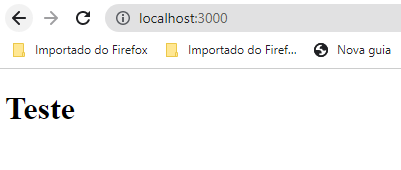
**UTILIZANDO O SOCKET.IO**

**CONFIGURAÇÃO DO PROJETO PARA O SOCKET IO**

1. Inicie o projeto npm na pasta do projeto
2. Digite npm init.
3. Instalando as bibliotecas.
4. Express: npm install express –save
5. Nodemon: npm install nodemon --save
6. Para as views vamos instalar a ejs: npm install ejs --save
7. Socket-IO: npm install socket.io –save
8. Crie o arquivo index.js
9. importe o express e crie a instancia da aplicação.
10. *var* express = require('express');
11. *var* app = express();
12. vamor montar um servidor http nativo do node, isso exige que agente mude a configuração original do express.
13. //colocando a aplicação express para rodar dentro do servidor http nativo do node
14. *var* http = require('http').createServer(app);
15. isso foi feito poqure que a aplicação express e a aplicação socket-io precisam estar rodando no mesmo servidor http. Para isso primeiro colocamos a aplicação express dentro do servidor http nativo do node.
16. Depois importamos o socket.io e passamos como parâmetro o servidor http que foi criado anteriormente.
17. *var* io = require('socket.io')(http);

**CONFIGURANDO A VIEW NO PROJETO**

1. iremos utilizar as views do ejs.
2. app.set('view engine','ejs');
3. vamos criar uma rota principal essa rota irá renderizar um arquivo html chamado index.ejs
4. esse arquivo será criado dentro da pasta views.
5. //criando rota principal que irá renderizar o arquvio index.ejs
6. app.get('/', (*req*, *res*) *=>* {
7. *res*.render('index')
8. })
9. No arquivo index.ejs vamos criar uma estrutura html e um h1 com uma mensagem
10. <!DOCTYPE html>
11. <html lang="en">
12. <head>
13. <meta charset="UTF-8">
14. <meta http-equiv="X-UA-Compatible" content="IE=edge">
15. <meta name="viewport" content="width=device-width, initial-scale=1.0">
16. <title>Document</title>
17. </head>
18. <body>
19. <h1>Teste</h1>
20. </body>
21. </html>
22. Agora vamos configurar a aplicação para rodar utilizando o servidor nativo que criamos
23. http.listen(3000, () *=>* {
24. console.log("Servidor rodando");
25. })
26. Colocando a aplicação para rodar, no terminal na pasta do projeto digite: nodemon index.js
27. Agora digite no navegador localhost:3000 e aparecerá a mensagem que criamos na view index.ejs
28. 

**Como Iniciar com o socket.io**

1. A primeira coisa que temos que fazer é abrir um evento de conexão, pois os web sockets são baseados em eventos.
2. Esse evento é um padrão do socket.io e é chamado sermpre que um cliente se conecta a sua aplicação web socket.
3. //evento de conexão do socket.io
4. io.on("connection",(*clienteBackEnd*) *=>* {
5. console.log(*clienteBackEnd*);
6. console.log(*clienteBackEnd*.id)
7. })
8. Para testar esse cliente temos que ir no fron end no arquivo index.ejs e importar o script front end do socket.io
9. <script src="">
10. </script>
11. O socket.io possui tanto a biblioteca para back end como para front end. E são utilizadas para comunicar o backend com o frontend em tempo real.
12. Para isso podemos ir no goolgle e pesquisar socket.io cdn.
13. 
14. Vamos copiar o link e colar no src do script criado no arquivo index.ejs.
15. <script src="https://cdn.socket.io/4.4.1/socket.io.min.js" integrity="sha384-fKnu0iswBIqkjxrhQCTZ7qlLHOFEgNkRmK2vaO/LbTZSXdJfAu6ewRBdwHPhBo/H" crossorigin="anonymous"></script>"></script>
16. Vamos criar uma variável em um script abaixo do link que importamos.
17. Essa variável irá receber o io() com o endereço do servidor que criamos e está rodando o socket.io
18. <script>
19. *var* clienteFrontEnd = io("http://localhost:3000");
20. </script>
21. Fazendo isso estaremos conectando o socket do frontend com o backend.
22. Vamos rodar novamente a aplicação e irá aparecer a mensagem do backend quando recarregarmos a página no front end.