Módulo 19 – Websocket – Atividade 02

Q1) Vamos programar um aplicativo de quiz ! Cada cliente pode criar perguntas (que só aceitam resposta “sim”/”não”), e todos podem votar, inclusive o mesmo cliente pode votar mais de uma vez numa mesma pergunta. Em detalhes, o funcionamento do aplicativo é o seguinte:

a. Quando um cliente acessa a página (frontend só terá 1 página), ele já estabelece uma conexão websocket ao servidor. Mas ainda não pode criar perguntas, votar em perguntas, e não recebe do servidor nenhuma pergunta criada por outros clientes. Na página, só é visível um input para preencher um `username` e um botão para “Entrar”.
b. Então o cliente preenche seu `username` num input e clica no botão "Entrar". Nesse momento é enviada uma mensagem websocket ao servidor, que só aceita a entrada caso o `username` não esteja sendo usado por outro cliente ainda. Caso a entrada seja válida, o servidor envia uma mensagem indicando isso, caso contrário envia uma mensagem de erro. Quando o cliente entende que a entrada foi válida, aparece no HTML mais um input e um botão (para preencher um texto de questão e clicar para criá-la). Já se o cliente entender que a entrada foi inválida, exibe num alert() uma mensagem de erro.
c. Logo após a entrada bem sucedida, o servidor envia ao cliente (em uma única mensagem) a lista de todas as perguntas já criadas no passado, que o cliente renderiza na tela. Mas as perguntas que forem criadas no futuro serão recebidas uma por mensagem (no momento da criação, ver abaixo).
d. A partir do momento da entrada, o cliente está ativo, poderá criar perguntas, votar em perguntas (suas ou de outros) e verá todas as perguntas que forem criadas por outros usuários a partir de então. Quando o cliente cria uma pergunta, o **servidor** (não o cliente) cria um ID único para essa pergunta, e então o servidor envia a todos os clientes uma notificação contendo a pergunta que foi criada. Quando o cliente vota em uma pergunta (pelo ID dela), o servidor envia a todos os clientes uma notificação contendo a pergunta que foi votada e a nova contagem de votos das respostas dela. Quando um cliente recebe notificação de que uma pergunta foi criada, ele renderiza a nova pergunta no HTML. Quando recebe notificação de que uma pergunta foi votada, atualiza no HTML a contagem de votos dela.
e. Quando um cliente sai da página (recarrega ou fecha o browser), o servidor libera o `username` dele. Então é possível que outro cliente entre com o mesmo nome (não há autorização, um `username` pode ser utilizado por qualquer cliente).

O anexo de aula chamado “código-base” tem um projeto pré-montado com HTML pronto e um esqueleto de javascript cliente e servidor. Você pode usá-lo se quiser (é opcional). O anexo “explicação do código-base” é um vídeo explicando como esse código está estruturado e como usá-lo.

Você **pode** usar bibliotecas de websocket, por exemplo socket.io, no cliente e/ou no servidor. Entregue seu projeto compactado (excluindo *node_modules*).

Referências

Exemplos completos de aplicações complexas com websocket:

- Jogo do par ou ímpar, com código executável e explicado: https://codesandbox.io/p/sandbox/websocket-cliente-servidor-exemplo-forked-7p4tfs
  - Talvez você precise clicar em “Fork” no canto superior direito do codesandbox para criar uma cópia individual do projeto (se houver outras pessoas experimentando ao mesmo tempo, caso no qual compartilhariam o mesmo servidor e pode ficar difícil testar)
- Playlist jogo de pegar o item (Filipe Deschamps, youtube): https://www.youtube.com/playlist?list=PLMdYygf53DP5SVQQrkKCVWDS0TwYLVitL
  - O Filipe mostra como usar “Design Patterns” (padrões de design) para que o código do jogo fique fácil de entender e alterar quando preciso (ou quando quiser adicionar funcionalidades novas), separado em componentes que são responsáveis por coisas distintas.
  - O vídeo que realmente começa a usar websocket é o “*Me veja programar um Backend que troca informação em tempo-real!*” , mas para entendê-lo talvez você precise dos anteriores (que é onde ele cria e explica o jogo funcionando só com frontend, antes de introduzir um backend)
- Playlist jogo de pong (canal “Seja um Programador”, youtube): https://www.youtube.com/playlist?list=PLIShO1rZkwrKiVQJIirbr5UthJT387iDl
  - É um “mini-curso” de várias coisas, inclui explicações muito básicas de node.js e browser antes de entrar no websocket. Vale pular as partes conhecidas.
  - Ele usa React no frontend !

Respostas

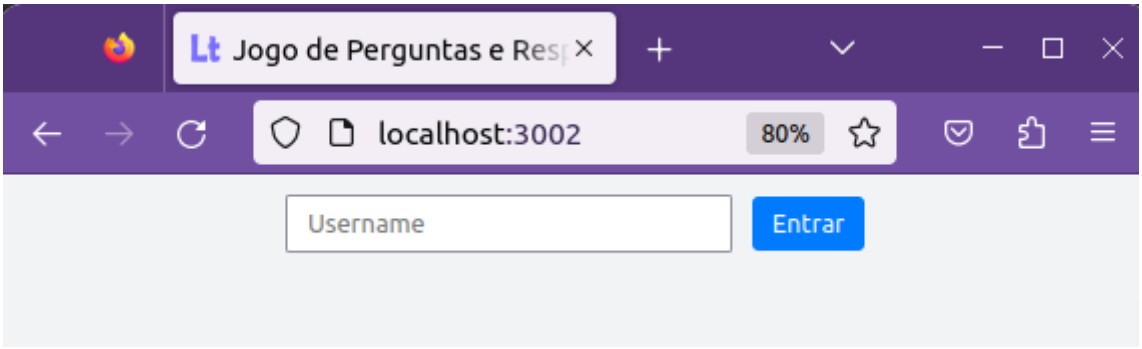
Q1)

Aplicando o comando “**npm run start**” para iniciar a aplicação (em modo de desenvolvimento é possível usar o nodemon via “**npm run dev**”).

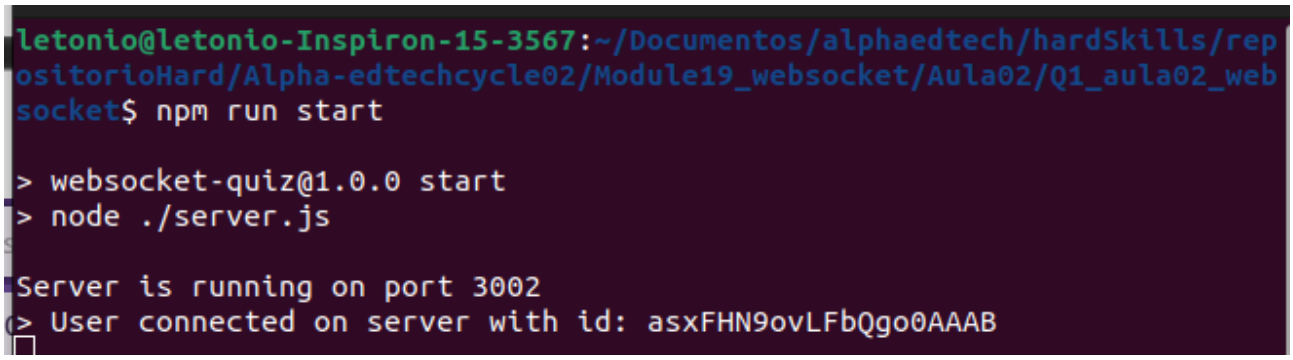
```
letonio@letonio-Inspiron-15-3567:~/Documentos/alphaedtech/hardSkills/repositorioHard/Alpha-edtechcycle02/Module19_websocket/Aula02/Q1_aula02_websocket$ npm run start

> websocket-quiz@1.0.0 start
> node ./server.js

Server is running on port 3002
```

Vamos estabelecer a primeira conexão, acessando o endereço:

http://localhost:3002

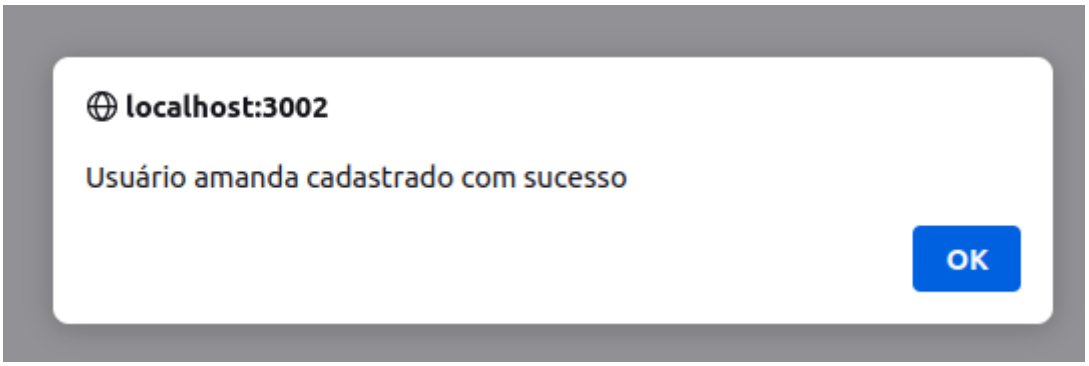


Ao acessar o endereço, aparece uma notificação no terminal, indicando que a conexão foi bem-sucedida e o id do socket do client:

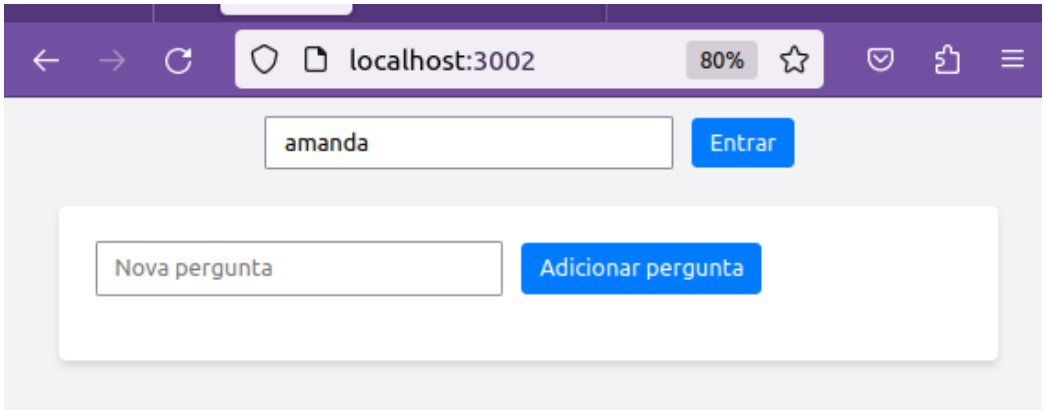


Mas, conforme especificado no enunciado da questão, não foi estabelecido nenhum nome/username para o usuário. Ele precisa inserir um username no campo para começar a usar.

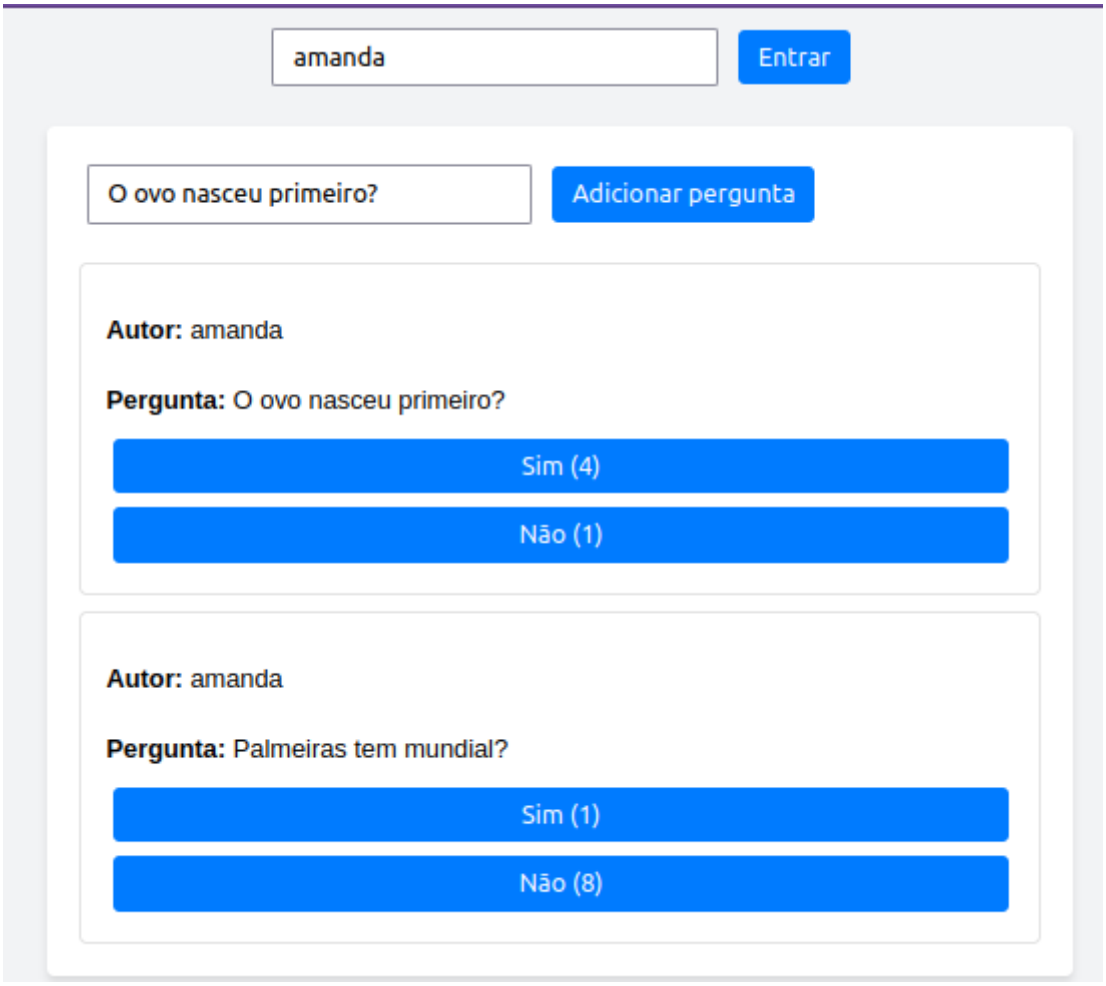
Para o primeiro usuário, empregamos o nome “amanda”. Ao clicar em Entrar, aparece uma mensagem de usuário cadastrado com sucesso.



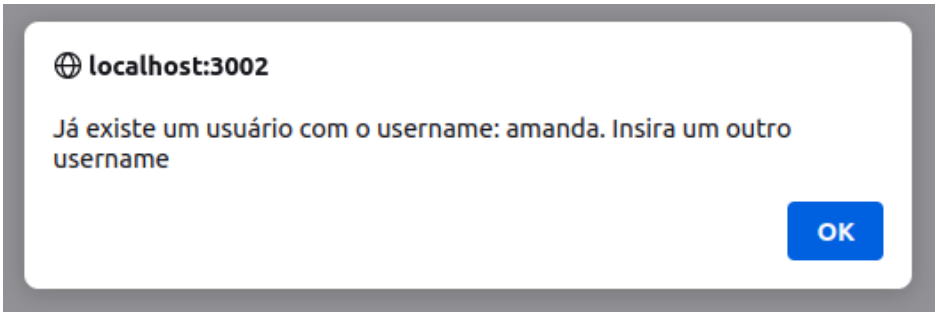
Clicando em “OK”, aparece o campo que permite inserir novas perguntas.



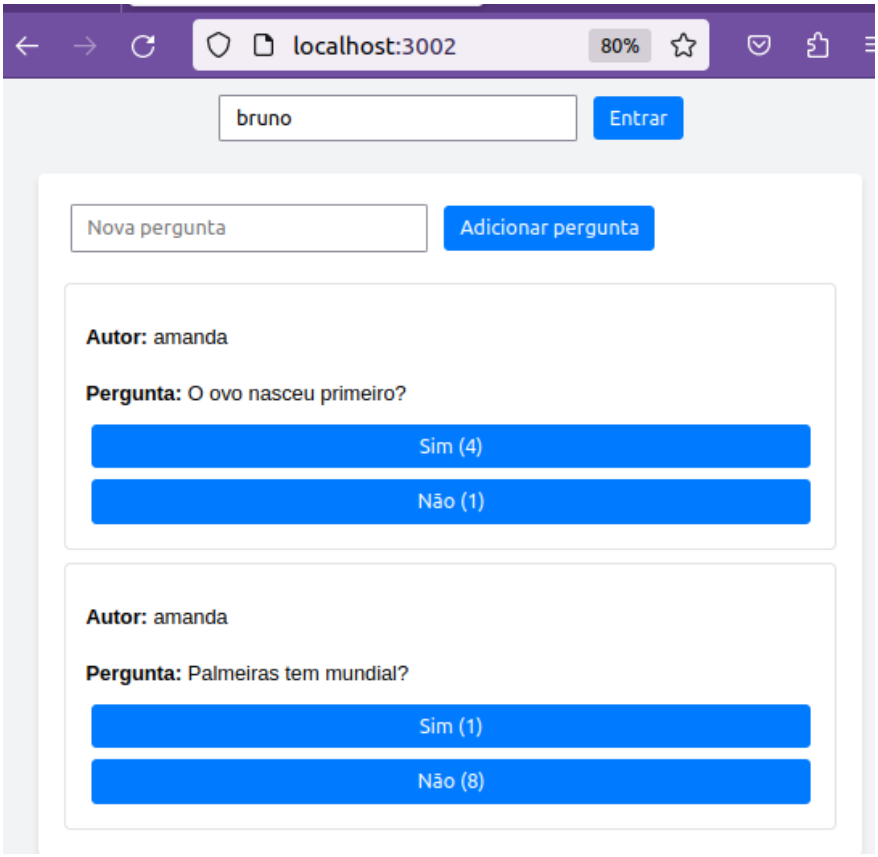
Vamos adicionar duas perguntas:



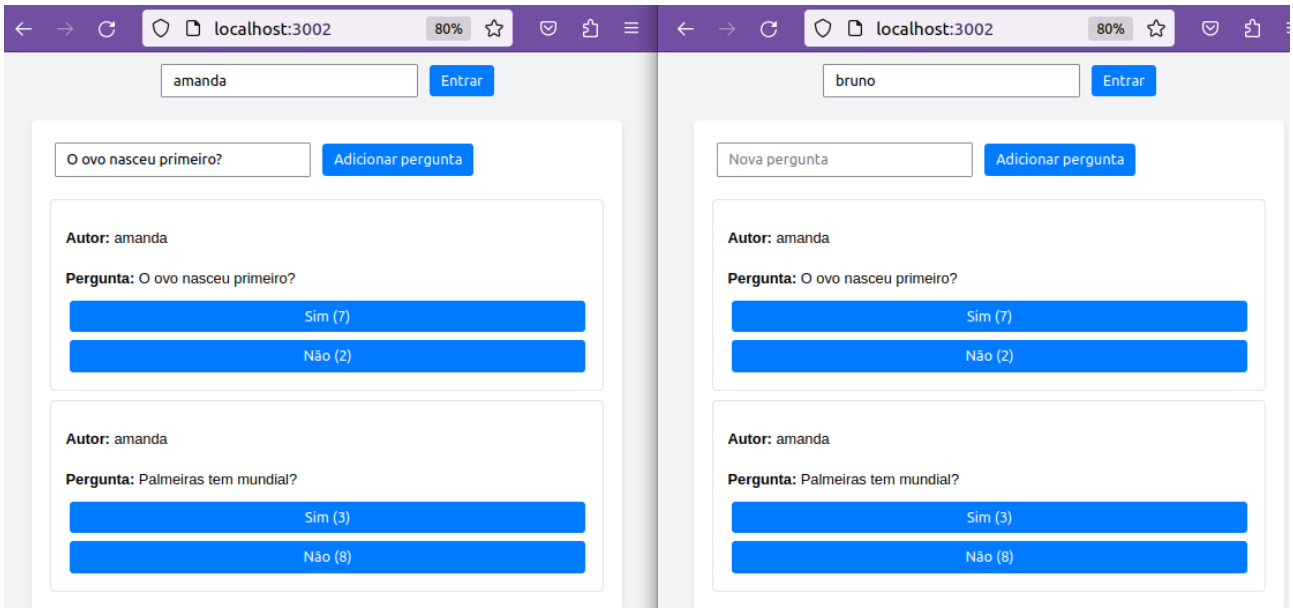
Abrindo outra aba do navegador e tentando inserir o mesmo nome de usuário (amanda), recebemos um alert explicando que já está sendo usado.



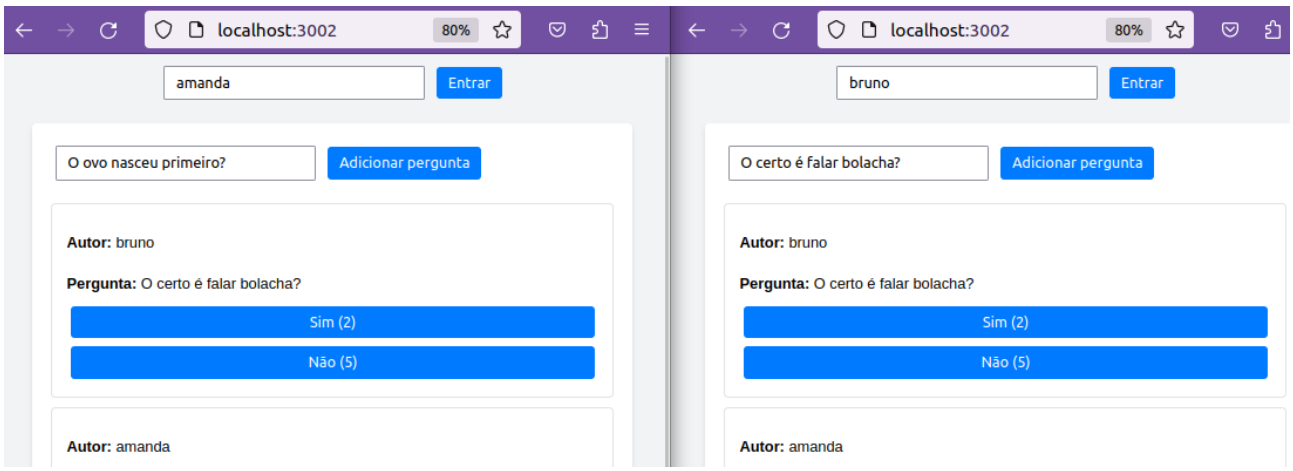
Para o segundo client, vamos adotar o username “bruno”. Fazendo isso e pressionando okay, recebemos o conteúdo com todas as enquetes anteriores e com a mesma numeração.



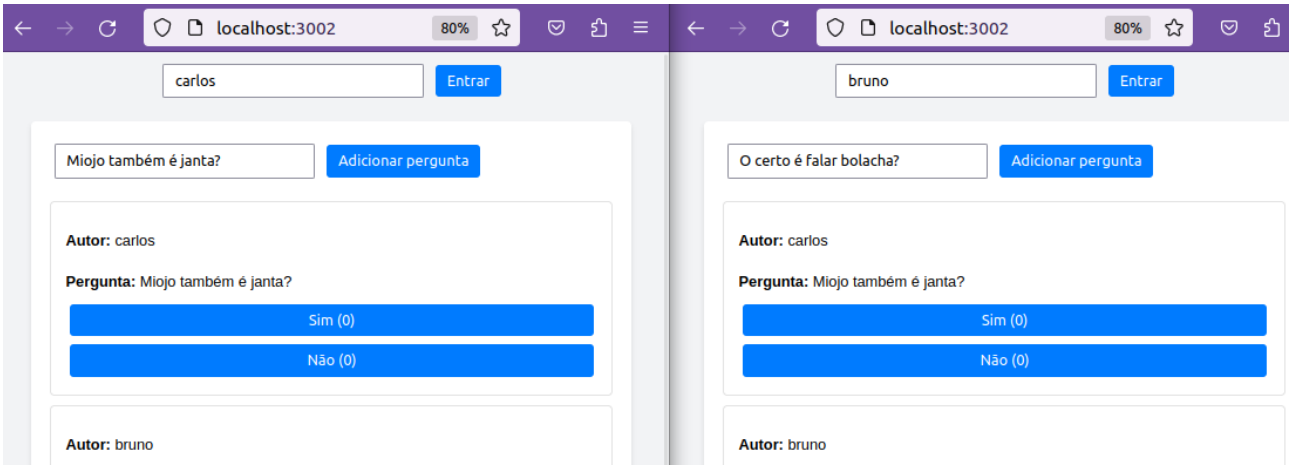
Está sincronizado, adicionar sim ou não em um dos navegadores automaticamente o outro fica atualizado também.



A partir do usuário “bruno”, vamos adicionar uma nova enquete:



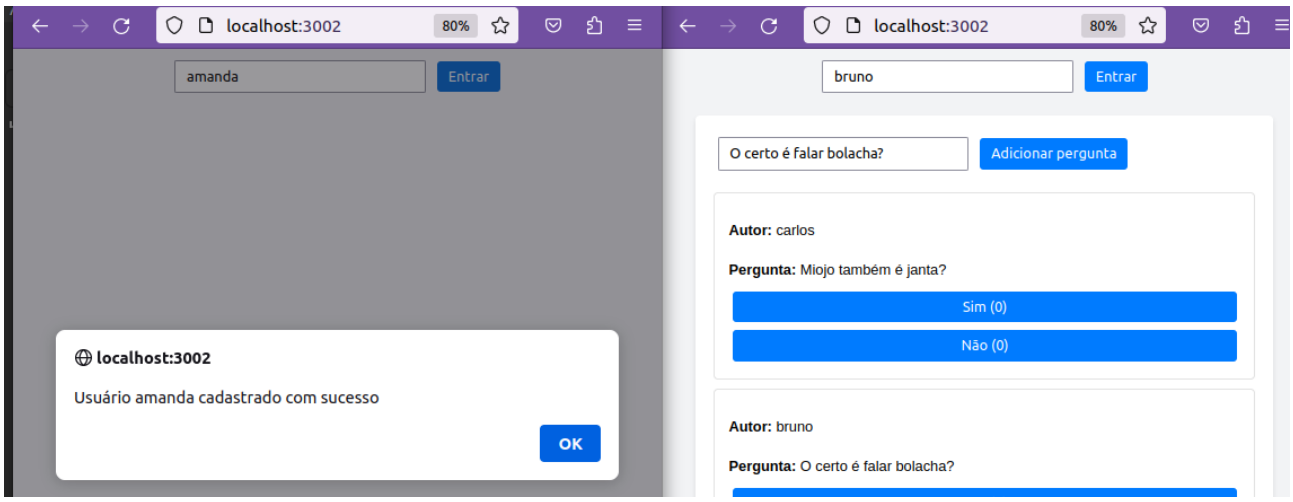
A próxima etapa é adicionar um terceiro usuário, que será chamado de “carlos”. Em seguida, vamos desconectar a usuária “amanda”. Dessa forma seu username ficará disponível para que outros possam utilizar.

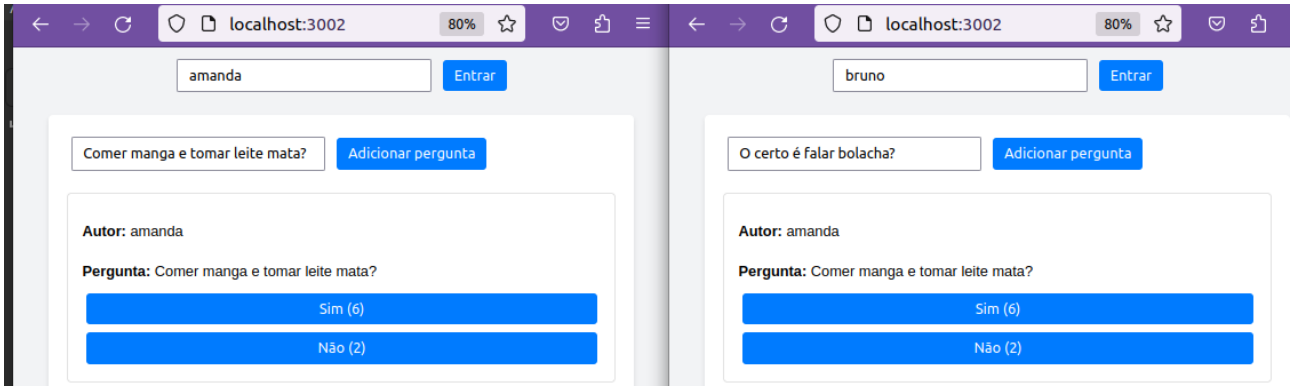


Fechando o navegador da “amanda”, aparece no terminal uma mensagem sobre a desconexão:

```
> Emitting remove-user
> User disconnected: amanda
```

Acessando outra aba, vamos tentar adicionar um usuário com o nome “amanda” e, dessa vez, conseguiremos adicionar com sucesso, pois o username não está mais sendo usado.





Com isso, finalizamos a demonstração da aplicação websocket.

```
> Emitting add-question
> Emitting remove-user
> User disconnected: amanda
> Emitting remove-user
> User disconnected: bruno
> Emitting remove-user
> User disconnected: carlos
```