



**Universidade Federal da Grande Dourados
Faculdade de Ciências Exatas e Tecnologia
Bacharelado em Sistemas de Informação**

**Gustavo Mateus Tinoco (2011001687)
Michel Éder Matos (2011001386)**

TRABALHO PRÁTICO SOCKET

**Dourados – MS
Setembro/2016**

Introdução

Para iniciar o desenvolvimento foi iniciada busca sobre a API Socket.IO que é responsável por conexões RPC via navegador. Para começar foi feita a aplicação a plataforma de desenvolvimento *server-side* baseado em aplicações Javascript onde o cliente executa parte da execução da aplicação. Para instanciação do servidor foi usado um *Framework Web* para *Node.JS* que é o *Express* que fornece uma gama de ferramentas e recursos para aplicações Web.

Com a busca de conhecimento para tais ferramentas foi adotado um tutorial que dá instruções de como iniciar um server por Dan Nawara(2016) utilizando *NojeJS* e *Express*.

Desenvolvimento

A partir destas ferramentas citadas acima deles foi criado o arquivo “server.js” em que é onde acontece toda a mágica da aplicação onde são incluídos os *frameworks* citados. É colocado o número da porta que é: 8090. Descrevendo o código inicialmente decide-se instanciar a conexão com o comando chamando a *API.sockets.on('connection')* faz a chamada de conexão, dentro deste método são instanciados todas as funções que emitem ou recebem informação do jogo via transição de sockets. Os métodos utilizados são:

‘turn’: Socket que aguarda posição e qual player enviou a jogada. Também emite um ‘turnadd’ que envia aos clientes qual jogador e qual posição. Se jogador 1 ou jogador 2 ele envia valores diferentes. Também faz a validação de que não é a vez através do ‘naoehsuavez’ no cliente.

‘geticon’: Socket que aguarda qual ícone deve ser imposto ao cliente. Emite um ‘icon’ em que o cliente recebe e faz a transição para o ícone desejado (0) ou (X).

'win' : Socket que envia ao jogador(1 ou 2) qual mensagem deve ser exibida se ganhou ou perdeu, e emite também uma chamada que atualiza o placar dos jogadores.

'getchance' : Socket que verifica se o player e o socket do servidor são iguais e emite uma mensagem que é convertida pelo *Jquery* para transformação do texto de placar em branco e preto através do comando do *Jquery* `$('.classe').css('color',#fff);`

'disconnect' : Socket que valida a desconexão do usuário, e emite ao cliente uma *'reload'* que recebe e faz uma recarregamento na página do jogo.

Métodos de socket no game.js

Inicialmente o jogo só começa quando existem dois jogadores. Existe o método principal que valida a quantidade de usuários conectados ao servidor e emite a mensagem aos respectivos clientes. Conforme descrito abaixo os métodos utilizados no jogo.

'users' : Socket que valida a quantidade de usuários no servidor, caso 2 ele efetua um *append* nas tag <body> da página e inicia o jogo. Neste mesmo emite a mensagem de qual ícone foi clicado e qual resposta deve obter do *'geticon'* do servidor. Também foi necessário a implementação se caso existam 3 ou mais usuários e a estes a mensagem exibida será que tem que aguardar o servidor liberar algum dos que já estão jogando.

'reload' : Socket que aguarda do servidor caso seja requisitada efetua através do javascript um recarregamento da página.

'turnadd' : Principal socket que faz a validação do jogo. Pega todos os campos em que existem 0 ou X converte em array e faz uma validação com as possíveis combinações de vitória, empate e derrota do jogo. Caso seja verdadeiro alguma delas, é emitido ao servidor *'win'* um objeto contendo o jogador que enviou de acordo com o cliente. Caso o tamanho da movimentação seja igual a 8, é emitido ao servidor a chamada de empate e não são marcados scores para ambos os lados dos jogadores.

'minhaescolha' : Socket que aguarda do servidor quem fez a escolha e altera no rodapé do jogo para preto ou branco qual jogador deverá executar a próxima ação.

'naoehsuavez' : Socket que aguarda mensagem pois o jogador clicou em uma jogada que não era sua vez de jogar.

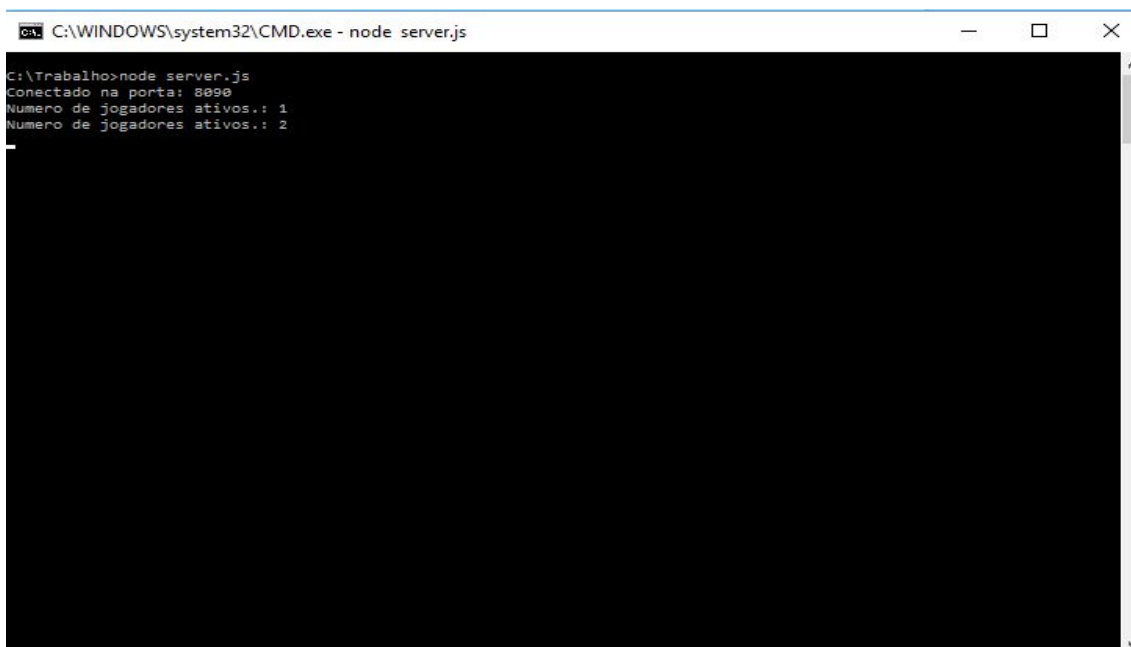
'winmsg' : Socket que recebe do servidor a mensagem de ganhador. Para ambos lados é disparado e exibido a mensagem de quem ganhou. Logo é emitido ao servidor no *'checkscore'* e resetado o jogo por um timeout de 3 segundos.

'atualizaplacar' : Socket que recebe uma requisição do servidor através do *'win'* e exibe qual lugar deverá marcar os pontos. Neste caso são zerados os placares onde são exibidos e através de uma variável global é feito um append dos valores recebidos do servidor.

Exemplo de execução

Primeiramente iniciar o servidor através do console, ambos em Windows ou Linux é necessário entrar no diretório que está instalado a aplicação e acrescentar o comando para Windows: *'node server.js'* e para linux *'nodejs server.js'* dentro da pasta do trabalho.

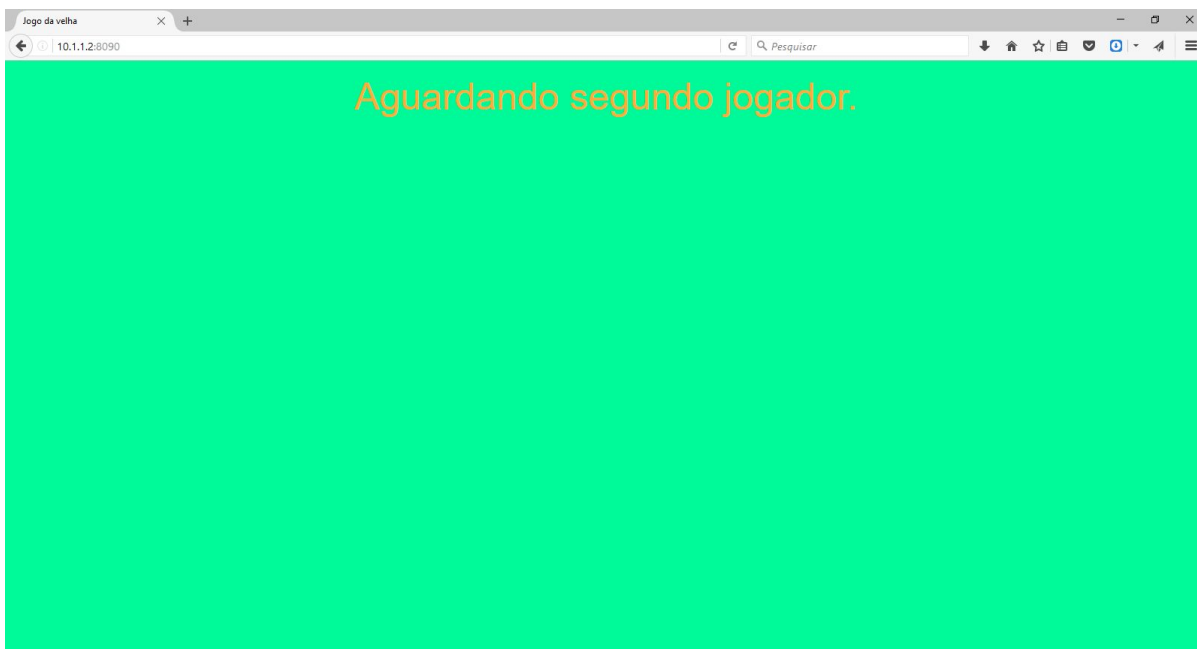
Imagem 1 - Exemplo no Windows.



```
C:\WINDOWS\system32\CMD.exe - node server.js
C:\Trabalho>node server.js
Conectado na porta: 8080
Numero de jogadores ativos.: 1
Numero de jogadores ativos.: 2
```

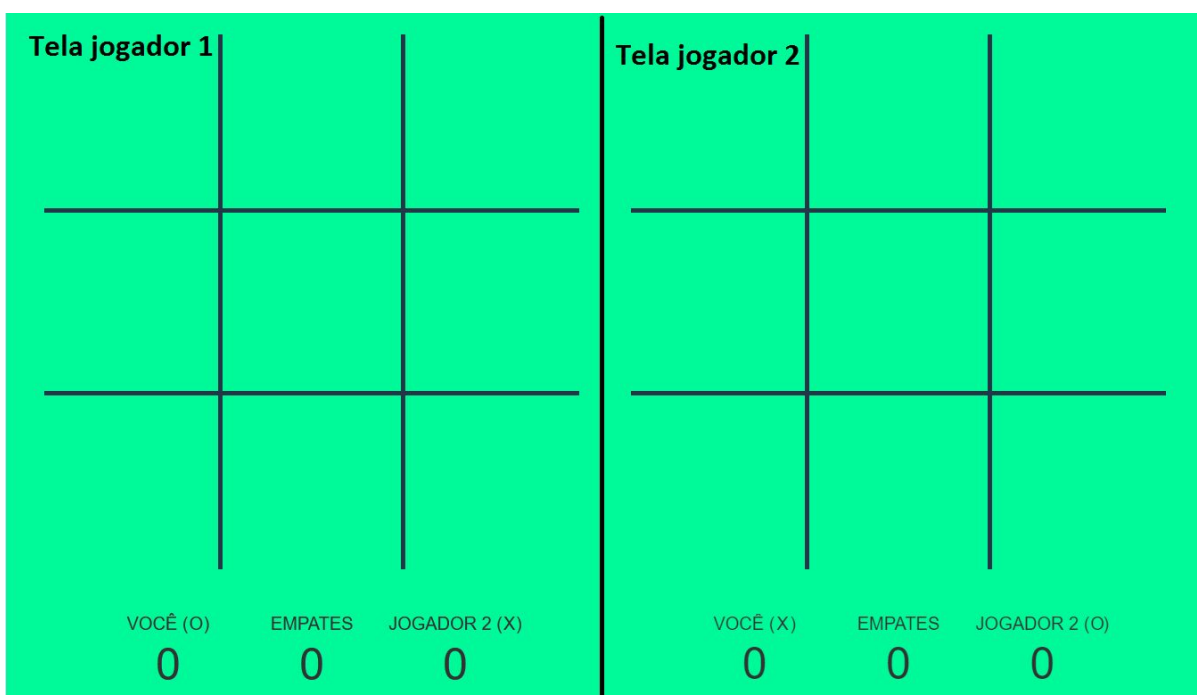
Após o primeiro jogador a conectar é exibido a tela de aguarde.

Imagem 2 - Tela de aguardando jogador



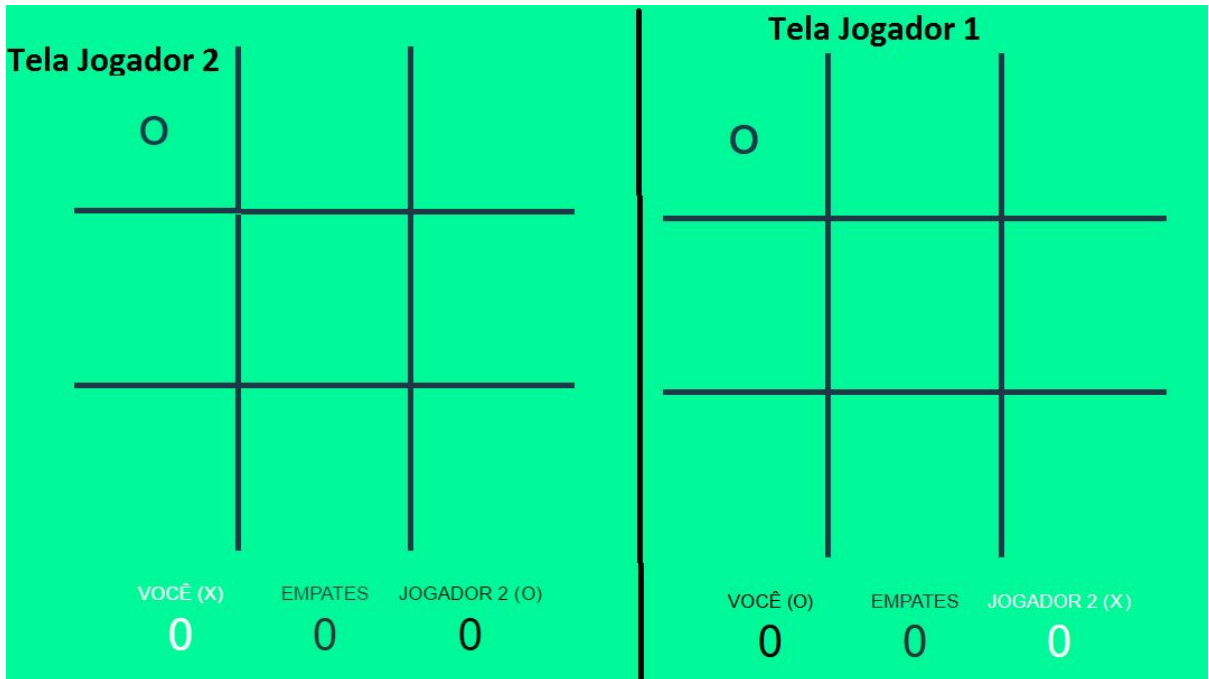
Após o segundo conectar é mostrado o jogo e seu board em branco.

Imagem 3 - Tela que aparece para jogadores, jogador 1 e jogador 2 com símbolos diferentes.



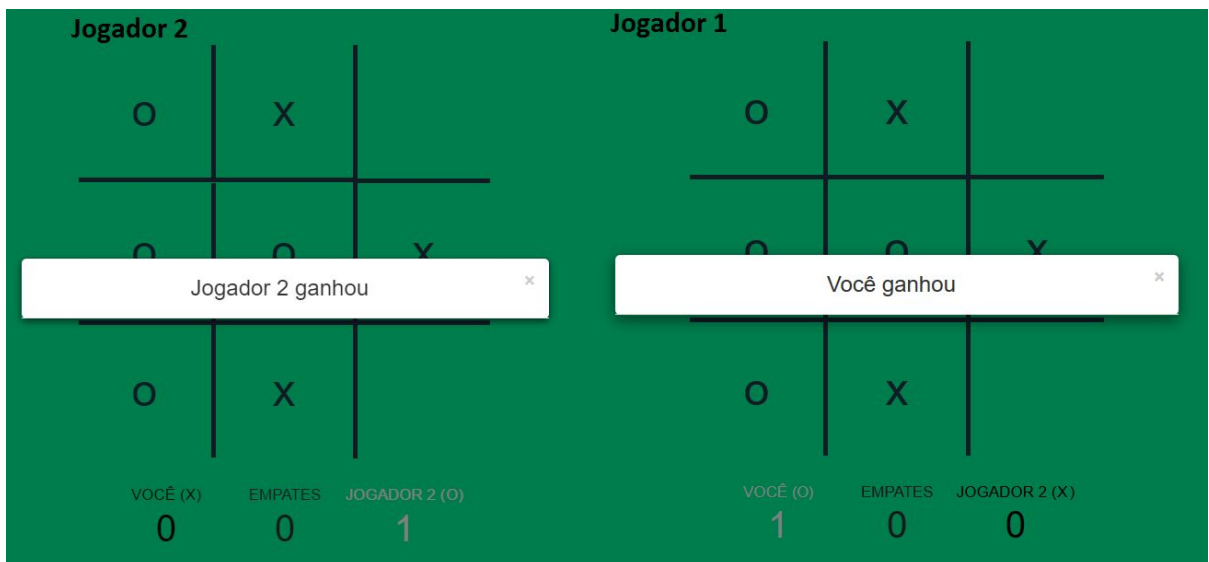
Após realizada uma jogada pelo jogador 1 é exibida a posição para o jogador instantaneamente:

Imagem 4 - Movimentação para ambos.



Após conclusos é exibida um modal caso houve ganhador, dependendo da jogada é exibida para ambos os lados mensagens diferentes.

Imagem 5 - telas quando ganha.



Conclusão

Para finalizar este projeto foi necessário vários dias de aprendizado sobre sockets e servidores de aplicação em *NodeJS*, foi escolhido esta ferramenta pois ao longo das leituras feitas sobre a ferramenta como citado por Gustavo Kerezi(2016) que cita uma opinião sobre a Nasa utiliza em seus sistemas para melhor escalabilidade, alta performance em suas aplicações Server-Side onde o cliente faz parte do processamento das informações.

Frente ao nosso curso de Bacharel em Sistemas de informações, aprender uma nova ferramenta e como utilizamos ao dia a dia, ajuda a nos integrar ao mercado de trabalho pelo menos com alguma introdução do que é, de como funciona e de qual utilidade podemos fazer com a ferramenta. Visto que poucas interagem deste modo de aprender e superar desafios de sair do mesmo código que aprendemos desde o começo do curso, buscamos diferenciar e aprender uma nova ferramenta.

Digamos que hoje nossa aplicação fosse escalável. Podemos pensar? Ela está num *framework Web NodeJS*. No caso para abrimos para o publico precisaríamos de um servidor que aceite o Node (Windows, Linux, Mac) disponíveis no site. Para executarmos a aplicação apenas navegadores com suporte a sockets.

Referências

Express. Disponível em: <<http://expressjs.com/pt-br/>>. 2016. Acesso em: 19 set. 2016.

G., Kerezi. (2016) **Por que a Nasa está usando NODE.JS?**. Disponível em: <<http://blog.geekhunter.com.br/por-que-a-nasa-esta-usando-node-js-e-por-que-voce-ta-mbem-deveria/>>. 2016. Acesso em: 20 set. 2016.

Nawara, D. (2016) **Express Generator and Socket.io**. Disponível em: <<https://github.com/onedesign/express-socketio-tutorial/>>. 2016. Acesso em: 19 set. 2016.

Socket.io. Disponível em: <<http://socket.io/docs/server-api/>>. 2016. Acesso em: 19 set. 2016.