

Тестовое задание

Игра

"Крестики-Нолики"



Содержание

Основная задача	2
Описание API	2
Стандарт на формат ответа сервера	3
Инициирование новой партии	4
Создание новой партии	4
Получение списка партий	4
Присоединение к существующей партии	5
Осуществление игрового процесса	6
Осуществление хода	6
Получение текущего состояния	6
Требования к серверу	8
Требования к клиенту	9
Пример верстки	10
Экран списка всех партий	10
Экран созданной партии	11
Экран подключения к партии	12
Экран партии в режиме просмотра	13

Основная задача

Написать приложение крестики нолики на доске "3 на 3". Приложение состоит из клиента и сервера. Для соискателей на вакансию backend разработчик требуется выполнить серверную часть, для соискателей на вакансию frontend разработчик клиентскую. Также приветствуется выполнение обеих частей.

При выполнении задания не требуется использовать Long polling, WebSockets и иные технологии, реализующие постоянный дуплексный канал связи между клиентом и сервером. Достаточно осуществлять регулярный опрос сервера на предмет изменения состояния раз в 2 секунды (см. описание API)

Выполненное тестовое задание присылать ссылкой на репозиторий.

Реализация на языке TypeScript, и в крайнем случае на JavaScript. Репозиторий должен содержать описание проекта, инструкции запуска и сборки проекта. Особым плюсом будет наличие тестов, для проверки работоспособности приложения.

К заданию необходимо отнестись с серьезностью и продемонстрировать все умения и знания которые у вас есть.

Дополнительное задание: Возможность создать доску своего размера

■ ■ ■

Описание API

Размещение сервера должно быть локальным и соответствовать виду <http://localhost:{PORT}>. Далее в примерах будет заменен на \${HOST}.

Все запросы являются RAW HTTP запросами с ContentType: application/json. Примеры запросов имеют вид:

- **Method:** \${method}
- **PATH:** \${path}
- **REQUEST BODY:** <request_body>
- **RESPONSE:** <response>

где \${method} любой из определенных стандартом HTTP (GET, POST, DELETE, ...),

■ ■ ■



Стандарт на формат ответа сервера

В случае успеха операции:

```
{
  "status": "ok",
  "code": 0
  "message": "ok"
  ...           // прочие поля с результатами выполненной операции
}
```

В случае возникновения ошибки:

```
{
  "status": "error",
  "code": <ERROR_CODE >, // код ошибки
  "message": "Something bad happened" // описание ошибки
}
```



Инициирование новой партии

Игроки делятся на 2 типа: создающие игру (играют крестиками) и присоединяющиеся к ней (играют ноликами). После создания / присоединения к игре игроки получают индивидуальный accessToken, который аутентифицирует их в игре.

Создание новой партии

- **Method:** post
- **PATH:** /games/new
- **REQUEST BODY:**

```
{
  "userName": "John Doe",
  size: 3 /* размер поля */
}
```

- **RESPONSE:**


```
{
  "status": "ok",
  code:0,
  "accessToken": "768b762c8c28",
  "gameToken": "123abc"
}
```

gameToken передается другому игроку по любому из имеющихся каналов связи, чтобы он мог присоединиться к игре.

Получение списка партий

- **Method:** get
- **PATH:** /games/list
- **REQUEST BODY:**
- **RESPONSE:**

```
{"status": "ok", code: 0, games: [
  {
    gameToken: "123abc",
    owner: "Chuck Norris", // автор игры
  }
]}
```



```
        opponent: "", // присоединенный игрок
        size: 3, // размер игрового поля
        gameDuration: 12323, // сколько уже идет игра в миллисекундах
        gameResult: "" || "owner" || "opponent" || "draw" // кто выиграл партию
        state: "ready" || "playing" || "done" // статус игры, "ready" – готов, "playing" –
идет игра, "done" - завершена
    }, ...
}
```

Присоединение к существующей партии

- **Method:** post
- **PATH:** /games/join
- **REQUEST BODY:**

```
{
  "gameToken": "123abc",
  "userName": "Chuck Norris"
}
```

- **RESPONSE:**

```
{
  "status": "ok",
  code:0,
  "accessToken": "5aec6c5d082f"
}
```

После создания игры и входа второго игрока, дальнейшее присоединение игроков возможно в режиме «просмотра партии».



Осуществление игрового процесса

Все методы игрового процесса требуют аутентификации путем передачи accessToken через HTTP header.

Осуществление хода

- **Method:** post
- **PATH:** /games/do_step
- **REQUEST BODY:**

```
{
  "row": 1,
  "col": 2
}
```


- **RESPONSE:**

```
{
  "status": "ok",
  code: 0
}
```

Получение текущего состояния

- **Method:** get
- **PATH:** /games/state
- **REQUEST BODY:**
- **RESPONSE:**

```
{
  "status": "ok",
  "code": 0,
  "youTurn": true, // true если сейчас ходит запрашивающий
  "gameDuration": 2342 // в миллисекундах
  "field": [
    "X?0",
```



```
"?X0",  
"???"  
],  
"winner" : "Chuck Norris" // выставляется если в игре определился победитель  
}
```

Если в инициированной игре нет активности в течение 5 минут, игра считается прерванной и данные о ней удаляются.

Требования к серверу

Реализация обязательна с использованием БД. Сервер должен быть stateless и не использовать хранение состояний в сессии пользователя.

У сервера должна быть минимальная конфигурация:

- HTTP порт
- Строка подключения к БД (в случае использования)

Технологии, которые необходимо задействовать на выбор:

- node.js LTS. исключительно ES6+
- Express.js / Nest.JS
- MongoDB / PostgreSQL / MySQL
- Система конфигурации окружения

Плюсом будет использование систем контейнеризации как Docker.

Требования к клиенту

Клиент должен быть приложением, выполняющимся в браузере и использующим один из JS фреймворков или нативные платформы (iOS, Android):

- AngularJS (1, 2)
- reactJS, reactNative
- Vue
- jQuery
- ...

Страницы приложения:

- Список текущих игр для возможности подключения
- Игровое поле включает в себя:
 - текущим состоянием партии, ожиданием игрока и временем игры
 - просмотр результатов партии или слежение за другой партией

Клиентский код должен собираться системой сборки на выбор: webpack, makefile, etc...

Наличие тестов приветствуется.

Прототип доступен по ссылке <https://invis.io/ATAZGLOUZ>

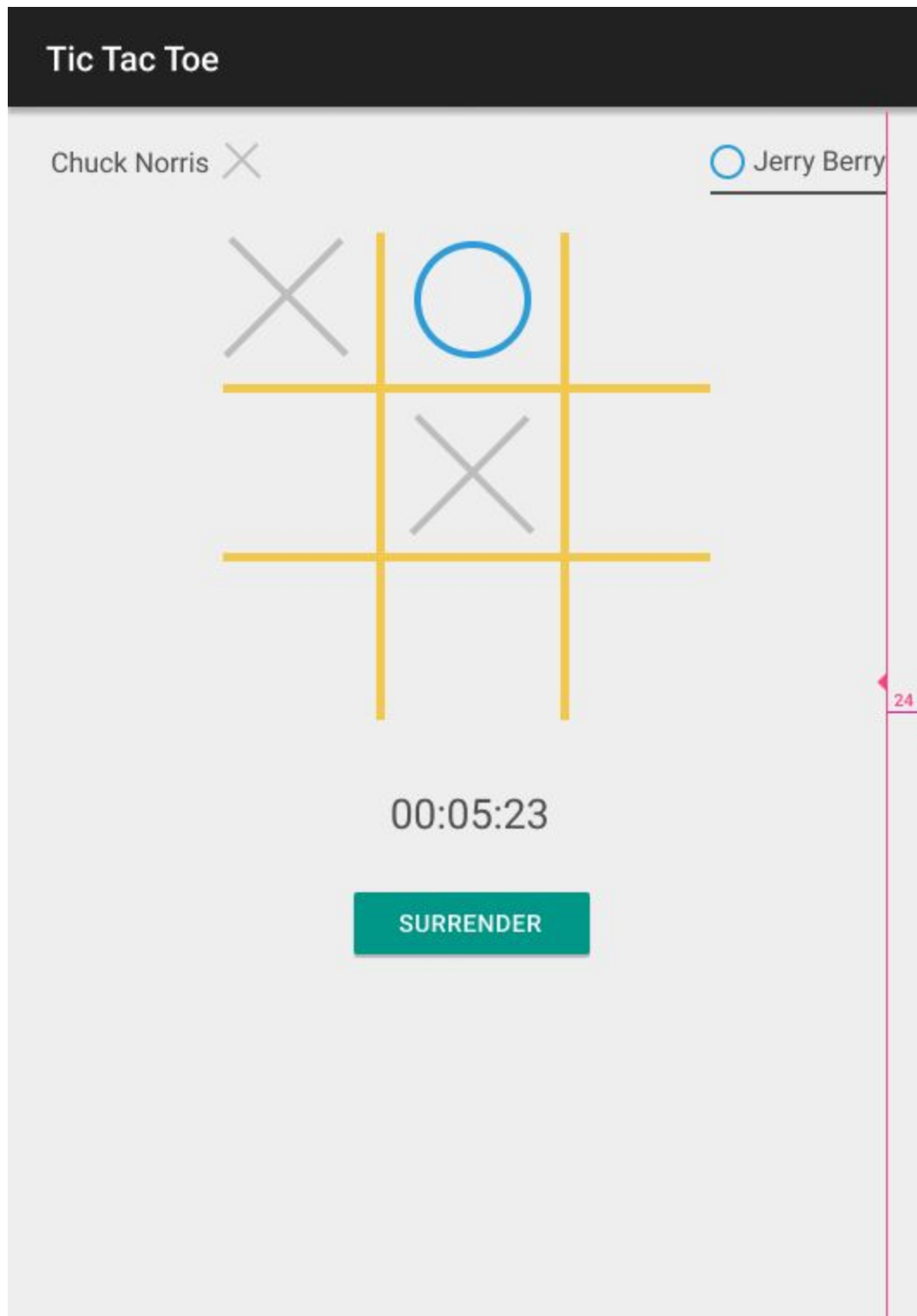
Доп. задание: Возможность вести несколько игр параллельно

Пример верстки

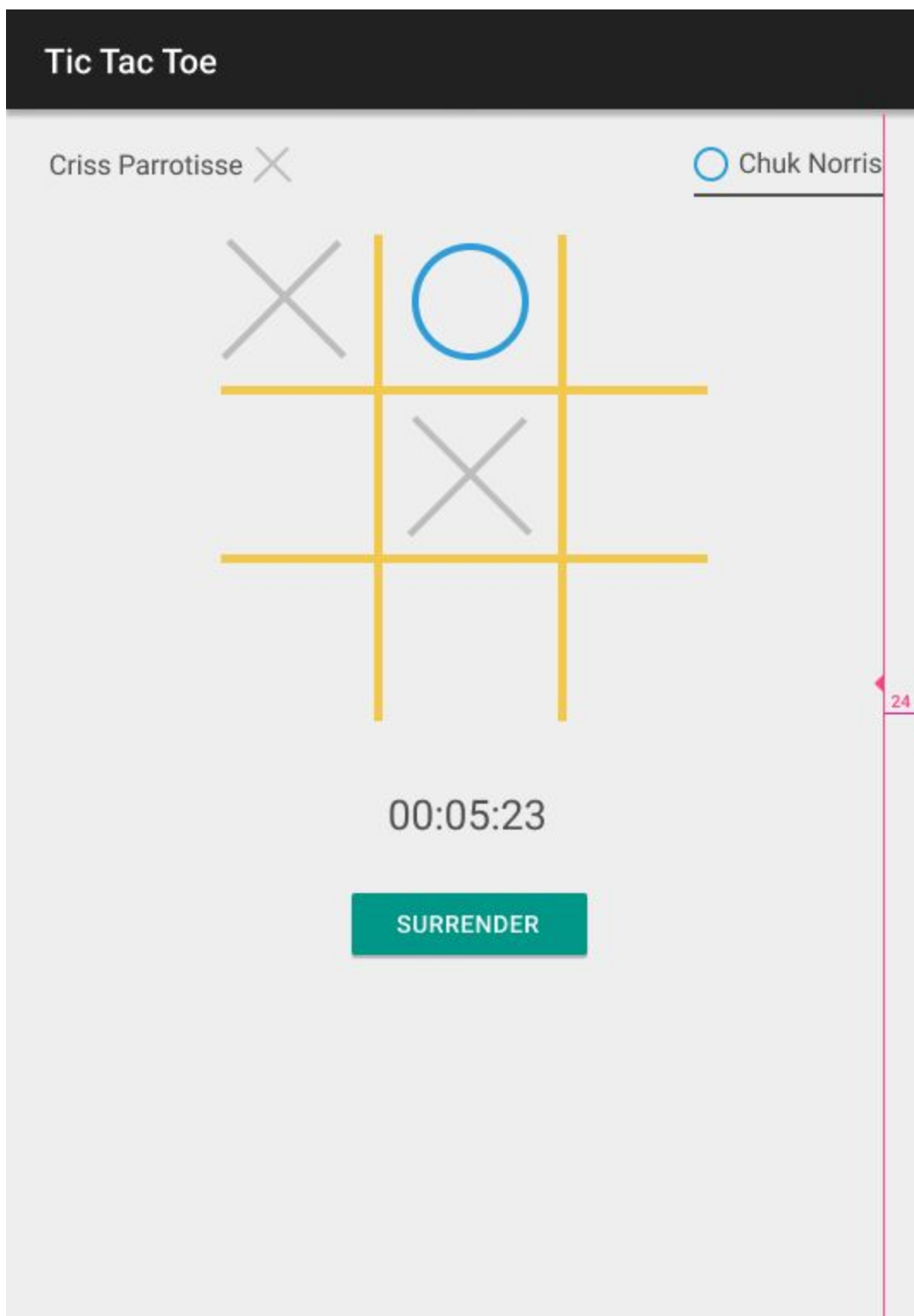
Экран списка всех партий



Экран созданной партии



Экран подключения к партии



Экран партии в режиме просмотра

