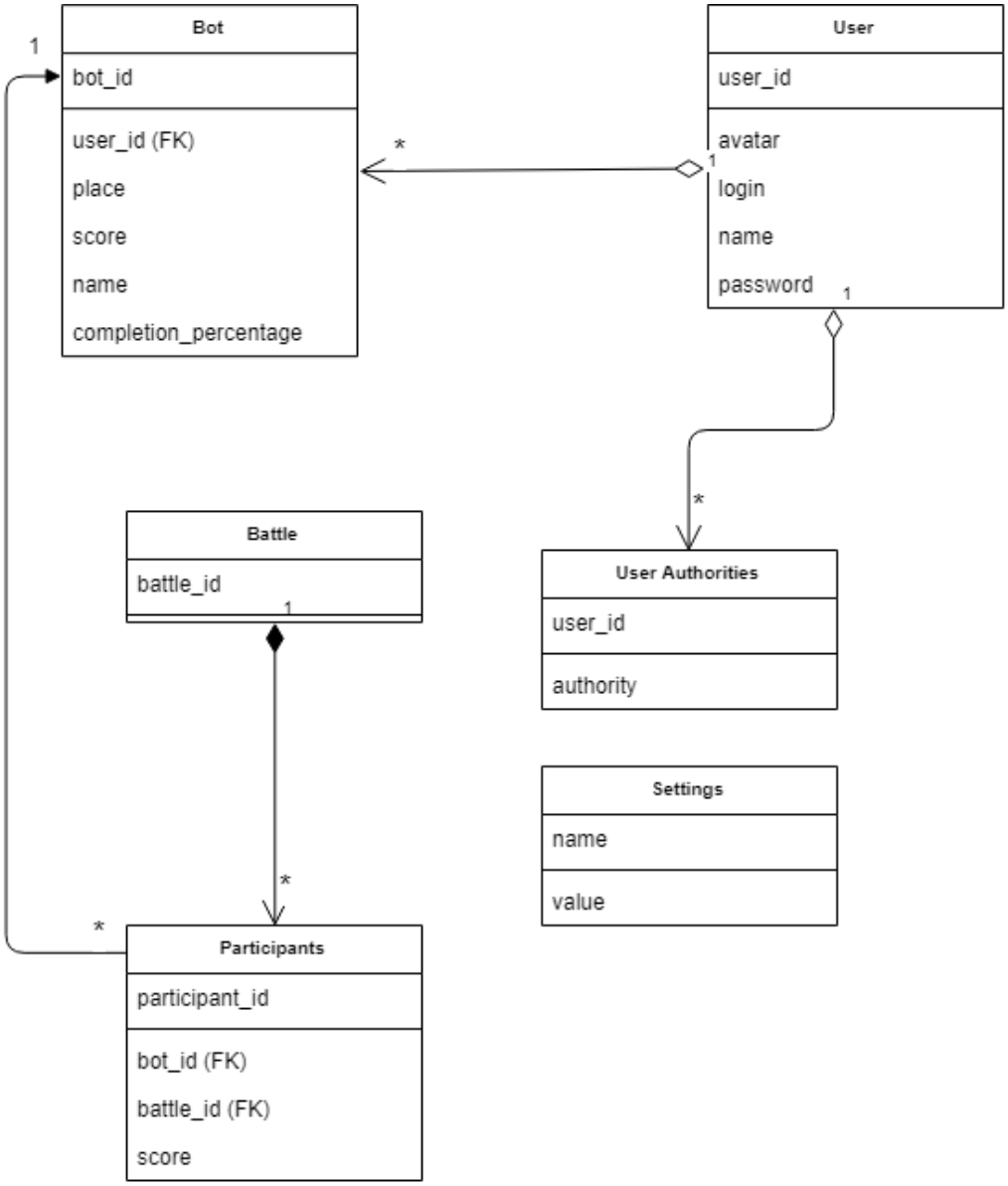

Рисунок 2.5 – Логическая модель базы данных системы управления платформой AI-Battles

Опишем подробно сущности, представленные на модели. Сущность «Bot» представляет собой валидный исходный код участника мероприятия, содержит информацию о своём текущем положении в рейтинге, количество очков, название и процент, характеризующий сколько боев бот сыграл всего. «User» – это зарегистрированный пользователь платформы, имеет имя, логин, пароль и аватар. «Battle» – сущность сражения между ботами, «Participants» – боты, участвующие в сражении, «Settings» – настройки платформы, «User

Authorities» – роли пользователей.

После того, как были описаны сущности и построена логическая модель системы управления платформой «AI-Battles», необходимо описать требования к аппаратно-программному обеспечению системы.

2.4 Требования к аппаратно-программному обеспечению системы управления платформой AI-Battles

Техническое обеспечение системы, а также требования к безопасности и производительности должны эффективно выполняться, чтобы обеспечить качество работы и использовать существующие технические средства.

Аппаратно-программное обеспечение должно соответствовать следующим требованиям:

- операционная система – Linux или Microsoft Windows или macOS;
- оперативная память – 4 GB и выше;
- Docker и Docker Compose;
- подключение к сети интернет.

Требования к безопасности должны быть следующими: требуется защитить исходный код общей части информационной системы. Требуется разграничение доступа. Пароли сотрудников и участников хранятся в зашифрованном виде. На уровне СУБД должно быть реализовано разграничение доступа к данным в БД.

Выводы по главе 2

В главе 2 было произведено и описано логическое проектирование системы управления платформой «AI Battles».

Была выбрана технология логического моделирования, а также произведено проектирование БД и построение логической модели. Были описаны выявленные требования к аппаратно-программному обеспечению системы управления платформой.

Глава 3 ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ПЛАТФОРМОЙ AI-BATTLES

3.1 Выбор архитектуры системы управления платформой AI-Battles

Выбор архитектуры панели управления целиком зависит от архитектурного решения самой платформы, частью которой и будет являться система. Панель управления будет реализовывать трехзвенную «клиент-серверную» архитектуру, собираться в Docker-образ и разворачиваться в Docker-контейнере. Все запросы к платформе проксируются веб-сервером nginx.

Docker-контейнер представляет собой оболочку над приложением и его зависимостями, в котором находится всё необходимое для запуска: среда исполнения, библиотеки, инструменты. Это гарантирует, что приложение, распространяемое в контейнере будет исполняться одинаково на любой операционной системе или устройстве. Это сокращает время подготовки рабочего окружения, избавляет от конфликтов зависимостей, упрощает развертывание системы. Каждый контейнер создается из образа. Контейнеры могут быть созданы, запущены, остановлены, перенесены или удалены. Каждый контейнер изолирован и является безопасной платформой для приложения.

Если контейнеров на сервере несколько, управлять ими вручную становится проблематично. Для этого есть технология Docker Compose. Она существует поверх докера и просто позволяет управлять контейнерами на основе единого конфигурационного файла, в котором описаны контейнеры, их параметры и их взаимосвязи, к примеру контейнер Portal Backend имеет право соединяться с портом 8081 контейнера Game Service.

Диаграмма развертывания платформы представлена на рисунке 3.1:

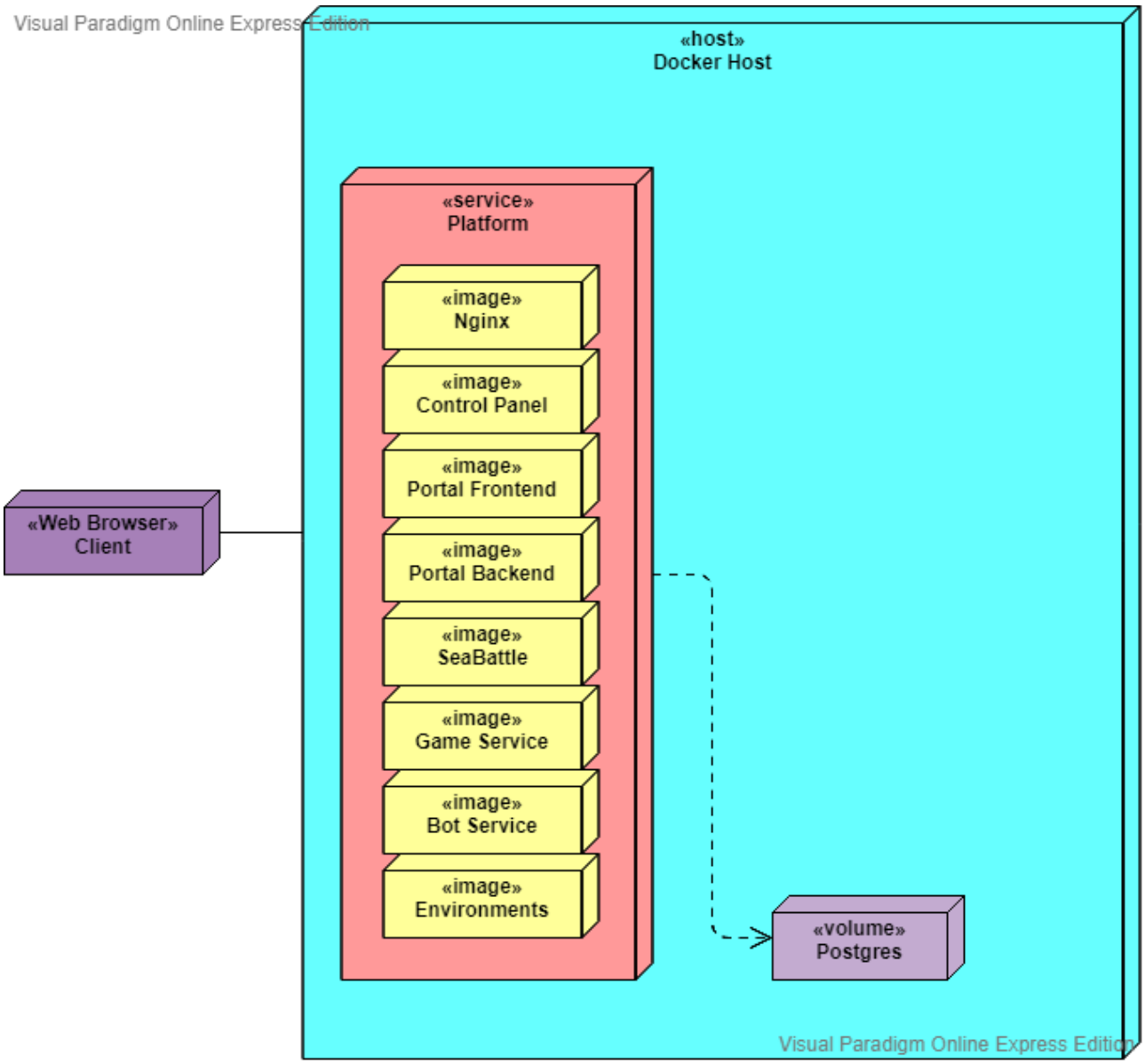

Рисунок 3.1 – Диаграмма развертывания платформы «AI-Battles»

Система управления платформой обозначена Docker образом «Control Panel». Платформа имеет зависимость от образа базы данных Postgres.

3.2 Выбор технологии разработки программного обеспечения

Одним из важнейших этапов при разработке является этап выбора технологии реализации. Для этой задачи был проведен сравнительный анализ прогрессивных JavaScript-фреймворков для разработки приложений. Обязательным требованием была поддержка фреймворками SPA-режима (Single Page Application). В таком режиме страница не перезагружается при

обновлении данных, взаимодействие с пользователем организуется через динамически подгружаемые HTML, CSS, Javascript посредством AJAX. То есть приложение для конечного пользователя максимально походит на нативное, с той разницей, что исполняется в рамках браузера, а не в собственном процессе операционной системы. Это решает проблемы кроссплатформенности и переносимости приложения.

Также приложение должно собираться в бандл, размер которого должен быть как можно меньше, чтобы сократить время первоначальной загрузки страницы пользователем.

Важнейшим критерием является предоставление компонентной структуры фреймворком. Это облегчает сопровождение и понимание кода, способствует уменьшению количества ошибок и возможности переиспользования.

К еще одному фактору относится поддержка реактивности в приложении и средствам управления состоянием.

Таблица 3.1 – Сравнительный анализ прогрессивных JavaScript фреймворков

Требование	Аналог		
	Vue	Angular 2+	React
Быстродействие выполнения	+	+	+
Кривая обучения	+	-	-
Гибкость и модульность	+	-	+
Требование	Аналог		
Размер бандла (минифицированный и сжатый)	+	-	-
Чистота кода	+	+	-
Наличие опыта работы с фреймворком	+	+	-
Итог	6	3	2

По итогам сравнительного анализа прогрессивных JavaScript фреймворков был выбран Vue.js, так как данный фреймворк обладает самым

меньшим размером бандла, имеет дружественную кривую обучения, высокую производительность на базе виртуального DOM и богатую модульную экосистему.

3.3 Разработка программного обеспечения системы управления платформой AI-Battles

3.3.1 Схема взаимосвязи модулей приложения системы управления платформой

Одним из фундаментальных строительных блоков прогрессивного веб-приложения являются компоненты. Они представляют собой переиспользуемые экземпляры со своим именем, входными параметрами, методами, опциями и жизненным циклом. Обычно приложение организуется в виде дерева вложенных компонентов (Рисунок 3.2)

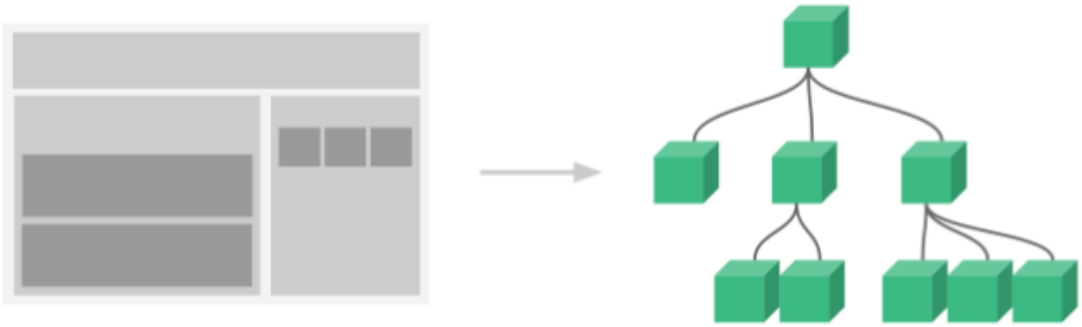

Рисунок 3.2 – Структура вложенных компонентов

У каждого компонента есть секции шаблона, стилей и кода. Данные в приложении протекают в однонаправленном потоке – сверху вниз, от родителя к потомкам, через входные параметры компонентов. Входные параметры — это пользовательские атрибуты, которые можно установить на компоненте. Когда значение передаётся в атрибут входного параметра, оно становится свойством экземпляра компонента и может быть доступно из шаблона.

На рисунке 3.3 изображено простейшее представление концепции однонаправленного потока данных:

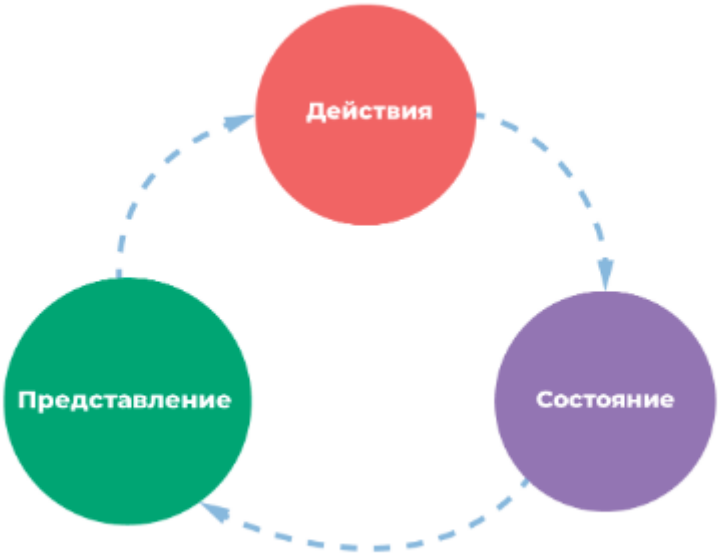

Рисунок 3.3 – Схема представления однонаправленного потока данных

Когда появляется несколько компонентов, основывающихся на одном и том же состоянии возникает проблема передачи одних и тех же входных параметров в глубоко вложенные компоненты, также действия из разных представлений могут оказывать влияние на одни и те же части состояния приложения. Это усложняет сопровождение кода и ведет к ошибкам.

Решением этой проблемы является вынос всего общего состояния приложения из компонентов и управления им в глобальном синглтоне.

Таким образом, состояние компонент хранится централизованно, с распределением на модули, а также правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом. Был применен паттерн управления состоянием и библиотека Vuex. Данная концепция лучше структурирует код и облегчает его поддержку. Также становятся доступными дополнительные возможности по отладке приложения.

Общая схема работы централизованного хранилища представлена на рисунке 3.4:

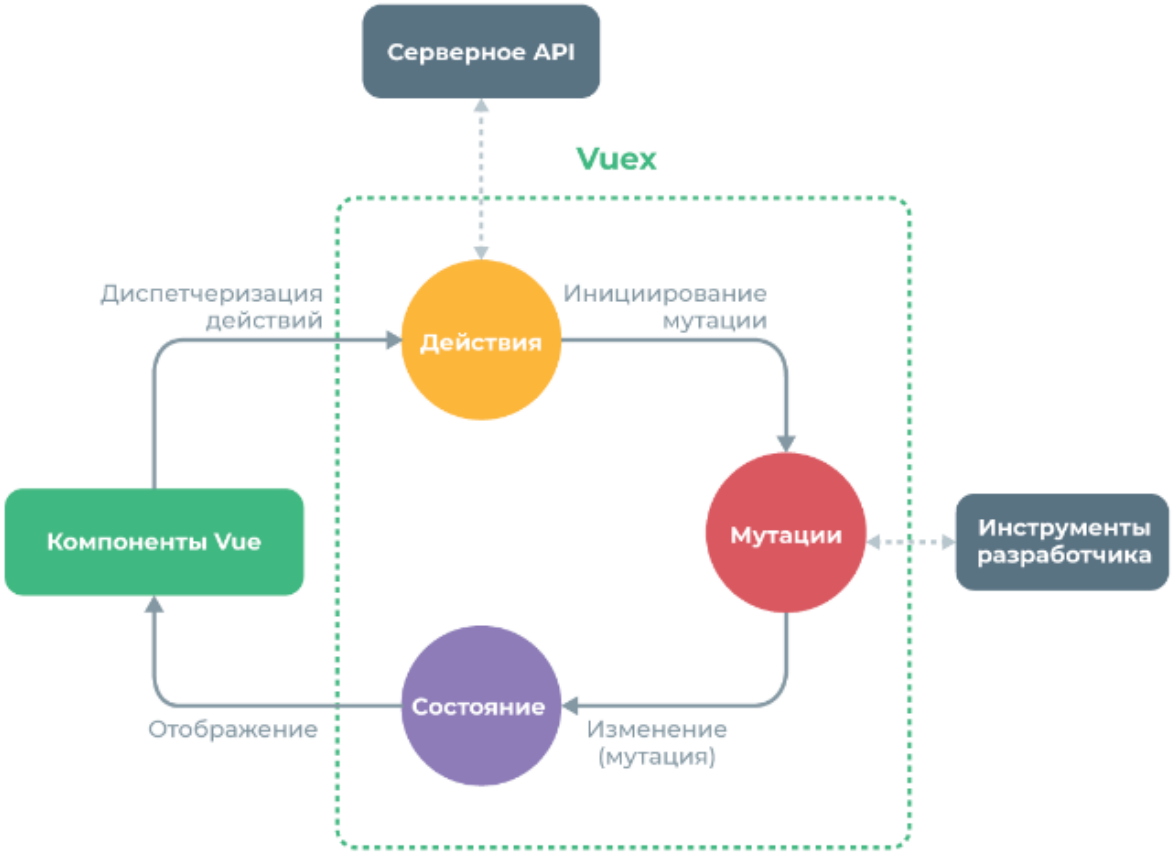

Рисунок 3.4 – Общая схема работы централизованного хранилища

На схеме представлено централизованное хранилище, основными понятиями в котором являются:

«Состояние» – объект, содержащий всё глобальное состояние приложения.

«Мутации» – единственный способ изменить состояние, каждая мутация имеет строковый тип и функцию-обработчик, в которой происходит изменение состояния, вызов функции-обработчика напрямую запрещен, правильный подход - инициирование обработки мутации с указанием её типа.

«Действия» – иниицируют мутации и могут использоваться для инкапсуляции асинхронных операций.

На рисунке 3.5 изображена структура директорий проекта, содержащая компоненты и хранилище:

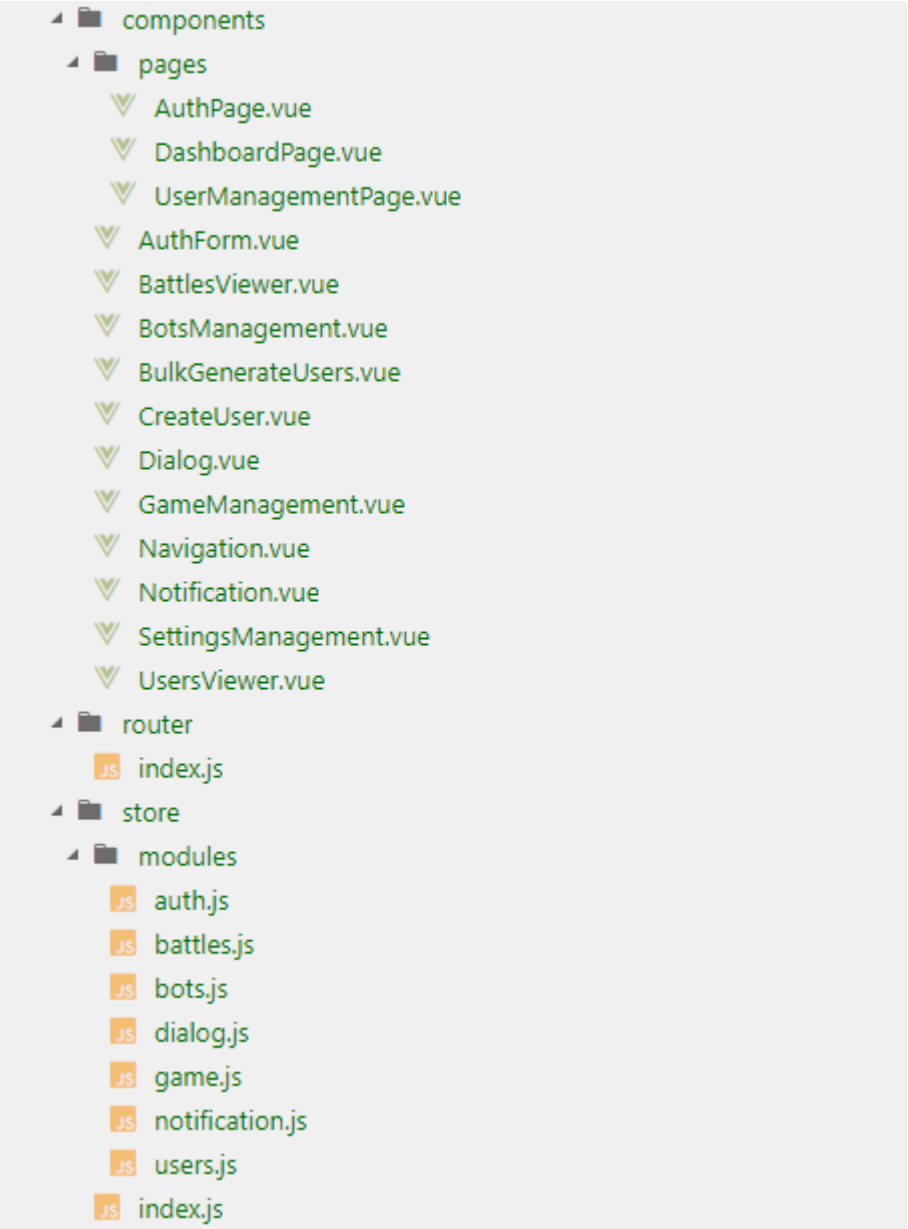

Рисунок 3.5 – Структура директорий проекта, содержащая компоненты и хранилище

Как видно из рисунка, хранилище поделено на функциональные модули, предназначенные для отдельных частей системы. Основная бизнес-логика сосредоточена именно в действиях, определенных в этих модулях. Отдельные компоненты используют состояние и действия, находящиеся в хранилище.

Рассмотрим модуль хранилища users и часть зависимых от него компонентов подробнее. На рисунке 3.6 представлен фрагмент кода действий модуля users:

```
const actions = {
  getUsers: async ({ commit, dispatch }, {search, page}) => {
    commit('request');
    try {
      const usersResponse = await getAllUsers({search, page});
      commit('success');
      commit('setUsers', usersResponse.data);
    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
    }
  },
  doBulkGeneration: async ({ commit, dispatch }, { names, credentialsLength }) => {
    commit('request');
    try {
      const response = await generateUsers({ names, credentialsLength });
      commit('success');
      dispatch('notification/info', { text: names.length + ' user(s) successfully generated' }, { root: true });
      const dialogPayload = {
        content: response.data,
        title: 'Created users: ',
        saveData: response.data
      };
      dispatch('dialog/show', dialogPayload, { root: true });
    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
    }
  },
  doUserCreation: async ({ commit, dispatch }, payload) => {
    commit('request');
    try {
      const response = await createUser(payload);
      const responseData = response.data;
```

Рисунок 3.6 – Фрагмент кода действий модуля «users»

Асинхронное действие получения списка пользователей getUsers в необязательных параметрах принимает строку с именем, по которой может вестись поиск пользователей и страницу, в которой ведется поиск, в случае отсутствия этого параметра поиск будет вестись по всем страницам, сервер отдаёт ответ в виде страниц с данными. При выполнении действия вызывается мутация request, изменяющая поле статуса хранилища на значение, сигнализирующее, что запрос на сервер отправлен, но ответ пока не получен. Далее происходит ожидание ответа от сервера, используется REST API платформы. Методы, отсылающие HTTP запросы на сервер для модуля users представлены на рисунке 3.7. В случае успеха статус меняется мутацией на «success» и состояние мутирует полученными данными. В случае неудачи статус меняется на «error» и запускается действие из другого модуля хранилища – уведомлений, выполняющего показ сообщения пользователю, в

данном случае об ошибке.

```
import client from '@/api/rest/client/api-client';

export function getCurrentUser() {
  return client.get('/user');
}

export async function generateUsers({ credentialsLength, names }) {
  return client.post('/admin/users/generate', { credentialsLength, names })
}

export function getAllUsers({ search, page }) {
  return client.get('/admin/users', {
    params: { search, page }
  });
}

export function createUser({ name, login, password, admin }) {
  return client.post(`/admin/users`, {
    name,
    login,
    password,
    roles: admin ? ['ADMIN'] : null
  });
}


```

Рисунок 3.7 – Код методов для отправки HTTP запросов

На рисунке 3.7 представлен код методов, осуществляющих отправку HTTP запросов по API платформы. Рассмотрим метод «getAllUsers», осуществляющий получение страницы пользователей с возможностью фильтрации по имени. В качестве HTTP метода используется GET, с дополнительными параметрами, указанными в объекте «params», данные параметры передадутся в виде набора данных в URL пользовательского агента, в соответствии со спецификацией HTTP 2.0 – RFC 1866.

Код HTTP клиента, который используют методы с рисунка выше представлен на рисунке 3.8:

```
1  import axios from 'axios';
2  import store from '@/store';
3  import router from '@/router';
4
5  const client = axios.create({
6      baseURL: `/api`,
7      xsrfCookieName: 'XSRF-TOKEN',
8      xsrfHeaderName: 'X-XSRF-TOKEN',
9      headers: {'X-XSRF-TOKEN': 'XSRF-TOKEN'}
10 });
11
12
13 client.interceptors.response.use(null, error =>  {
14     if (error.response.status === 401 || error.response.status === 403) {
15         store.dispatch('auth/logout');
16         router.push('/');
17     }
18     return Promise.reject(error);
19 });
20
21 export default client;
22
```

Рисунок 3.8 – Код HTTP клиента

В качестве HTTP клиента использовалось свободное решение «axios», основанное на промисах. Для использования «axios» необходимо создать его инстанс на основе заданной конфигурации, после этого полученный клиент можно использовать из любой части приложения для отправки и получения данных, а также для некоторых дополнительных возможностей, таких как перехват и преобразования данных. Перехват определен на строке 13 и определяет следующую логику: если статус ответа от сервера соответствует коду HTTP ошибки 401 или 403, в общем случае обозначающие отсутствие или нехватку прав доступа, то произвойдет вызов действия из централизованного хранилища для сброса текущей сессии и редиректа на страницу аутентификациии.

На рисунке 3.9 представлен фрагмент шаблона компонента просмотра пользователей, использущий хранилище с состоянием:

```
1  <template>
2    <v-container>
3      <v-layout justify-center>
4        <v-flex xs12 sm10 md8 lg6>
5          <v-card>
6            <v-toolbar color="white" card>
7              <v-toolbar-title>
8                <v-text-field
9                  hint="Search by name"
10                 persistent-hint
11                 prepend-icon="search"
12                 ref="search"
13                 v-model="search"
14                 @input="throttledSearch"
15                 :loading="isLoading"
16                 maxlength="30"
17                 counter
18               />
19             </v-toolbar-title>
20             <v-layout align-center justify-end>
21               <v-btn :disabled="pageNumber === 0" icon @click="prevPage">
22                 <v-icon>arrow_back</v-icon>
23               </v-btn>
24               <span v-if="users.length" class="caption">{{`${pageNumber}/${totalPages - 1}`}}</span>
25               <v-btn :disabled="maxPageReached || !users.length" icon @click="nextPage">
26                 <v-icon>arrow_forward</v-icon>
27               </v-btn>
28             </v-layout>
29           </v-toolbar>
30           <v-divider class="mt-3"></v-divider>
31
32           <virtual-list :size="size" :remain="remain" :start="start" :offset="offset">
33             <v-card-text v-if="!users.length && !isLoading">
34               <div class="text-xs-center">
35                 <p>Users not found</p>
36               </div>
37             </v-card-text>
38             <v-list v-if="!isLoading">
39               <template v-for="user in users">
40                 <div :key="user.id">
41                   <v-list-tile>
42                     <v-list-tile-content>
43                       <v-list-tile-title v-html="user.name"></v-list-tile-title>
44                       <v-list-tile-sub-title v-html="user.id"></v-list-tile-sub-title>
45                     </v-list-tile-content>
46                   </v-list-tile>
47                   <v-divider class="mt-2 mb-2"/>
48                 </div>
49               </template>
50             </v-list>
51             <v-container v-else>
```

Рисунок 3.9 – Фрагмент шаблона компонента просмотра пользователей

Данный шабон содержит вложенные компоненты, в том числе виртуальный скролл на строке 32, обеспечивающий порциональное отображение данных эмулируя нативную для бразуера прокрутку в целях обеспечения высокой производительности.

3.4 Описание функциональности системы управления платформой AI-Battles

Данная система предлагает организаторам мероприятий или обслуживающим платформу лицам авторизоваться в платформе. После этого появляется возможность просматривать и фильтровать сражения, просматривать, фильтровать, добавлять, удалять ботов, а также выгружать их исходный код, просматривать и фильтровать пользователей, добавлять пользователя с выбором роли, массово генерировать пользователей, менять настройки платформы, запускать обсчёт игр, назначать соревнования для ботов.

Рассмотрим некоторые функции системы от лица пользователя. При запуске системы появляется меню с возможностью пройти аутентификацию и авторизацию в системе. Изображено на рисунке 3.10:

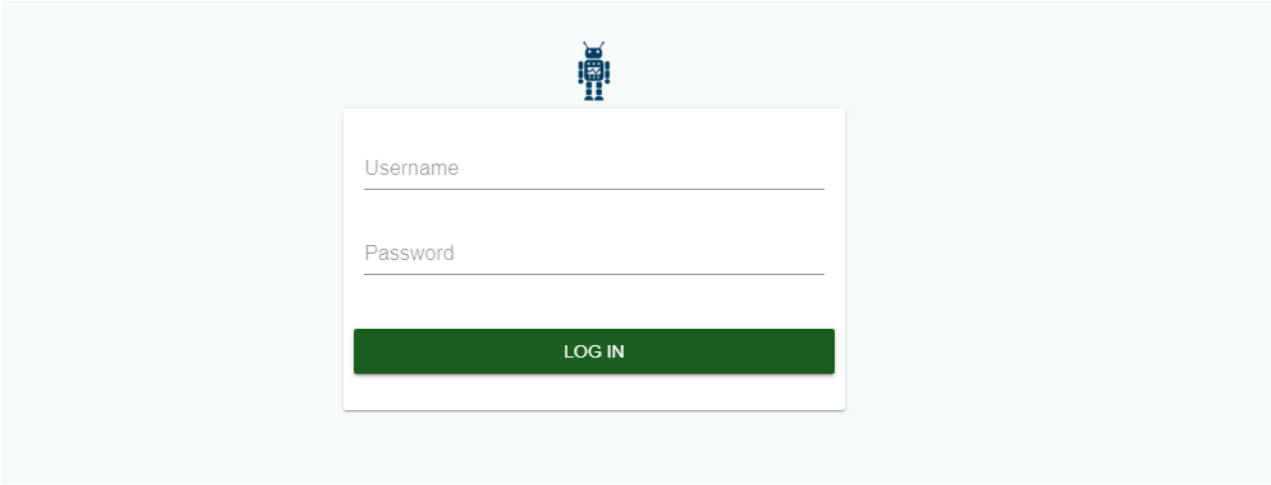

Рисунок 3.10 – Страница аутентификации

После того, как пользователь аутентифицируется в системе с необходимыми правами, он попадает на главную страницу панели управления, на которой находится выдвижное меню и компонент запуска и переключения игр.

При клике на кнопку запустится расчёт сражений для выбранной игры и будет показано уведомление. Изображено на рисунке 3.11:

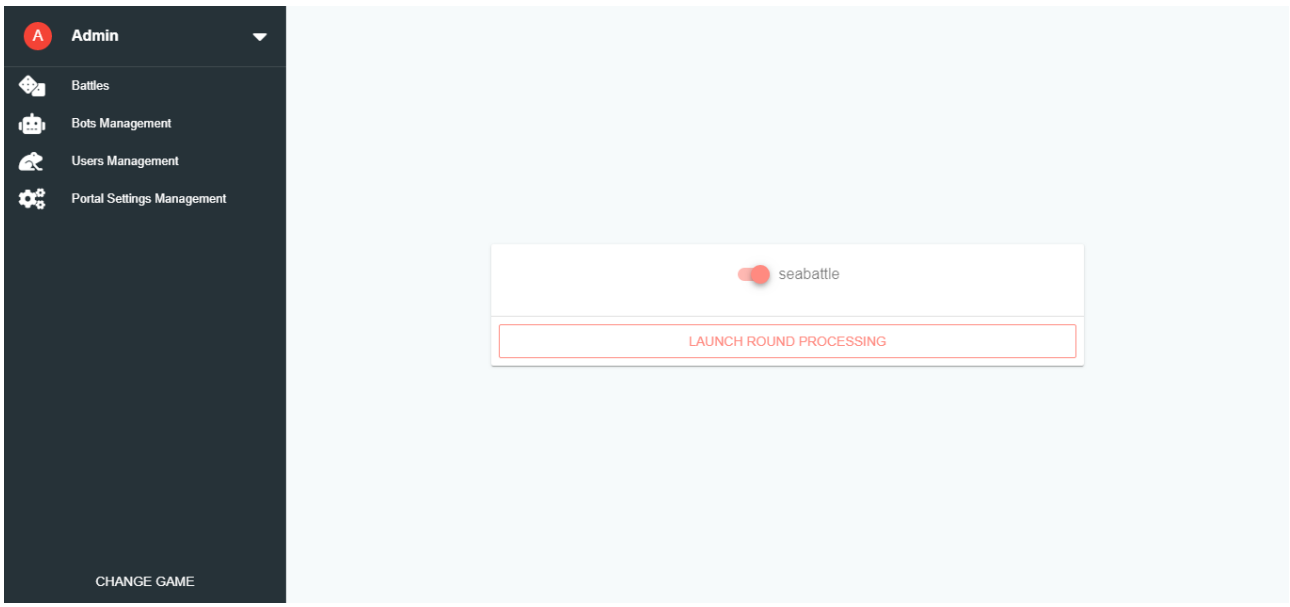

Рисунок 3.11 – Главная страница панели управления

При клике по пункту меню «Battles» пользователь навигируется на соответствующую страницу. Работа приложения на любой странице происходит без перезагрузки сайта. На компоненте «Battles» пользователь может просматривать карточки с информацией о сражениях по текущей игре, фильтровать карточки, при клике на карту пользователь навигируется на страницу управления ботами с выбранным ботом из карточки. Страница «Battles» представлена на рисунке 3.12:

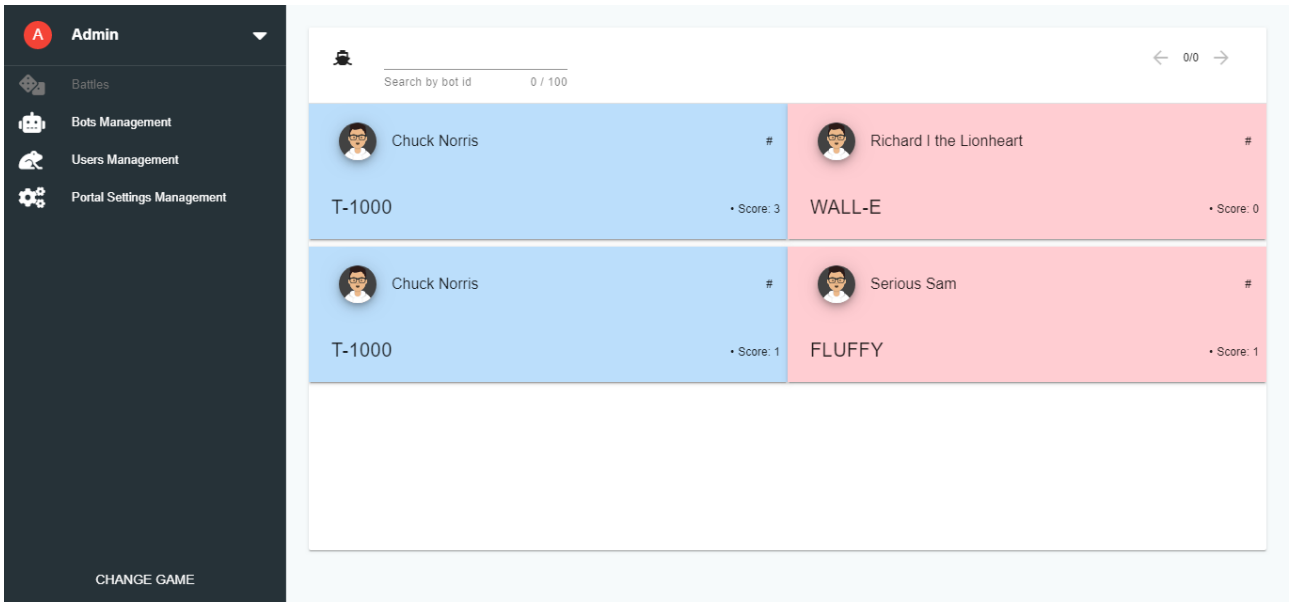

Рисунок 3.12 – Страница «Battles»

При клике по пункту меню «Bots Management» пользователь перейдет к компоненту управления ботами пользователей. Их можно фильтровать по пользователю, сражению или игре. На каждой карточке отображается информация о боте, а также кнопки для действий. При клике на «крест» бот удалится из платформы и будет показано соответствующее уведомление, при клике на иконку сохранения будет показан диалог, в теле которого будет предпросмотр исходного кода, при клике на «Save» в диалоге исходный код сохранится на устройство пользователя, а диалог закроется. При клике на иконку просмотра пользователь перейдет на страницу «Battles», где увидит список сражений в которых участвовал или участвует прямо сейчас этот бот. Страница изображена на рисунке 3.13:

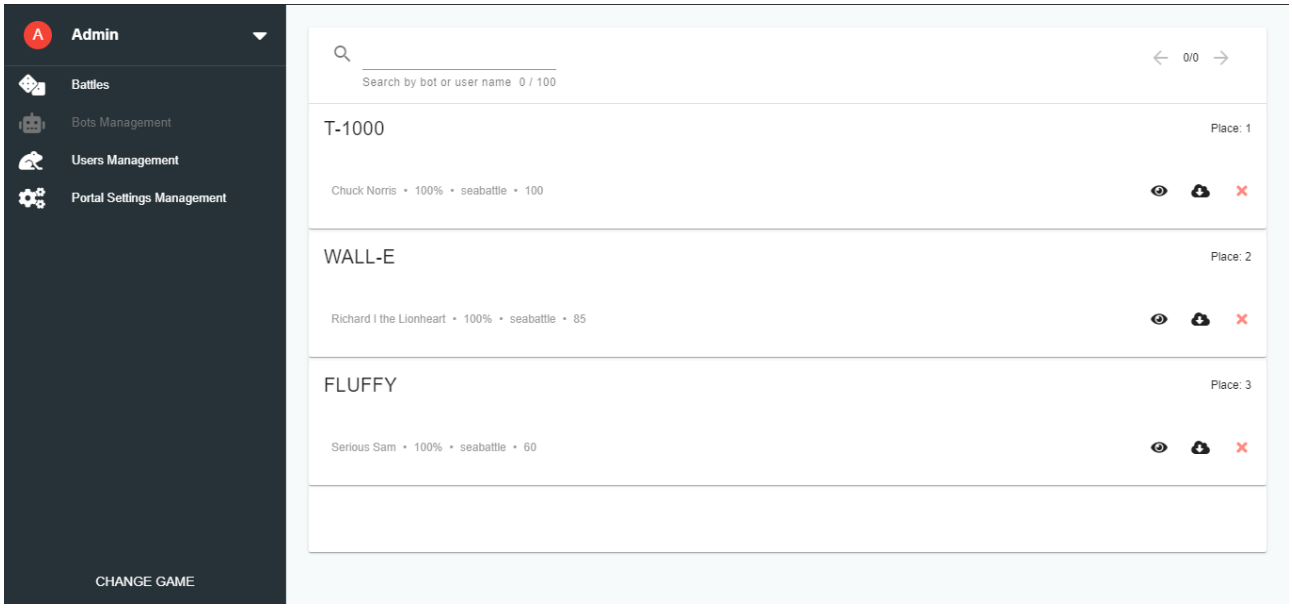

Рисунок 3.13 – Страница «Bots Management»

При клике по пункту меню «Users Management» пользователь навигируется на страницу управления зарегистрированных в системе юзеров. Данная страница состоит из трёх подстраниц:

- All Users.
- User Creation.
- Bulk User Generation.

На подстранице All Users пользователю доступен просмотр всех

зарегистрированных пользователей с фильтрацией по именам и пагинацией, аналогичная пагинация также присутствует и на описанных выше страницах. Данное решение позволяет не загружать большой массив данных сразу, а подгружать его порциями. Подстраница изображена на рисунке 3.14:

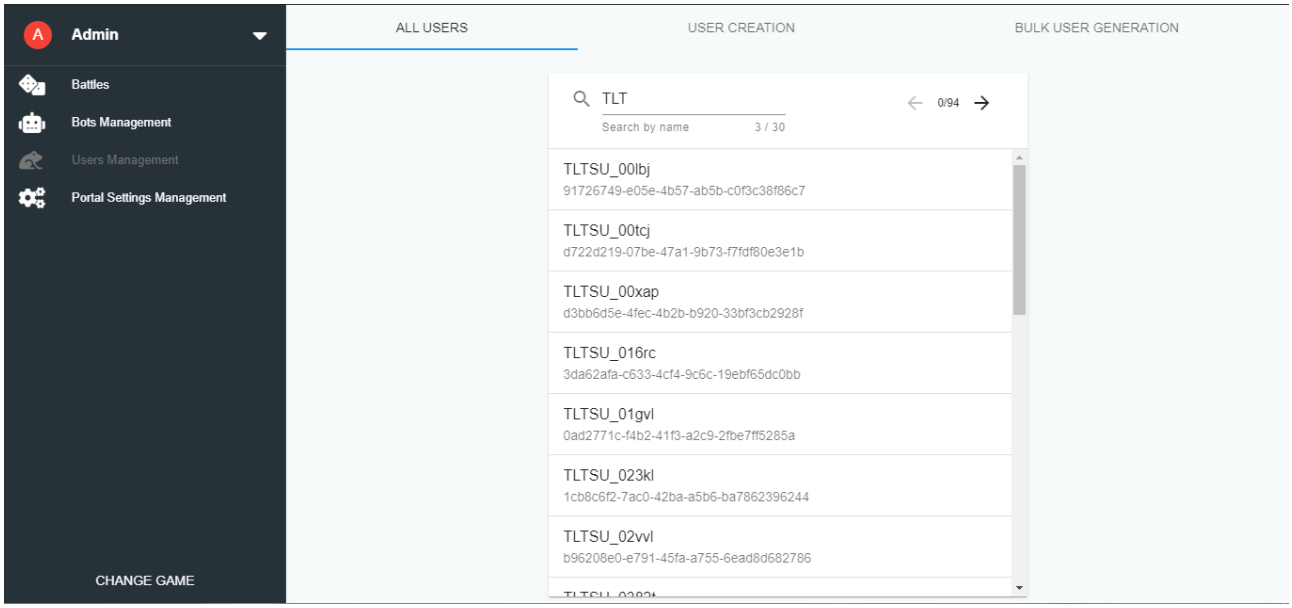

Рисунок 3.14 – Подстраница «All Users»

На следующей подстранице «User Creation» пользователю доступна форма регистрации нового юзера, на данной форме присутствует клиентская валидация, в случае неверного ввода пользователь увидит красные подсказки у невалидных полей. В случае успешной регистрации система покажет уведомление. Форма представлена на рисунке 3.15:

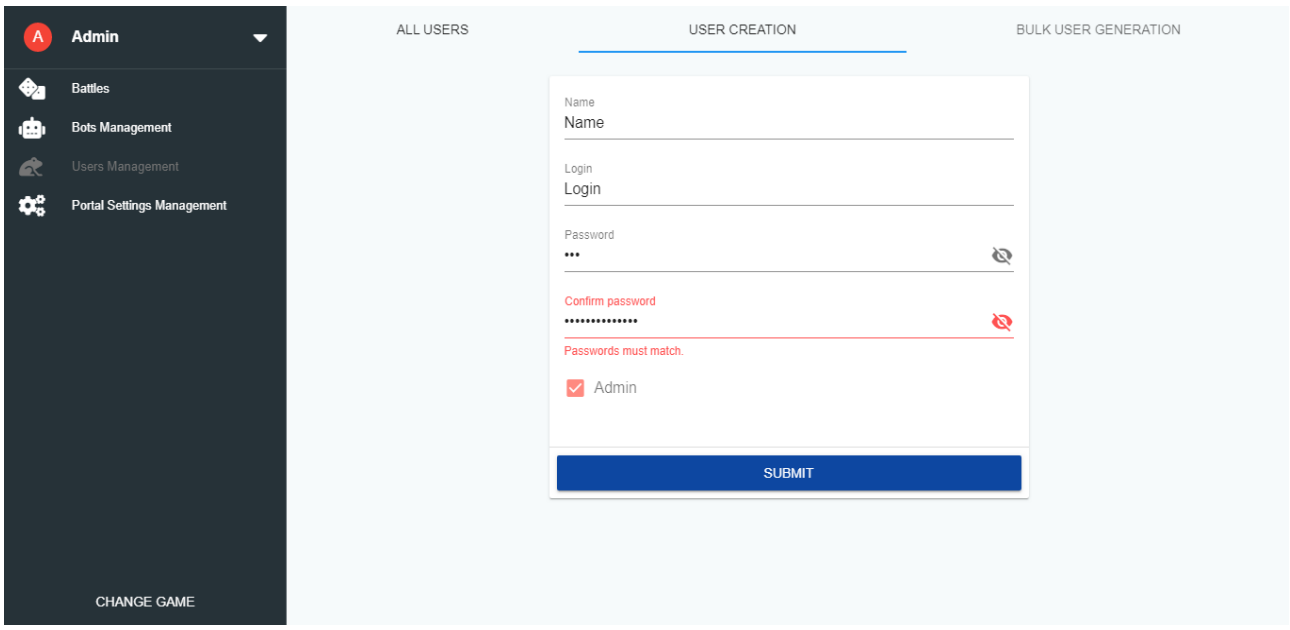

Рисунок 3.15 – Подстраница «User Creation»

На последней подстранице «Bulk User Generation» представлен пользовательский интерфейс для массовой генерации юзеров. Пользователь может выбрать количество требуемых к генерации юзеров, длину их логинов и паролей, редактировать имена юзеров. После успешной генерации система покажет уведомление и диалог, содержащий логины и пароли свежесозданных пользователей. Изображено на рисунке 3.16 и 3.17:

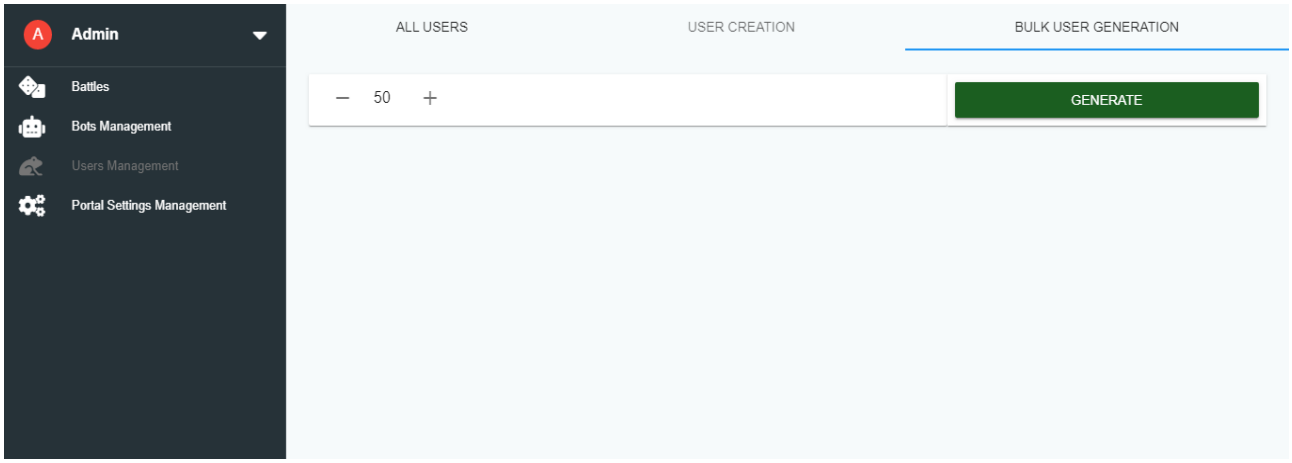

Рисунок 3.16 – Подстраница «Bulk User Generation», выбор количества

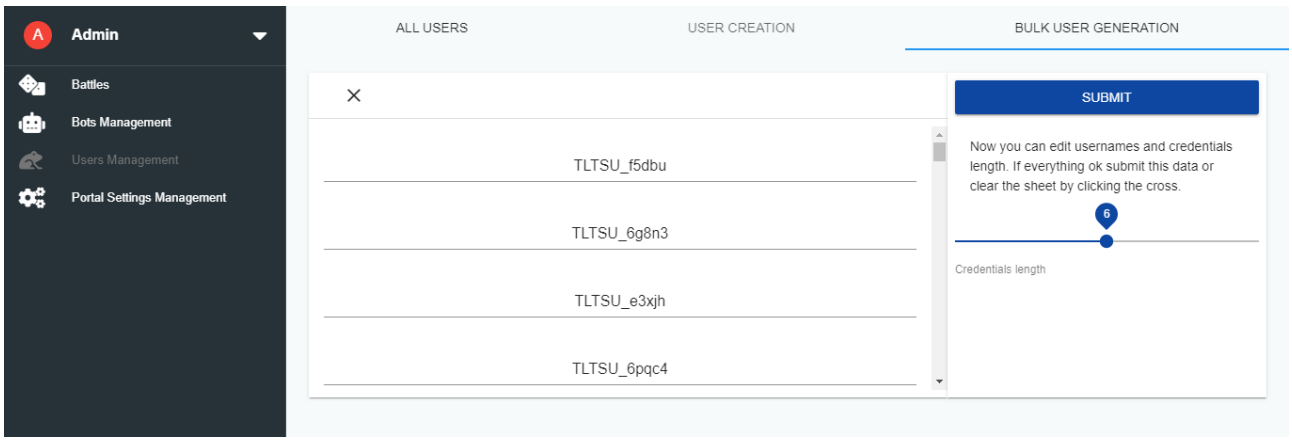

Рисунок 3.17 – Подстраница «Bulk User Generation», редактирование имен и выбор длины логинов и паролей

При клике по пункту меню «Portal Settings Management» пользователь навигируется на страницу управления настройками платформы. Она представлена компонентой, содержащей список настроек с переключателями. При клике на кнопку «Save» пользователь в случае успешного обновления параметров на платформе увидит уведомление.

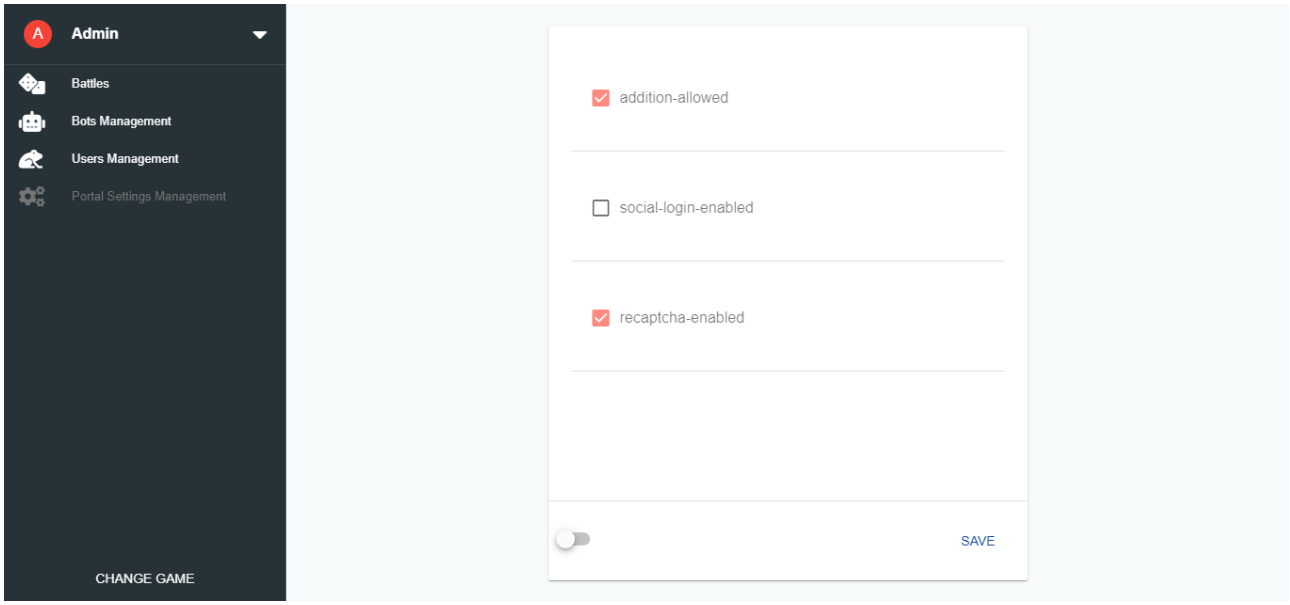

Рисунок 3.18 – Страница «Portal Settings Management»

При клике на пункт меню «Change Game» пользователь перейдет на главную страницу.

В ходе работы была спроектирована система управления платформой AI-

игр, в которой были решены основные функциональные задачи, а также реализовано удобство пользования. Во время описания основного принципа работы системы управления, был представлен алгоритм генерации пользователей. По итогам описания основного принципа работы системы управления AI-игр, можно сделать вывод, что разработанная система работоспособна и удовлетворяет заданным требованиям.

3.5 Тестирование системы управления платформой AI-Battles

Тестирование программного продукта – это процесс, выполняемый для проверки системы на предмет ошибок и соответствия требованиям к этой системе.

Метод «Чёрного ящика» будет использован в качестве метода тестирования. Концепция этого метода состоит в том, что проверка системы, согласно предъявляемым требованиям осуществляется без изучения программного кода.

Для проведения теста, необходимо составить тест кейсы, которые представлены в таблице 3.2.

Таблица 3.2 – Тест-кейсы для системы управления платформой AI-игр

№	Название	Инструкции	Результат
1	Проверка наличия всех необходимых элементов и тестирование способности системы		
1.1	Наличие модуля входа/входа в систему	Проверить наличие модуля входа/выхода в систему	Модуль в наличии и готов к работе.
1.2	Укомплектованность главного меню	Проверить наличие элементов в меню (Battles, Users Management, Bots Managament, Settings)	Все необходимые элементы в наличии и готовы к работе.
1.3	Укомплектованность меню страницы Users Management	Проверить наличие элементов в меню (All Users, Users Creation, Bulk User Generation)	Все необходимые элементы в наличии и готовы к работе.
1.4	Формы для ввода данных	Проверить наличие форм для ввода данных на странице Users Management, а	Все необходимые формы в наличии и готовы к работе.

№	Название	Инструкции	Результат
		также валидацию полей	
1.5	Наличие каждого компонента в каталоге	Проверить наличие всех компонент в каталоге	Компоненты в наличии и готовы к работе.
2	Тестирование функциональных возможностей системы		
2.1	Соответствие элементов меню их компонентам	Проверить соответствие элементов меню их компонентам	Все компоненты соответствуют своим элементам меню
2.2	Соответствие вывода информации о настройках пластформы	Проверить соответствие вывода информации о настройках пластформы	Все настройки выводятся верно
2.3	Соответствие вывода информации о пользователях	Проверить соответствие вывода информации о пользователях пластформы	Все пользователи выводятся верно
2.4	Соответствие вывода информации о сражениях	Проверить соответствие вывода информации о сражениях в пластформе	Все сражения выводятся верно
2.5	Соответствие вывода информации о ботах	Проверить соответствие вывода информации о ботах в пластформе	Все боты выводятся верно
3	Тестирования бизнес-логики		
3.1	Работа панели без аутентификациии	Попытка зайти на главную страницу без аутентификациии	Неаутентифицированного пользователя перенаправляет на страницу аутентификациии
3.2	Работа панели без прав	Попытка аутентифицироваться за пользователя, не имеющего прав	Аутентификация не проходит, появляется уведомление о недостаточных правах доступа
3.3	Модуль управления пользователями	Проверить работоспособность компонент создания пользователя и генерации пользователей	Все компоненты работают нормально
3.4	Модуль управления ботами	Проверить работоспособность удаления бота,	Модуль работает исправно

№	Название	Инструкции	Результат
		загрузки исходного кода, добавления бота, перехода в сражения по боту	
3.5	Модуль сражений	Проверить работоспособность перехода на бота, фильтрации	Модуль работает исправно
3.6	Модуль игр	Проверить работоспособность запуска расчёта соревнований	Модуль работает исправно
4	Тестирования дополнительных функций		
4.1	Выход из системы	Проверить работоспособность выхода пользователя из панели управления	Функция работает исправно

По результатам прохождения тестирования система управления платформой «AI Battles» доказала свою работоспособность и удовлетворила все выдвинутые технические требования.

Выводы по третьей главе

По результатам третьей главы была выбрана архитектура и технология разработки системы управления платформой AI-игр. Была описана схема взаимосвязи модулей приложения и реализована сама система управления, отвечающая предъявленным требованиям заказчика.

Был наглядно представлен функционал системы, экранные формы, были раскрыты возможности системы.

Для проверки работоспособности системы было проведено тестирование методом «Чёрного ящика». После прохождения всех тест-кейсов было доказано, что система является стабильной и исправной, пригодной для использования в реальных, боевых условиях, а также положительно влияет на работу организатора мероприятий. В итоге система управления платформой «AI Battles» была признана работоспособной.

ЗАКЛЮЧЕНИЕ

Итогом ВКР является разработанная система управления платформой AI-Battles. Данная система предназначена для автоматизации и управлением сущностями платформы. Разработанная система существенно облегчит администрирование сервисов платформы.

Во время проектирования системы управления AI-игр был произведен анализ предметной области, на основании которого было принято решение о разработке системы. Были сформулированы основные требования к системе управления платформой AI-Battles и были определены основные функции, которые должна выполнять система.

Было произведено логическое моделирование, построены диаграммы вариантов использования, классов, последовательности, а также концептуальная и логическая модель данных.

После логического моделирования, необходимо было выбрать архитектуру и технологию разработки системы управления платформой AI-игр, в результате чего был выбран прогрессивный веб-фреймворк Vue.js

После выбора средств разработки было произведено описание основных принципов работы системы управления платформой AI-Battles. Система позволяет запускать обсчёт соревнований, просматривать и фильтровать соревнования, просматривать, фильтровать, добавлять, удалять ботов, выгружать их исходный код, просматривать пользователей, регистрировать новых пользователей с правами доступа, в том числе и администраторскими, массово генерировать пользователей, изменять параметры платформы.

Далее было произведено тестирование системы управления AI-играми методом «чёрного ящика». По результатам прохождения тест-кейсов разработанная система была признана работающей исправно и пригодной для использования в реальных условиях.

Реализованная система для управления платформой AI-Battles имеет широкий функционал для администрирования сервисов, тем самым

обеспечивая сокращение временных затрат на обслуживание платформы.

Дальнейшее развитие системы управления платформой AI-Battles предусматривает увеличение функциональных возможностей, а также модернизацию параллельно с платформой AI-Battles.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения (ИСО 5807-85). Введ. 1992-01-01.- М.: Изд-во стандартов, 1992. – 14с.

2. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения. Взамен ГОСТ 24.003-84, ГОСТ 22487-77; Введ. 1992-01-01.- М.: Изд-во стандартов, 1992. – 14с.

3. ГОСТ 34.320-96. Информационные технологии. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы. Введ. 2001-07-01.- М.: Изд-во стандартов, 2001. – 46с. - (Основополагающие стандарты).

4. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Введ. 1990-01-01.-М.: Изд-во стандартов, 1990.– 12с. - (Основополагающие стандарты).

Научная и методическая литература

5. Гринберг А.С. Информационные технологии управления: учебное пособие для вузов / А.С. Гринберг, Н.Н. Горбачев, А.С. Бондаренко. – М.: ЮНИТИ-ДАНА, 2017. – 478 с. – 5-238-00725-6.

6. Гущина, О.М. Методические рекомендации к выполнению выпускной квалификационной работе бакалавра: учеб.-метод. пособие / О.М. Гущина, С.В. Мкртычев, А.В. Очеповский. – Тольятти : ТГУ, 2018. – 77 с.

7. Носова Л.С. Case-технологии и язык UML: учебно-методическое пособие/ Носова Л.С. — Челябинск, Саратов: Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 67 с.

8. Барский А.Б. Параллельные информационные технологии:

учебное пособие/ Барский А.Б.— Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 503 c

9. Смирнов А.А. Технологии программирования: учебное пособие/ Смирнов А.А., Хрипков Д.В.— М.: Евразийский открытый институт, 2011. — 191 c.

10. Ковалевская Е.В. Методы программирования: учебное пособие/ Ковалевская Е.В., Комлева Н.В.— М.: Евразийский открытый институт, 2011. — 320 c.

11. Щербаков А.П. Основные термины и определения компьютерных технологий и автоматизированных систем [Электронный ресурс]: методические указания к лабораторной работе по дисциплине «Основы проектирования и компьютерные технологии»/ Щербаков А.П.— Электрон. текстовые данные. — Липецк: Липецкий государственный технический университет, ЭБС АСВ, 2017. — 8 c.

Электронные источники

12. Vue.js Guidebook. [Электронный ресурс] Режим доступа: https://ru.vuejs.org/v2/guide/index.html свободный (дата обращения 10.03.2019).

13. Vuetify Reference. [Электронный ресурс] Режим доступа: https://vuetifyjs.org/ru/getting-started/quick-start свободный (дата обращения 10.03.2019).

14. Сайт проекта «Habrahabr» [Электронный ресурс]: статья / «Подходы к проектированию RESTful API». Режим доступа: https://habrahabr.ru/company/dataart/blog/277419, свободный (дата обращения 10.03.2019).

15. Терехов А.Н. Технология программирования [Электронный ресурс]: учебное пособие/ Терехов А.Н.— Электрон. текстовые данные.— Москва, Саратов: Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017.— 152 c.— Режим доступа: http://www.iprbookshop.ru/67370.html.— ЭБС «IPRbooks»

16. Сычев А.В. Web-технологии [Электронный ресурс]/ Сычев А.В.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2017.— 184 с.— Режим доступа: http://www.iprbookshop.ru/56344.html.— ЭБС «IPRbooks»

Литература на иностранном языке

17. Djirdeh H., Murray N., Lerner A.: Fullstack Vue: The Complete Guide to Vue.js., 2018. – 442c.

18. Brenda J. ,Saurabh S., Shevat A.: Designing Web APIs : Building APIs That Developers Love. 2018., – 200c.

19. Richardson C.: Microservices Patterns With examples in Java., 2018. – 520c.

20. Poulton N.: Docker Deep Dive., 2018. – 432c.

21. Wall C.: Spring in Action, Fifth Edition, 2018. — 520c.

ПРИЛОЖЕНИЕ А

Фрагмент кода компоненты генерации пользователей

BulkGenerateUsers.vue

```
      offset: 0,
      size: 70,
      remain: 4,
      innerWidth: innerWidth
    };
  },
  computed: {
    ...mapGetters("users", ["isLoading"])
  },
  methods: {
    reset() {
      this.generated = false;
      this.names = [];
    },
    increment() {
      this.count = parseInt(this.count, 10) + 1;
    },
    decrement() {
      this.count = parseInt(this.count, 10) - 1;
    },
    generateUsers(count) {
      let rounds = parseInt(count);
      if (rounds === 0 || isNaN(rounds)) return;

      let generatedNames = [];
      for (var i = 0; i < rounds; i++) generatedNames.push(this.generateName());
      this.names = generatedNames;
      this.generated = true;
    },
    generateName() {
      return (
        this.newUserPrefix +
        Math.random()
          .toString(36)
          .slice(-5)
      );
    },
    submit() {
      this.doBulkGeneration({
        names: this.names,
        credentialsLength: this.loginLength
      }).then(() => {
        this.reset();
      });
    },
    onResize() {
      this.remain = Math.round(this.$refs.content.clientHeight / this.size);
    },
    ...mapActions("users", ["doBulkGeneration"])
  }
};
```

ПРИЛОЖЕНИЕ Б

Фрагмент кода централизованного хранилища ботов

bots.js

```
const actions = {
  getBots: async ({ commit, dispatch }, {userId, search, page}) => {
    commit('request');
    try {
      const botsResponse = await getBots({userId, search, page});
      commit('success');
      commit('setBots', botsResponse.data);
    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
    }
  },
  getBotById: async ({ commit, dispatch }, botId) => {
    commit('request');
    try {
      const botResponse = await getBot(botId);
      commit('success');
      commit('setBotById', botResponse.data);
    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
    }
  },
  deleteBot: async ({ commit, dispatch }, {botId, botName}) => {
    commit('request');
    try {
      await deleteBot(botId);
      dispatch('notification/info', { text: `${botName} deleted` }, { root: true });
      commit('deleteBot', botId);
      commit('success');

    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
    }
  },
  getBotSource: async ({ commit, dispatch }, {botId, botName, userName }) => {
    commit('request');
    try {
      const response = await getBotSource(botId);
      commit('success');
      const dialogPayload = {
        content: response.data,
        title: `${botName} source, owner: ${userName}`,
        saveData: response.data,
        fileName: `${botName}`
      };
      dispatch('dialog/show', dialogPayload, { root: true });
    } catch (err) {
      commit('error');
      dispatch('notification/error', { text: err.message }, { root: true });
```

ПРИЛОЖЕНИЕ В

Фрагмент кода компоненты уведомлений

Notification.vue

```vue
<template>
  <v-snackbar :value="isShown" @input="close" :color="color" :timeout="timeout" top>
    {{ message }}
    <v-btn dark flat @click="close">Close</v-btn>
  </v-snackbar>
</template>

<script>
import { mapActions, mapState, mapMutations } from "vuex";

export default {
  name: "Notification",
  computed: {
    isShown: {
      get() {
        return this.isShown;
      },
      set(value) {
        this.setShowState(value);
      }
    },
    ...mapState("notification", ["timeout", "color", "isShown", "message"])
  },
  watch: {
    isShown: function(value) {
      this.setShowState(value);
    }
  },
  methods: {
    ...mapMutations("notification", ["setShowState"]),
    ...mapActions("notification", ["close"])
  }
};
</script>

<style scoped>
</style>
```

ПРИЛОЖЕНИЕ Г

Фрагмент кода компоненты просмотра сражений

BattlesViewer.vue

```
<v-layout row wrap v-if="!isLoading">
  <template v-for="battle in battles">
    <v-flex xs12 :key="battle.id">
      <v-layout row child-flex wrap mb-2>
        <v-flex xs6 v-for="(pt, i) in battle.participants" :key="i">
          <v-hover>
            <v-card
              tile
              class="battle-card"
              slot-scope="{ hover }"
              v-ripple="{ center: true }"
              :class="`elevation-${hover ? 12 : 2}`"
              @click.native="showBotDetails(pt.id)"
              :color="isOdd(i) ? 'lightRed' : 'lightBlue'"
            >
              <v-card-title>
                <v-list-tile class="grow">
                  <v-list-tile-avatar color="grey darken-3">
                    <v-img class="elevation-6" :src="pt.user.avatar || defautAvatar"></v-img>
                  </v-list-tile-avatar>
                  <v-list-tile-content>
                    <v-list-tile-title>{{ pt.user.name }}</v-list-tile-title>
                  </v-list-tile-content>
                </v-list-tile>

                <v-layout caption justify-end align-center>
                  <span># {{pt.place}}</span>
                </v-layout>
              </v-card-title>

              <v-card-actions>
                <v-list-tile>
                  <v-list-tile-title
                    class="title font-weight-regular text-uppercase"
                  >{{pt.name}}</v-list-tile-title>
                </v-list-tile>
                <v-layout caption justify-end align-center>
                  <span v-if="pt.completionPercentage">{{pt.completionPercentage}} %</span>
                  <span class="mr-1 ml-1">•</span>
                  <span>Score: {{ pt.score }}</span>
                </v-layout>
              </v-card-actions>
            </v-card>
          </v-hover>
        </v-flex>
      </v-layout>
    </v-flex>
  </template>
</v-layout>
<v-container v-else>
  <v-layout align-center justify-center>
```