**Механизм ролей: Клиент-серверное взаимодействие в игре "Крестики-нолики"**

1. Введение

Данный документ описывает механизм взаимодействия клиента и сервера в реализации игры "Крестики-нолики". В данной архитектуре сервер отвечает за обработку игровой логики, управление игровыми комнатами и взаимодействие с базой данных. Клиент предоставляет пользователю интерфейс для подключения к серверу, выбора игровой комнаты, совершения ходов и отображения состояния игры.

2. Роли

2.1 Сервер

\* Обработка подключений: Сервер прослушивает определенный порт для входящих соединений от клиентов. При установлении соединения создается новый сокет для взаимодействия с клиентом.

\* Аутентификация: Сервер проверяет аутентификационные данные (имя пользователя и пароль), полученные от клиента, сравнивая их с данными в базе данных.

\* Управление игровыми комнатами: Сервер предоставляет функционал для создания, присоединения к и выхода из игровых комнат. Отслеживает состояние комнат (свободна/занята, текущие игроки).

\* Обработка игровой логики: Сервер обрабатывает ходы игроков, проверяет наличие победителя, обновляет состояние игрового поля и отправляет информацию об изменениях клиентам.

\* Взаимодействие с базой данных: Сервер хранит информацию о пользователях и играх в базе данных.

2.2 Клиент

\* Подключение к серверу: Клиент устанавливает соединение с сервером по указанному адресу и порту.

\* Аутентификация: Клиент отправляет серверу аутентификационные данные пользователя.

\* Выбор игровой комнаты: Клиент получает список доступных игровых комнат от сервера и позволяет пользователю выбрать комнату для игры.

\* Отображение игры: Клиент отображает пользователю игровое поле, текущий ход и статус игры (победитель, ничья).

\* Отправка ходов: Клиент отправляет серверу информацию о ходе игрока.

3. Взаимодействие

Взаимодействие между клиентом и сервером осуществляется посредством обмена JSON-сообщениями по **установленному TCP-соединению**. Каждое сообщение содержит поле "type", определяющее тип сообщения, и поле "data", содержащее данные, специфичные для типа сообщения.

3.2 Пример сценария игры

1. Подключение и аутентификация: Клиент подключается к серверу и отправляет запрос на аутентификацию. Сервер проверяет данные и отправляет клиенту подтверждение успешной аутентификации.

2. Выбор комнаты: Клиент запрашивает список доступных комнат. Сервер отправляет список комнат. Клиент выбирает комнату и отправляет серверу запрос на присоединение.

3. Начало игры: Сервер подтверждает присоединение к комнате. Если в комнате уже есть другой игрок, сервер отправляет обоим клиентам сообщение `UPDATE\_GAME\_STATE` с начальным состоянием игры.

4. Ход игрока: Клиент отправляет серверу сообщение `ACTION` с информацией о ходе.

5. Обработка хода: Сервер обрабатывает ход, проверяет наличие победителя, обновляет состояние игры и отправляет обоим клиентам сообщение `UPDATE\_GAME\_STATE` с обновленным состоянием игры.

6. Окончание игры: Игра продолжается до тех пор, пока не будет определен победитель или не будет достигнута ничья. Сервер отправляет клиентам финальное сообщение `UPDATE\_GAME\_STATE` с результатом игры.

4. Заключение

Данный механизм ролей обеспечивает разделение обязанностей между клиентом и сервером, что позволяет создать масштабируемую и эффективную реализацию игры "Крестики-нолики". Использование JSON-сообщений для обмена информацией обеспечивает гибкость и расширяемость системы.