ДОКУМЕНТАЦИЯ
по web-приложению «Точки»

Оглавление

Стр.

1. Реализация программы

1.1 Серверная часть

Серверная часть является связующим элементом между клиентами. Он обрабатывает большое количество событий, происходящие до, во время и после игры, поэтому в качестве веб-сервера была выбрана программная платформа Node.js. В его основе лежит событийно-ориентированный и асинхронный подход к программированию с неблокирующим вводом/выводом, что позволяет одновременно обрабатывать множество запросов, не боясь того, что долгие вычисления одного запроса будут задерживать остальные.

1.1.1 Хранение данных

Используя асинхронный подход, программист не может хранить данные в глобальных переменных, т.к. в случае одновременного обращения к ней из двух точек программы могут возникнуть проблемы. Поэтому для хранения данных было использовано хранилище данных Redis, которое поддерживает атомарные неконкурирующие операции. Также оно обладает высокой скоростью доступа, т.к. в процессе работы все данные хранятся в оперативной памяти и периодически копируются на жесткий диск.

Redis является хранилищем вида «ключ-значение», где в качестве значения может быть строка, словарь вида «ключ-значение», множество неповторяющихся строк и множество строк, отсортированное по определенному значению. Ключи и их значения представлены в таблице 2.1.

1.1.2 Клиент-серверное взаимодействие

Для обрабоки клиентских запросов было выбрано расширение для веб-сервера Express, позволяющее использовать методы протокола HTTP, такие как GET и POST, а также заголовки предаваемых запросов, такие как адрес запрашиваемой страницы, для описания запроса. На каждое описание Express позволяет назначать свой обработчик. Таким образом работа приложения разграничивается по функционалу. Все типы клиентских запросов и их обработики указаны в таблице 3.1, данные, которые должен передать клиент в ходе запроса, указаны в таблице 3.2, а данные и коды ошибок, которые возвращает сервер, указаны в таблице 3.3.

Взаимодействуя с клиентом, серверу необходимо не только отвечать на

поступающие запросы, но и отправлять некоторые данные клиенту. Для этого был выбрана модель «издатель-подписчик» (в дальнейшем — pub/sub). В этой модели существуют «каналы», на которые клиент или сервер имеют право публиковать сообщения. Сервер также может фильтровать и оставлять только те сообщения, содержимое которых соответствует API клиент-серверного взаимодействия. Также клиенты могут изъявить желание получать все сообщения, публикуемые на определенном канале, для этого они подписываются на них. Данная модель обеспечивается расширением для веб-сервера Faye, скрывающее внутри себя особенности взаимодействия между подписчиками и издателями и предоставляющее для этого удобный интерфейс. За обработку публикуемых сообщений на сервере отвечает функция `incomingMessage`, вызываемая при каждом сообщении. Клиент должен к каждому сообщению присоединять объект `auth`, который позволяет идентифицировать пользователя. Структура данного объекта описана в таблице 3.5. Сервер в свою очередь проверяет правильность авторизационных данных и удаляет данный объект из сообщения для того, чтобы другие пользователи не могли его увидеть. Если данный объект отсутствует или авторизационные данные не верны, сервер возвращает пользователю код ошибки «401::Authentication required» и не публикует его сообщение. В случае успешной идентификации к сообщению прикрепляется поле `id`, содержащее идентификационный номер пользователя. В дальнейшем сервер проверяет канал, на который было опубликовано сообщение. Если канал служебный и начинается с префикса «/meta», то сообщение пропускается без обработки, за исключением канала «/meta/disconnect». Сообщения на данный канал публикуются пользователем только при отключении от сервера, поэтому другим игрокам сообщается о выходе игрока с помощью сообщения типа «quit». Далее если сообщение публикуется на канал «/game/queue», его обработка передается функции `queueChannel`, а если канал подходит под регулярное выражение `/game/([a-f0-9]+)$`, то оно было отправлено на канал идущей в данный момент игры и его обработка передается функции `gameChannel`.

1.1.3 Канал «/game/queue»

На данном канале находятся игроки, находящиеся в режиме ожидания соперника. Подписавшись на данный канал, игроки получают актуальную информацию обо всех событиях, происходящих в очереди ожидания, таких как появление нового игрока, выход другого игрока, предложения о создании игры

и само создание новой игры. Каждое событие сопровождается публикуемым сообщением на данном канале. Подробная информация о каждом типе сообщений указана в таблице 3.4. Обработчик `queueChannel` обрабатывает все эти сообщения и проверяет их корректность. Также он поддерживает правильность списка ожидающих игроков в хранилище данных, в частности, при сообщении типа «heartbeat» он записывает или обновляет POSIX время последнего появления игрока, отправившего это сообщение, в множество «queue» в хранилище данных. При сообщении типа «quit» он удаляет данного пользователя из множества «queue». При сообщении типа «request» он добавляет имя игрока, отправившего запрос, и его рейтинг, а также вызывает функцию `addRequest`, которая добавляет информацию о запросе в словарь «request:[id]», а также POSIX время запроса в множество «requests» хранилища данных для того, чтобы сборщик мусора удалял запросы с истекшим сроком годности. Также запросы удаляются при сообщении типа «decline» с помощью функции `cancelRequest`. При сообщении типа «accept» обработчик передает запрос в функцию `createGame`, которая проверяет, актуален ли запрос и может ли данный пользователь подтвердить данный запрос. Наконец, если запрос подтвержден успешно, сервер создает новую игру и приглашает в нее игроков с помощью сообщения типа «newgame».

1.1.4 Канал «/game/[channel]

На данном канале находятся игроки, играющие в игру с буквенно-цифровым идентификатором «[channel]»(канал игры). Канал игры формируется случайным образом при ее создании и состоит из 40 битного случайного числа, представленного в шестнадцетеричном формате. На данном канале публикуются все события, происходящие в данной игре, На каждом сообщении, публикуемом на канал, обработчик указывает номер игрока в данной игре: «1» — если сообщение отправил первый игрок, «2» — если сообщение отправил второй игрок, «0» — если игрок, отправивший сообщение, не участвует в игре. Технически, если номер игрока равен «0», то он не может повлиять на игру, но может ее смотреть, но режим просмотра чужой игры отсутствует и не тестировался на реализованном клиенте. Также обработчик запоминает POSIX время игроков в полях «timestamp_1» и «timestamp_2» при получении сообщения типа «heartbeat». При сообщении типа «move» обработчик вызывает функцию `doMove`, которая обрабатывает ход и по итогам хода

добавляет недостающие данные («score», «zones», «captured») в запрос. При сообщении типа «request» он вызывает функцию `addRequest`, которая добавляет информацию о запросе в словарь «request:[id]», а также POSIX время запроса в множество «requests» хранилища данных для того, чтобы сборщик мусора удалял запросы с истекшим сроком годности. Также запросы удаляются при сообщении типа «decline» с помощью функции `cancelRequest`. При сообщении типа «accept» обработчик проверяет, актуален ли запрос и может ли данный пользователь подтвердить данный запрос, и в случае успеха вызывает функцию gameOver, которая останавливает игру с выбранным результатом (ничья или по текущему счету), рассчитывает новый рейтинг Эло игроков и отправляет сообщение типа «gameover».

1.1.5 Обработка хода

Обработка хода — самая трудоемкая задача сервера. Во первых, обработчик проверяет, является ли пункт, на который совершатся ход, захваченным или занятым другой точкой. Если пункт свободен, он занимается точкой цвета того игрока, который выполняет ход. Далее массив, содержащий все зоны окружения, очищается и начинается его заполнение заново. Зоны окружения создаются с помощью функции `findZones`. Для каждого цвета данная функция вызывается отдельно, сначала для цвета игрока, который выполнил ход, затем для цвета его противника. В начале функция `findZones` выполняет «закраску» поля из всех граничных точек. Для этого она вызывает функцию `fillArea` для каждой точки на границе поля. `fillArea` в свою очередь выполняет поиск в глубину из заданной точки, заходя в верхнего, нижнего, левого и правого соседа точки (не выходя за границы поля). Зайти в точку функция `fillArea` может в двух случаях:

1. Цвет точки (массив `field.color`) не равен цвету, по которому в данный момент строятся зоны окружения.
2. Состояние захвата (массив `field.captured`) такое, что точка захвачена противником.

Получившаяся в итоге закрашенная область будет так называемой «открытой» областью, на которой нет зон выбранного цвета. Если пройти по массиву и заменить закрашенную точку незакрашенной и наоборот (инвертировать область), то все зоны выбранного цвета станут закрашенными. Далее вызывается функция `findAreas` для нахождения отдельных зон в этой области.

Она выполняет поиск точек сочленения с помощью функции `findCutpoints`, а затем разделяет область с помощью найденых точек сочленения на зоны с помощью функции `splitAreas`. Дальше убираются те зоны, которые не содержат точек соперника. Для оставшихся функция `borderLine` строит замкнутые линии, ограничивающие данные области, а затем проверяет, чтобы данные линии не проходили через захваченные соперником точки. Для оставшихся после данных операций областей вызывается функция `updateCaptured`, которая «захватывает» каждые точки этих областей выбранным цветом. В конце концов оставшиеся зоны помещаются в массив `field.zones`. После того, как все зоны каждого цвета были найдены, функция `updateScores` обновляет текущий счет игры.

1.2 Клиентская часть

Клиентская часть представляет собой одностраничный HTML-документ. Он выполняется в браузере пользователя и является полностью статичным. Динамичным приложение делают скрипты, написанные на языке программирования JavaScript, который подключаются к этому HTML-документу, взаимодействует с сервером и изменяют содержимое страницы. На этапе проектирования был выбран шаблон проектирования MVC, аббревиатура которого расшифровывается как «модель-представление-контролер». Данный шаблон представляет собой цикл, в котором пользователь управляет контролером, контролер изменяет модель, модель влияет на представление, которое в конце концов видит пользователь. В качестве модели выступает сервер, хранящий и перерабатывающий данные, в качестве контролера выступает скрипт «controller.js» который взаимодействует с сервером, синхронизируя локальные переменные в соответствии с моделью, и с пользователем, а в качестве представления выступают шаблоны на языке HTML5, которые с помощью специальных атрибутов и HTML тегов (директив) самостоятельно обновляются в соответствии с локальными переменными. Каркасом для приложения, работающего по описанной архитектуре, является AngularJS — фреймворк, написаный на языке программирования JavaScript.

1.2.1 Корневой документ

Главным документом данного приложения является файл «index.html». В нем загружаются AngularJS и все остальные скрипты. Связующим элементом для них является файл «app.js», в котором описывается структура приложения, а именно подключаемые модули. К данному приложению подключаются следующие модули:

- ngRoute — встроенный в AngularJS модуль, который позволяет использовать URL адрес для определения выводимого шаблона. Данный модуль позволяет эмулировать многостраничный сайт, используя всего один HTML документ.
- ngStorage — подключаемый сторонний модуль, который позволяет взаимодействовать с локальным хранилищем браузера для хранения своих данных. В данном приложении оно используется вместо cookies для хранения авторизационных данных. Эти данные описаны в таблице 3.5.
- dots.controllers — модуль, в котором хранятся собственные контролеры, описанные в таблице 5.1. Данный модуль хранится в файле «controllers.js».
- dots.directives — модуль, в котором хранятся собственные директивы. Данный модуль храниться в файле «directives.js»

Также в файле описываются правила маршрутизации. Эти правила привязывают к определенным URL адресам вызываемые контролеры и шаблоны, они также описаны в таблице 5.1. Примечательно то, что, при переходе к другому адресу, пользователь не переходит на другой документ, а остается в корневом документе, так как URL адрес приписывается к «index.html» через символ «#». Для этого в «index.html» добавлен тег `<div ng-view></div>`, в который AngularJS загружает с помощью AJAX шаблон в соответствии с URL адресом.

1.2.2 Кореневой контролер

Каждый контролер имеет собственную область видимости и присоединяется к какому-либо HTML тегу. Но также возможно присоединить контролер к тегу, охватывающему все остальные, например к тегу `<body>`. Такой контролер называется корневым, а все остальные, подключенные к тегам внутри `<body>`, являются его потомками, которые наследуют область видимости корневого контролера. Таким образом в корневом контролере можно хранить данные, общие для всех страниц, например авторизационные данные.

В данном приложении корневым контролером является `appCtrl`. При запуске приложения данный контролер запрашивает данные из локального хранилища. Структура этих данных описана в таблице 5.3. Затем корневой контролер запускает клиент для сервера публикации/подписок Faye и настраивает его так, чтобы к каждому сообщению прикреплялись авторизационные данные. Также он настраивает обработчик ошибок при AJAX запросах: в случае ошибки «401::Unauthorized» пользователь должен переместиться на страницу входа.

1.2.3 Страница входа

При открытии приложения пользователь отправляется на страницу входа, на которой находится HTML форма, в которую вводятся имя пользователя и пароль уже зарегистрированного пользователя. Так как имя пользователя и пароль не должны быть пустыми, к полям ввода данных был применена встроенная в AngularJS директива `required`, означающая обязательность ввода. В случае отсутствия данных, пользователю отображается подсказка о необходимости ввести данные. После того, как пользователь нажал на кнопку «Войти», вызывается функция `$scope.login`, которая занимается обработкой формы. Она проверяет наличие введенных данных и отправляет эти данные на сервер. Если сервер успешно авторизировал пользователя, то он отправляет авторизационные данные, описанные в таблице 3.5, которые контролер записывает в локальное хранилище. Затем пользователь отправляется в комнату ожидания соперника. Если сервер вернул код ошибки, контролер сообщает клиенту о неправильно введенных данных. Если пользователь еще не зарегистрирован, то он может нажать на ссылку «Зарегистрироваться» и перейти на страницу регистрации.

1.2.4 Страница регистрации

Если пользователь заходит в приложение в первый раз, ему необходимо пройти процесс регистрации. На странице регистрации также находится HTML форма, в которую вводятся желаемые имя пользователя и пароль регистрируемого пользователя. К обоим полям применена директива `required` , которая означает обязательность ввода данных в эти поля. Также к полю ввода пароля была применена директива `ng-minlength="6"`, которая означает минимальную длину вводимого пароля. При нажатии на кнопку «Зарегистрироваться» вызывается функция `$scope.register`, которая занимается обработ-

кой формы. Она проверяет соответствие введенных данных требованиям и отправляет эти данные на сервер. Если данные не соответствуют требованиям, пользователю отображается соответствующая подсказка. Если сервер успешно произвел регистрацию, пользователь уведомляется об этом и появляется ссылка, ведущая на страницу входа. Если сервер не смог зарегистрировать пользователя, значит пользователь с таким именем уже существует, о чем и уведомляется пользователь.

1.2.5 Комната ожидания соперника

При входе в комнату ожидания соперника, контролер проверяет наличие авторизационных данных в локальном хранилище и в случае неудачи отправляет пользователя на страницу входа. Далее контролер запрашивает у сервера информацию о других пользователях, и сохраняет ее в переменной `$scope.queue`, в соответствии с которой обновляется представление в виде HTML шаблона. Далее контролер подписывается на канал «/game/queue», на котором он следит за событиями, описаными в таблице 4.1. Также запускается счетчик, который каждые 5 секунд публикует сообщения типа «heartbeat». При входе/выходе игроков соответственно обновляется очередь ожидающих игроков. Если пользователю предлагается начать новую игру, то отображается счетчик обратного отсчета на 15 секунд и кнопки, с помощью которых пользователь может начать или отменить новую игру. По истечении времени или в случае отмены предложения соперником игрок теряет возможность начать предложенную игру, о чем он уведомляется на одну секунду.

Если пользователь самостоятельно выбирает понравившегося пользователя, то ему отображается счетчик обратного отсчета на 15 секунд и кнопка, с помощью которой пользователь может отменить предложение. По истечении времени или в случае отказа противника игрок уведомляется об этом на одну секунду.

Если оба пользователя согласны с тем, чтобы начать новую игру, контролер отписывается от канала «/game/queue» и отправляется на страницу игры.

1.2.6 Счетчик обратного отсчета

Все счетчики обратного отсчета в игре ализованы с помощью созданой для этой цели директивы `<countdown>`, описаной в файле «directives.js». Данная директива также имеет свой шаблон, описаный в файле `countdown.html`.

Данной директиве можно передать время, на которое будет включаться счетчик, переменные, за которыми она будет следить и в соответствии с которыми она будет изменять свое состояние и представление. Также в данную директиву передаются функции, вызываемые при определенных событиях и сообщения, отображаемые вместе с этими событиями. Все параметры и их описания указаны в таблице 5.2.

1.2.7 Игровой процесс

При входе на страницу игры, контролер проверяет наличие авторизационных данных в локальном хранилище и в случае неудачи отправляет пользователя на страницу входа. Далее контролер запрашивает у сервера информацию о текущем состоянии игры, а именно поле, которое сохраняет в объект `$scope.field`, структура которого описана в таблице 2.2, идентификационные номера игроков, участвующих в игре, и текущий счет. В зависимости от идентификационных номеров игроков, участвующих в игре контролер определяет номер игрока, которым является пользователь и сохраняет этот номер в переменной `$scope.player`. Далее он выполняет функцию отрисовки поля `$scope.redrawField` и функцию отрисовки зон окружения `$scope.redrawZones`.

Игровое поле представляет из себя HTML тег `<canvas>`. Данный тег позволяет рисовать внутри себя геометрические примитивы, а также, как и любой другой тег, позволяет обрабатывать нажатия мышкой внутри поля. Но примитивы нельзя удалять по отдельности, поэтому поле отрисовывается каждый раз при изменении состояния игры заново. Также нельзя определить примитив, на которой пользователь нажал мышкой, известны только координаты относительно левого верхнего угла, поэтому определением пункта, на который было совершено нажатие, занимается функция `$scope.canvasClick`. Контролеру известно расстояние между линиями, на которых размещаются пункты (оно является постоянным и находится в переменных `$scope.ceilWidth` и `$scope.ceilHeight`), а также расстояние от начала координат до первой линии (оно является постоянным и находится в переменных `$scope.offsetX` и `$scope.offsetY`). Для того, чтобы определить координаты пункта, из координат нажатия мышки необходимо вычесть расстояние от начала координат до первой линии, а затем разделить на расстояние между линиями и округлить до ближайшего целого.

После того, как игровое поле было отрисовано, контролер подписывается на канал игры «/game/[channel]» и начинает следить за событиями, ко-

торые там происходят и которые описаны в таблице 4.2. Затем запускается счетчик, который каждые 5 секунд публикует сообщения типа «heartbeat». Контролер поддерживает номер игрока, который в данный момент должен совершать ход, который хранится в переменной `$scope.current_player` и в соответствии с которым обновляется надпись над игровым полем. Если номер игрока соответствует тому, кто должен совершать ход, пользователь может выбрать пункт, на который он будет ставить точку. При нажатии на пункт публикуется сообщение типа «move». Такое же сообщение может опубликовать и соперник, если он совершает ход. Контролер следит за этими сообщениями и обновляет состояние игры в соответствии с ними. Также контролер следит за предложениями о ничье или завершении игры и отображает счетчики обратного отсчета. Наконец, если на канале будет опубликовано сообщение типа «gameover», контролер перемещает игрока в комнату ожидания соперника.

2. Структуры данных

Таблица 2.1 — Структура хранилища данных

Ключ	Тип	Значение
user:[id]	Словарь	Содержит данные об игроках. Вместо [id] необходимо указывать идентификационный номер пользователя.
user:[id]->name	Строка	Имя пользователя.
user:[id]->password	Строка	Пароль в открытом виде.
user:[id]->token	Строка	Авторизационный токен, который клиент должен передавать в каждом запросе для подтверждения авторизации. Выдается клиенту при вводе логина и пароля в окне авторизации.
user:[id]->rating	Строка	Рейтинг пользователя.

Продолжение таблицы 2.1

Ключ	Тип	Значение
last_userid	Строка	Идентицикационный номер пользователя, использованный при регистрации последнего пользователя. Используется для обеспечения уникальности идентификационных номеров. Начальное значение — «0»
users	Словарь	Имена пользователей, сопоставленные с их идентификационными номерами.
queue	Упорядоченное множество	Идентификационные номера пользователей, находящихся в режиме ожидания соперника. Номера упорядочены по времени последнего появления, представленного в виде количества секунд с 1 января 1970 года (POSIX время).
requests	Упорядоченное множество	Идентификационные номера пользователей, требующих некоторого действия от другого пользователя (запрос), упорядоченные по времени создания запроса.
request:[id]	Словарь	Содержит данные о запросе. Вместо [id] необходимо указывать идентицикационный номер пользователя.

Продолжение таблицы 2.1

Ключ	Тип	Значение
request:[id]->type	Строка	Тип запрашиваемого действия. Может принимать значения: «newgame» — создание новой игры, «draw» — закончить игру в ничью, «surrender» — закончить игру с текущим счетом.
request:[id]->target	Строка	Идентификационный номер пользователя, от которого требуется принять решение — выполнить предлагаемый запрос или нет.
request:[id]->gameid	Строка	Идентификационный номер игры, в рамках которой должен выполнится запрос. Если тип запроса равен «newgame», то значением данной строки будет являться «0».
games	Словарь	Буквенно-цифровой идентификатор игры (канал), сопоставленный с идентификационным номером игры.

Продолжение таблицы 2.1

Ключ	Тип	Значение
last_gameid	Строка	Идентификационный номер последней созданной игры. Используется для обеспечения уникальности идентификационных номеров. Начальное значение — «0».
game:[id]	Словарь	Содержит данные об игре. Вместо [id] необходимо указывать идентификационный номер игры.
game:[id]->channel	Строка	Канал игры
game:[id]->field	Строка	Состояние поля в текущий момент. Храниться в виде объекта, представленного в JSON-нотации. Подробное описание полей этого объекта можно узнать в таблице 2.2
game:[id]->player_1	Строка	Идентицикационный номер первого игрока.
game:[id]->score_1	Строка	Количество очков, набранное первым игроком на текущий момент.
game:[id]->timestamp_1	Строка	Время последнего появления первого игрока в формате POSIX.
game:[id]->player_2	Строка	Идентицикационный номер второго игрока.

Продолжение таблицы 2.1

Ключ	Тип	Значение
game:[id]->score_2	Строка	Количество очков, набранное вторым игроком на текущий момент.
game:[id]->timestamp_2	Строка	Время последнего появления второго игрока в формате POSIX.
game:[id]->current_player	Строка	Номер игрока, который совершает ход в данный момент времени. «1» — если в данный момент времени ходит первый игрок, «2» — если в данный момент времени ходит второй игрок.

Таблица 2.2 — Структура объекта «game[id]:field»

Поле	Значение
width	Ширина поля. Значение по умолчанию — 39
height	Высота поля. Значение по умолчанию — 32
color	Двумерный массив, содержащий цвет каждого пункта на поле. Цвет равен «0» если данный пункт не содержит точки, «1» — если точка на пункте принадлежит первому игроку, «2» — если точка на пункте принадлежит второму игроку.
captured	Двумерный массив, содержащий состояние захвата каждого пункта на поле. Состояние равно «0» если пункт не находится в зоне окружения, «1» — если пункт находится в зоне окружения (включая границы) перового игрока, «2» — если пункт находится в зоне окружения (включая границы) второго игрока.

Продолжение таблицы 2.2

Поле	Значение
zones	Массив, содержащий все зоны окружения существующие на поле в данный момент времени. Каждый элемент массива также является массивом, содержащим пункты, находящие на границе выбранной зоны окружения. Пункт является объектом, поля которого описаны в таблице 2.3

Таблица 2.3 — Структура объекта, содержащего координаты пункта

Поле	Значение
x	Номер горизонтальной линии, на которой располагается пункт. Линии нумеруются сверху-вниз, начиная с нуля.
y	Номер вертикальной линии, на которой располагается пункт. Линии нумеруются слева-направо, начиная с нуля.

3. RESTful API

Таблица 3.1 — HTTP запросы

URL	Метод	Обработчик	Описание
/auth	POST	onAuth	Авторизация в системе. В ответ клиент получает уникальные авторизационные данные, которые ему необходимо передавать в каждых последующих запросах для подтверждение личности.
/register	POST	onRegister	Регистрации нового пользователя в системе.
/queue	GET	getQueue	Получение списка игроков, находящихся в режиме ожидания соперника.
/gamedata	GET	getGameData	Получение текущего состояния игры.

Таблица 3.2 — Передаваемые данные HTTP запросов

URL	Метод	Передаваемые параметры	Значение
/auth	POST	name	Имя авторизуемого пользователя
		password	Пароль
/register	POST	name	Имя регистрируемого пользователя
		password	Пароль
/queue	GET	id	Идентификационный номер пользователя
		token	Авторизационный токен
/gamedata	GET	id	Идентификационный номер пользователя
		token	Авторизационный токен
		channel	Канал запрашиваемой игры

Таблица 3.3 — Получаемые данные HTTP запросов

URL	Получаемые данные	Значение
/auth	id	Идентификационный номер авторизированного пользователя
	token	Авторизационный токен
	HTTP статус	200 в случае успешной авторизации, 401 в случае неправильных авторизационных данных.
/register	HTTP статус	200 в случае успешной регистрации, 406 если уже существует пользователь с регистрируемым именем пользователя.
/queue	queue	Массив в формате JSON, содержащий очередь игроков, ожидающих соперника. Каждый элемент массива является объектом, структура которого описана в таблице 3.4
	HTTP статус	200 в случае успешного получения данных, 401 в случае неавторизированного доступа.

Продолжение таблицы 3.3

URL	Получаемые данные	Значение
/gamedata	game	JSON объект, содержащий текущее состояние игры. Данный объект является словарем «game:[id]», структура которого описана в таблице 2.1.
	HTTP статус	200 в случае успешного получения данных, 401 в случае неавторизированного доступа.

Таблица 3.4 — Структура объекта, хранящего данные о пользователе

Поле	Значение
id	Идентификационный номер пользователя.
name	Имя пользователя.
rating	Рейтинг пользователя.

Таблица 3.5 — Структура объекта, хранящего авторизационные данные

Поле	Значение
id	Идентификационный номер пользователя.
token	Авторизационный токен, выданный клиенту при авторизации.

4. Pub/sub API

Таблица 4.1 — Сообщения, публикуемые на канале «/game/queue»

Тип	Параметры	Описание
heartbeat	type	Данный тип сообщений предназначен для поддержания актуальности списка игроков, ожидающих соперника. Игрок исключается из этого списка, если в течении 10 секунд он не публиковал ни одного сообщения данного типа.
	id	Идентификационный номер пользователя, отправившего сообщение. Данное поле добавляется сервером.
	name	Имя пользователя, отправившего сообщение. Данное поле добавляется сервером.
	rating	Рейтинг пользователя, отправившего сообщение. Данное поле добавляется сервером.
quit	type	Данный тип сообщений публикуется игроком в том случае, если он добровольно исключает себя из списка игроков, ожидающих соперника, либо сервером, если игрок принудительно исключается из этого списка. Игрок может уже отсутствовать в списке, т.к. игрок может отправить это сообщение одновременно с сервером.
	id	Идентификационный номер выходящего пользователя. Данное поле добавляется сервером.

Продолжение таблицы 4.1

Тип	Параметры	Описание
request	type	Данный тип сообщений отправляется игроком в том случае, если он предлагает другому игроку начать новую игру.
	id	Идентификационный номер пользователя, отправившего предложение. Данное поле добавляется сервером.
	user.name	Имя пользователя, отправившего предложение. Данное поле добавляется сервером.
	user.rating	Рейтинг пользователя, отправившего предложение. Данное поле добавляется сервером.
	target	Идентификационный номер пользователя, которому предлагается начать новую игру.
accept	type	Данный тип сообщения отправляется игроком при согласии с предложением начать новую игру.
	id	Идентификационный номер согласившегося пользователя. Данное поле добавляется сервером.
	target	Идентификационный номер пользователя, предложение которого одобряет согласившийся игрок.
decline	type	Данный тип сообщения отправляется игроком для отмены предложения. Отменить предложение может либо игрок, отправивший его, либо игрок, принимающий его.
	id	Идентификационный номер игрока, отменяющего предложение. Должен совпадать либо с полем «init», либо с полем «target». Данное поле добавляется сервером.
	init	Идентификационный номер игрока, изначально отправившего предложение.
	target	Идентификационный номер игрока, который должен принять предложение.

Продолжение таблицы 4.1

Тип	Параметры	Описание
newgame	type	Данный тип сообщения обозначает начало новой игры. Его имеет право отправить только сервер.
	id	Идентификационный номер пользователя, отправившего сообщение. В данном случае он может принимать единственное значение «0», обозначающее сервер игры.
	player_1	Первый игрок, участвующий в новой игре.
	player_2	Второй игрок, участвующий в новой игре.
	channel	Канал игры, на который приглашаются игроки.

Таблица 4.2 — Сообщения, публикуемые на канале «/game/[channel]»

Тип	Параметры	Описание
heartbeat	type	Данный тип сообщений предназначен для проверки наличия игроков в игре. Если игрок не отправлял сообщения данного типа в течение 10 секунд, игрок считается выбывшим.
	id	Идентификационный номер пользователя, отправившего сообщение. Данное поле добавляется сервером.
	player	Номер игрока, отправившего сообщение, «1» — если первый игрок, «2» — если второй игрок, «0» — если игрок не участвует в игре. Данное поле добавляется сервером.

Продолжение таблицы 4.2

Тип	Параметры	Описание
move	type	Данный тип сообщения отправляется игроком в том случае, если он совершает ход.
	id	Идентификационный номер пользователя, совершившего ход. Данное поле добавляется сервером.
	player	Номер игрока, совершившего ход. Данный номер обязательно должен совпадать с игроком, который в данный момент должен ходить. Данное поле добавляется сервером.
	point	Объект, содержащий пункт, на который сходил игрок. Структура объекта описана в таблице 2.3.
	score	Массив, содержащий количество очков у каждого игрока. Элемент с индексом [0] всегда равен «0», с индексом [1] равен количеству очков первого игрока, с индексом [2] — количество очком второго игрока. Данное поле добавляется сервером.
	zones	Массив, содержащий все зоны окружения существующие на поле в данный момент времени. Является полной копией поля «zones», содержащегося в хранилище данных объекта «game[id]:field», описанного в таблице 2.2. Данное поле добавляется сервером.
	captured	Двумерный массив, содержащий состояние захвата каждого пункта на поле. Является полной копией поля «captured», содержащегося в хранилище данных объекта «game[id]:field», описанного в таблице 2.2. Данное поле добавляется сервером.

Продолжение таблицы 4.2

Тип	Параметры	Описание
request	type	Данный тип сообщений отправляется игроком в том случае, если он предлагает другому игроку закончить игру.
	id	Идентификационный номер пользователя, отправившего предложение. Данное поле добавляется сервером.
	player	Номер игрока, отправившего предложение. Не может принимать значение «0», т.к. закончить игру может только участник этой игры. Данное поле добавляется сервером.
	requestType	Способ, по которому в случае окончания игры будет выбираться победитель. «draw» — если игру предлагается закончить в ничью, «surrender» — победитель будет выбран согласно текущему счету.
accept	type	Данный тип сообщения отправляется игроком при согласии с предложением закончить игру.
	id	Идентификационный номер согласившегося игрока. Данное поле добавляется сервером.
	player	Номер согласившегося игрока. Не может принимать значение «0», т.к. закончить игру может только участник этой игры. Данное поле добавляется сервером.
decline	type	Данный тип сообщения отправляется игроком для отмены предложения.
	id	Идентификатор игрока, отменяющего предложение. Данное поле добавляется сервером.
	player	Номер игрока, отменяющего предложение. Не может принимать значение «0», т.к. закончить игру может только участник этой игры. Данное поле добавляется сервером.

Продолжение таблицы 4.2

Тип	Параметры	Описание
gameover	type	Данный тип сообщения обозначает конец игры. Его имеет право отправить только сервер.
	id	Идентификационный номер пользователя, отправившего сообщение. В данном случае он может принимать единственное значение «0», обозначающее сервер игры.
	score_1	Количество очков, набранное первым игроком.
	score_2	Количество очков, набранное вторым игроком.
	winner	Результат игры. «1» — если победил первый игрок, «2» — если победил второй игрок, «0» — если ничья.

5. Работа AngularJS

Таблица 5.1 — Контролеры

Название	Сопоставляемый URL	Шаблон	Описание
appCtrl	/	index.html	Корневой контроллер.
loginFormCtrl	/login	login.html	Контроллер формы входа.
registrationFormCtrl	/register	register.html	Контроллер формы регистрации.
waitingRoomCtrl	/queue	queue.html	Контроллер комнаты ожидания соперника.
gameScreenCtrl	/game/[channel]	game.html	Контроллер игрового процесса.

Таблица 5.2 — Параметры к директиве обратного отсчета

Поле	Описание
ng-show	Внешняя логическая переменная, которая «истинна», когда нужно включить и показать счетчик обратного отсчета и «ложна», когда его нужно выключить и скрыть. Если счетчик останавливается с помощью данной переменной, то никаких других событий не вызывается.
time	Параметр, характеризующий время, на которое включается счетчик.
on-accept	Функция, вызываемая при нажатии кнопки «Принять». Необязательный параметр, если присутствует параметр «on-cancel».
on-decline	Функция, вызываемая при нажатии кнопки «Отклонить». Необязательный параметр, если присутствует параметр «on-cancel».
on-cancel	Функция, вызываемая при нажатии кнопки «Отменить». Необязательный параметр, если присутствует параметр «on-accept» и «on-decline».
is-failure	Внешняя логическая переменная, которая в состоянии «истинно» выключает счетчик и показывает сообщение «failure-message».
failure-message	Сообщение, показываемое при выключении счетчика с помощью переменной «is-failure».
timeout	Сообщение, показываемое при выключении счетчика по окончании времени.

Таблица 5.3 — Структура локального хранилища

Поле	Описание
loggedIn	Логическая переменная, показывающее состояние авторизации. Если переменная «истинна», то пользователь авторизирован в системе и имеет право заходить на любые страницы. Если переменна «ложна», то пользователь допускается только к страницам входа и регистрации.

Продолжение таблицы 5.3

Поле	Описание
user	Объект, хранящий данные об авторизированном пользователе. Структура этого объекта описана в таблице 3.4.
auth	Объект, хранящий авторизационные данные. Структура этого объекта описана в таблице 3.5.

6. Руководство пользователя

Чтобы открыть игру, вам необходимо открыть сайт «http://acm.tpu.ru/dots/». После этого вы будете перемещены на страницу входа.

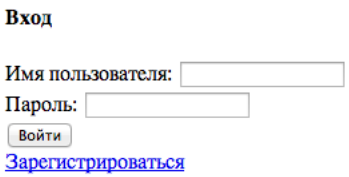

Рисунок 6.1 — Страница входа

Чтобы войти в игру, вы должны ввести имя пользователя и пароль. После этого нажмите кнопку «Войти». Если какое-либо поле не было заполнено, будет отображена ошибка.

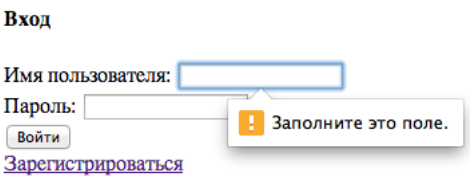

Рисунок 6.2 — Пустое поле

Если введенные имя пользователя и пароль оказались неверными, будет отображена ошибка.

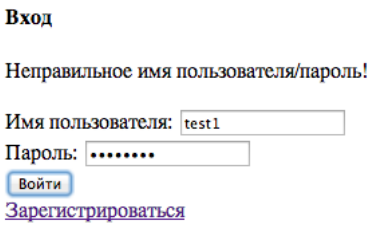

Рисунок 6.3 — Неправильное имя пользователя/пароль

Если вы заходите в игру в первый раз, вам необходимо пройти процедуру регистрации. Для этого нажмите кнопку «Зарегистрироваться» на странице входа. После этого вы будете перемещены на страницу регистрации.

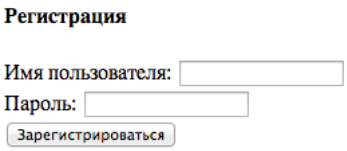

Рисунок 6.4 — Страница входа

На странице регистрации вам необходимо ввести желаемые имя пользователя и пароль и нажать на кнопку «Зарегистрироваться». Если какое-либо поле не было заполнено, будет отображена ошибка рядом с незаполненным полем. Также пароль должен содержать не менее 6 символов. Если введенный пароль менее 6 символов, будет отображена ошибка.

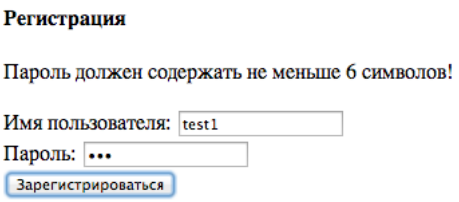

Рисунок 6.5 — Пароль содержит менее 6 символов

Если введенное имя пользователя уже существует, будет отображена ошибка.

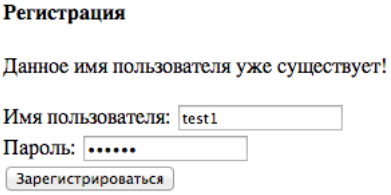

Рисунок 6.6 — Страница входа

Если регистрация пройдет успешно, вам будет отображено сообщение.

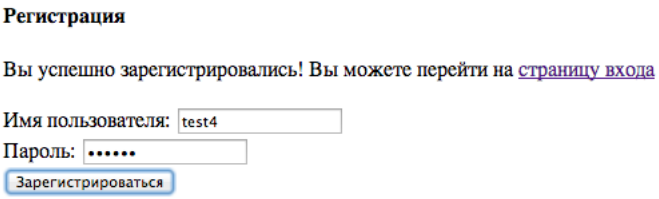

Рисунок 6.7 — Успешная регистрация

После этого вы можете перейти по ссылке на страницу входа и ввести свое имя пользователя и пароль. Если вход в систему будет произведен успешно, вы переместитесь в комнату игроков, ожидающих соперника.

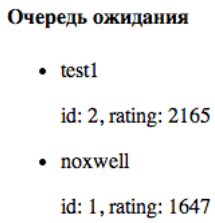

Рисунок 6.8 — Комната ожидания

В этой комнате вы можете выбрать из списка игроков соперника, с которым вы хотите начать новую игру. После того, как вы определились с выбором, вы можете нажать на имя пользователя соперника. После этого вы увидите счетчик обратного отсчета.

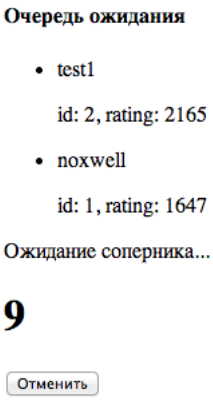

Рисунок 6.9 — Ожидание соперника

В это время у соперника также появится счетчик обратного отсчета.

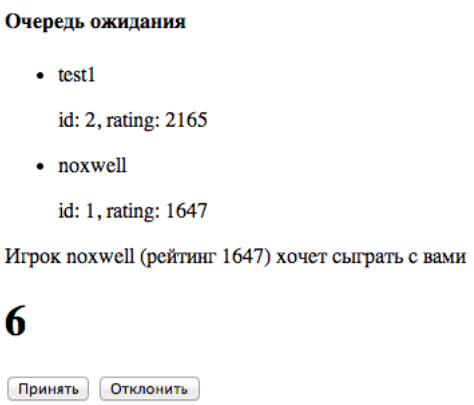

Рисунок 6.10 — Игрок хочет сыграть с вами

В течение 15 секунд вы можете отменить свое решение начать новую игру с этим соперником. Для этого вы должны нажать на кнопку «Отменить». В этом случае сопернику будет отображено сообщение.

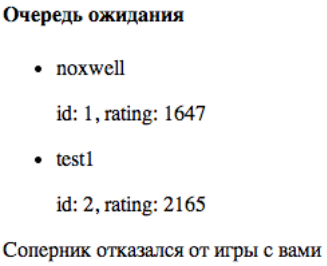

Рисунок 6.11 — Соперник отказался от игры с вами

Также и соперник может отказаться от игры с вами, нажав кнопку «Отклонить». В этом случае вам будет отображено такое же сообщение, как на рисунке 6.11. Если ни вы, ни соперник не примете решение в течение 15 секунд, вам обоим отобразится сообщение.

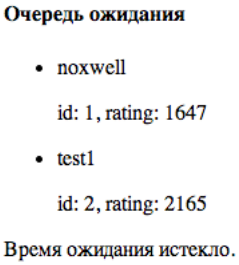

Рисунок 6.12 — Время ожидания истекло

Если соперник согласится начать с вами новую игру нажав кнопку «Принять», то вы оба перейдете на страницу игры.

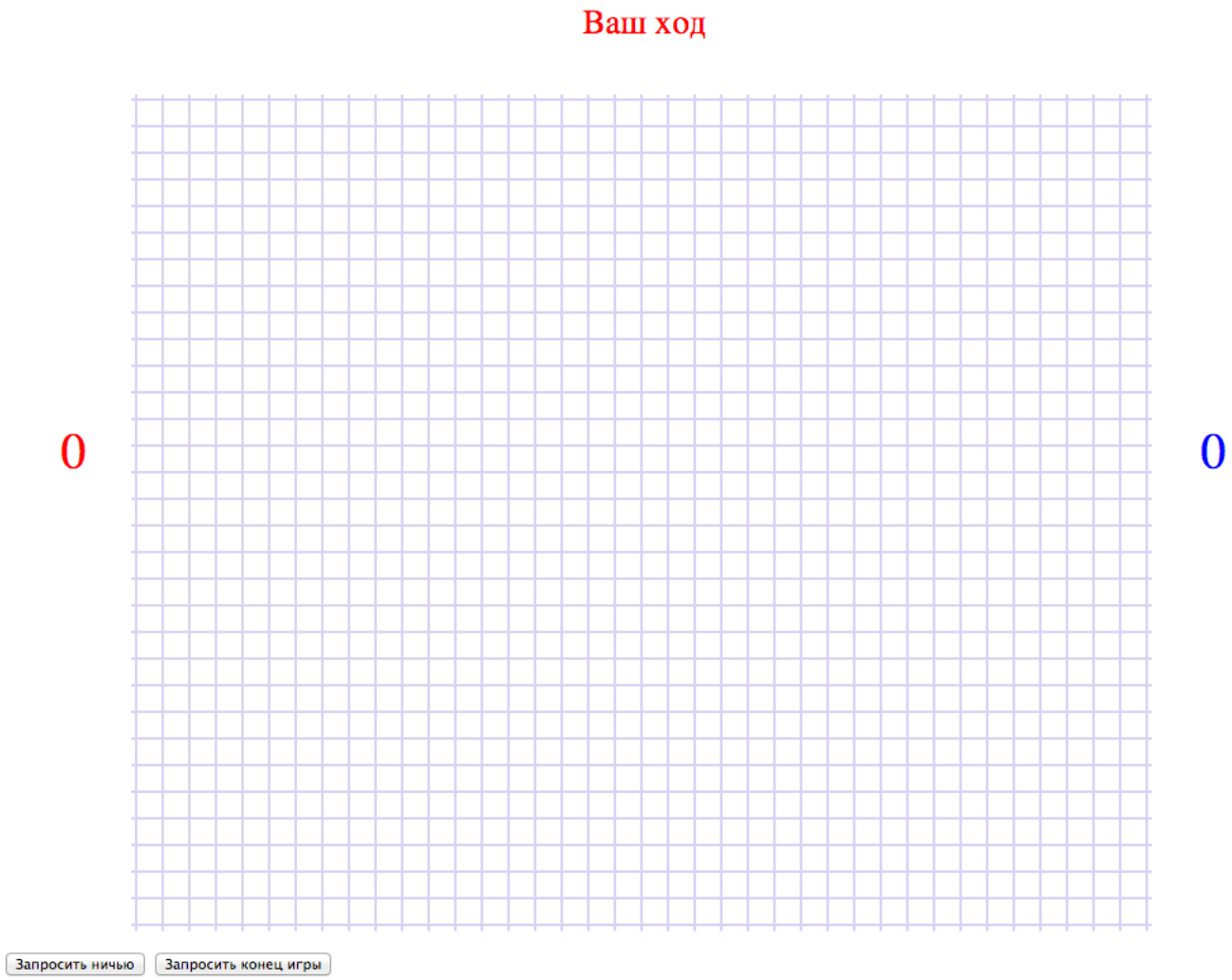

Рисунок 6.13 — Игровой экран

Посередине экрана располагается игровое поле. Слева располагаются набранные очки первого игрока, справа — очки второго игрока. Игрок, предложивший начать игру ходит первым. Если в данный момент вы должны ходить, над игровым полем будет отображено сообщение «Ваш ход», как на рисунке 6.13. Если в данный момент ходить должен соперник, вы не можете ходить и вам будте отображено сообщение «Ход соперника».

Рисунок 6.14 — Ход соперника

Ход осуществляется путем нажатия левой кнопкой мыши на пересечение вертикальных и горизонтальных линий на игровом поле (пункт). После того, как ход был совершен, на поле появляется точка определенного цвета, для точек первого игрока — красный, для точек второго — синий. Если после совершения хода точки вашего цвета можно соединить непрерывной замкнутой линией, а также в замыкаемой области находятся точки соперника, данная область окружается, а вам присваиваются очки за каждую точку соперника в этой области.

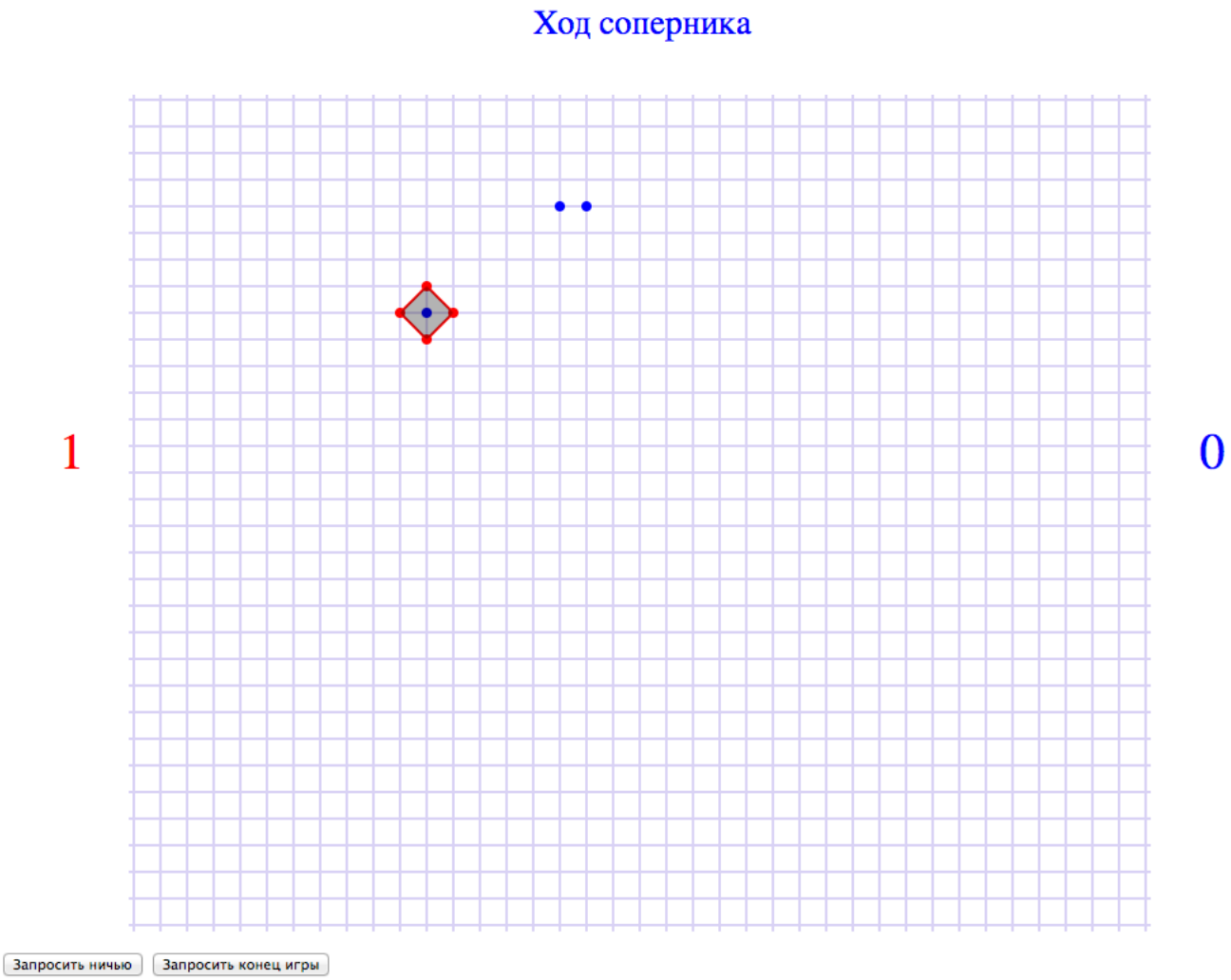

Рисунок 6.15 — Зона окружения

Вы можете попросить соперника завершить игру вничью, нажав кнопку «Запросить ничью», или с текущим счетом, нажав на кнопку «Запросить конец игры». После этого у вас запустится счетчик обратного отсчета.

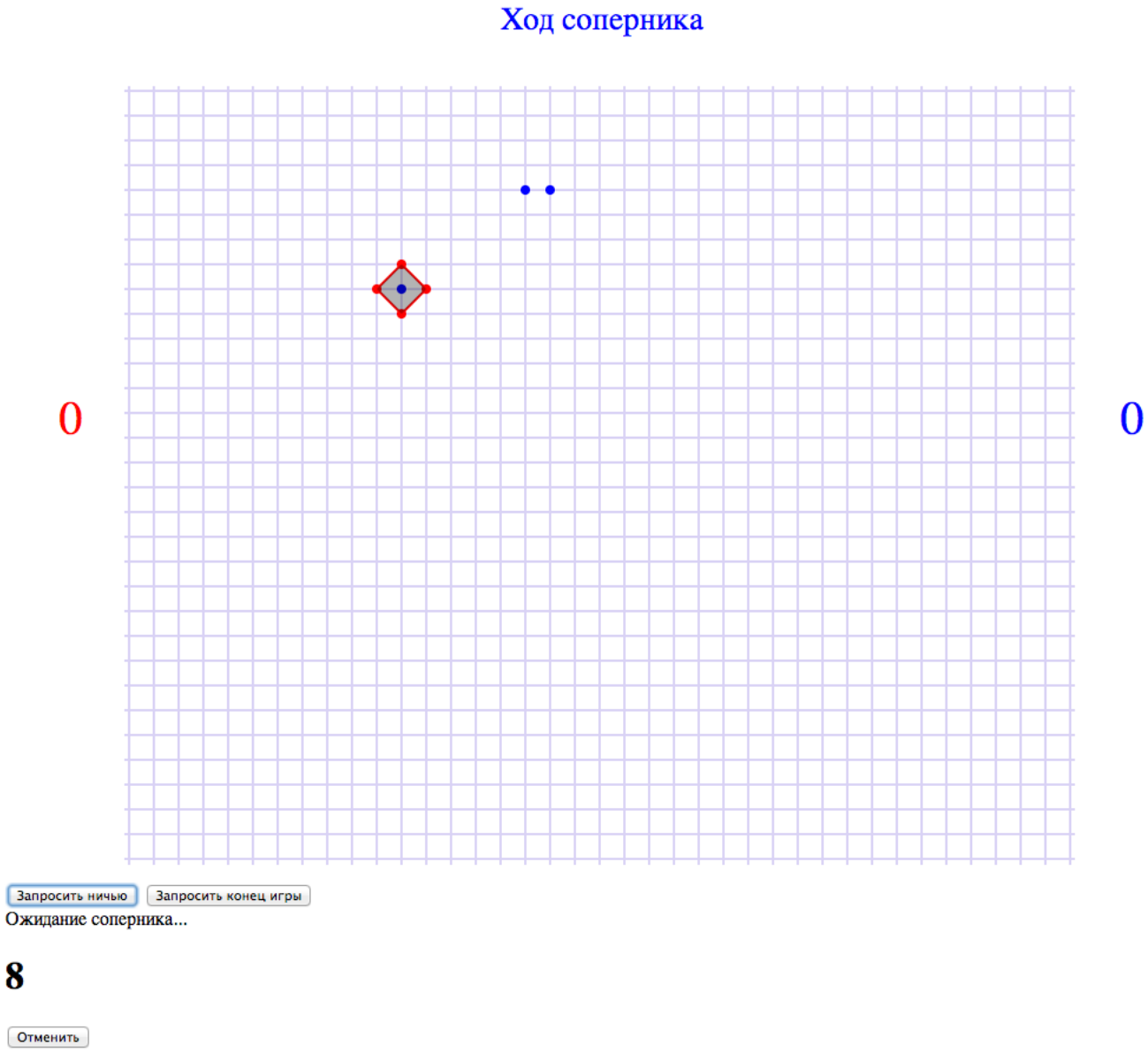

Рисунок 6.16 — Ожидание соперника

Аналогично у соперника появится счетчик обратного отсчета.

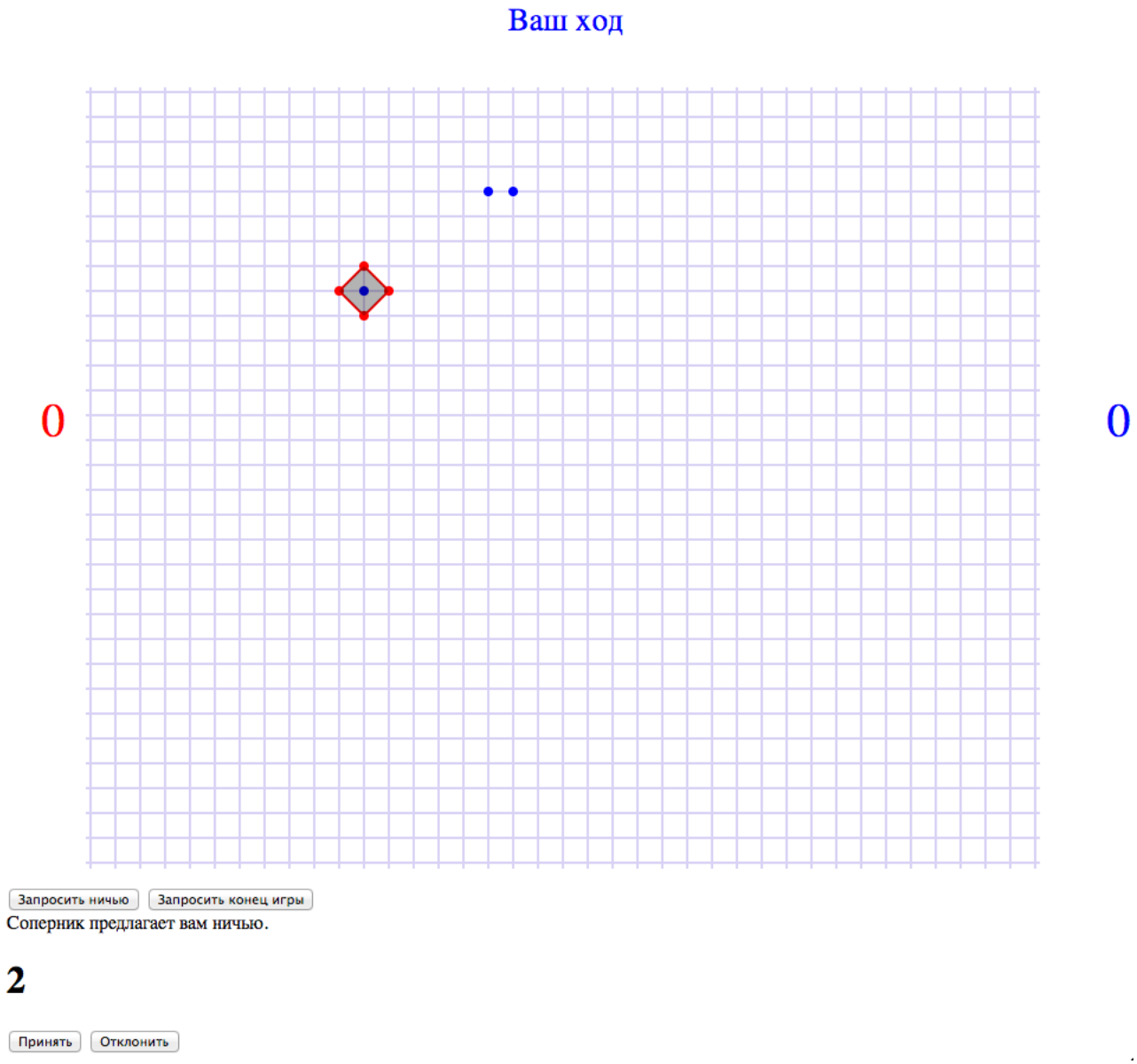

Рисунок 6.17 — Соперник предлагает завершить игру

В течение 15 секунд вы или соперник должны сделать выбор. Счетчик работает аналогично тому, который был в комнате ожидания, с помощью кнопок под счетчиком вы можете отменить запрос, аналогично соперник может отклонить его, о чем вам будет сообщено. Если соперник согласиться завершить игру, игра завершается, победитель награждается повышением рейтинга, у проигравшего рейтинг понижается. В случае ничьи рейтинг увеличивается у того, у кого изначально рейтинг был меньше. Рейтинг изменяется по особой формуле. После завершения игры вы возвращаетесь на страницу ожидания соперника.