Configuração de Servidor e comunicação

|  |  |  |
| --- | --- | --- |
| **Nome dos módulos/ arquivos** | **Conceito do modulo/ arquivo** | **Principais funções** |
| **Server.ts** | **Responsável por iniciar o servidor http da aplicação.** | **HttpServer.listen(port)**  **Obs: esse port pode ser o link do host de hospedagem ou porta onde o servidor será iniciado no localHost** |
| **HttpServer.ts** | **Responsável por configurar as rotas da aplicação e middlewares usados, além de criar o servidor Http** | **http.CreateServer()**  **new Server(httpServer)** |
| **WebSocketServer.ts** | **Responsável por criar os servidores websockets para criação de salas de bate papos além de fazer a troca de mensagens entre usuários** | **IO.on(socket)** |

Criação de rotas e configuração das mesmas

|  |  |  |
| --- | --- | --- |
| **Nome dos módulos/ arquivos** | **Conceito do modulo/ arquivo** | **Funções do método http recebido** |
| RouteCadastro.ts | Responsável pelo acesso a pagina de cadastro e também por criar usuários | get-> permite o front end acessar e carregar a página de cadastro além de autentificar o usuário.  post-> recebe os dados dos novos usuários e salva manda para o banco de dados |
| RouteChat.ts | Renderiza a página da sala que usuários selecionou | get-> autentica o usuário e permite ele entrar na sala selecionada |
| RouteCreateRoom.ts | Permite que usuários administradores criem salas de bate papo | get-> autentifica o usuário e renderiza a pagina de criação de sala  post-> verifica se o nome que o usuário digitou para criar a sala já existe e se não existir criar a sala e salva os dados da sala no DB. |
| RouteEditProfile.ts | Permite o usuário alterarem o seu username e senha | Get-> autentifica o usuário e renderiza a página de editar perfil  Post-> envia os dados que usuário digitou e verifica a existência do username do usuário e se não existir, ele altera a senha e o username do usuario |
| RouteLogin.ts | Permite usuários logarem no sistema | Get()-> Renderiza a página de login  Post-> verifica a senha e username do usuário e caso esteja tudo certo cria um token que será usado na autentificarão do usuário. |
| RouteLogin.ts | Desloga o usuário do sistema | Post-> apaga o token criado para o usuário e redireciona ele para página de login |
| RouteRoomsManager.ts | Permite o usuário administrador gerenciar as salas criado por ele, como por exemplo alterar o nome da sala, a sua descrição e também dá acesso a outros usuários a sala | Get-> autentifica o usuário e renderiza a pagina de gerenciamento de salas.  Post-> recebe as alterações feita pelo usuário administrador e salva no DB. |
| RouteSocialArea.ts | É responsável por mostrar a salas que o usuário pode acessar e também para direciona-lo para essas salas quando ele quer entrar nela | Get-> autentifica o usuário e renderiza a página de área social  Post-> envia o usuário para sala selecionada |
| RouteUploader.ts | Rota responsável por receber os arquivos enviados via chat pelos usuários e salva-las localmente no servidor para em seguida enviar para nuvem | Post-> autentifica o usuário e salva os arquivos |
| RouteUsersManager.ts | Realiza o gerenciamento de usuários, permitindo exclui-los do sistema para não ter mais acesso ao sistema | Get-> autentifica o usuário e renderiza a página de gerenciamento do usuário  Post-> Recebe os dados do usuário que o administrador selecionou e apaga. |

Tipos criados para facilitar manejo do código

|  |  |
| --- | --- |
| Nome do type | campos |
| TAlterRoom.ts | oldName?:string,  newName?:string,  description:string |
| TMessage.ts | room:string,  date:Date,  messages:string,  userName:string,  url\_file?:string,  nameFile?:string,  setor:string,  cargo: string,  nomeFuncionario: string |
| TRoomUser.ts | socketId: string,  userName: string,  room: string |
| TUserCreateRoom.ts | nameRoom: string,  userName: string,  descriptionRoom: string |
| TUserData.ts | id?: number;  userName:string,  password?:string,  nomeFuncionario: string,  setor: string,  cargo: string,  isadm: number,  isdeleted?: number |
| Obs: campos que contem “?” indicam que eles são opcionais | |

Funções realizadas no banco de dados pelo modulo/arquivo DataBaseConnection.ts

|  |  |
| --- | --- |
| Nome da função | Tarefa realizada |
| getUser(user:string) | Recebe um username e retorna ele do DB caso ele esteja cadastro |
| getUsersNotAdm() | Pega todos os usuários que não são administradores do sistema |
| updateUser(user: TUserData) | Recebe os dados de um usuário e atualiza no DB |
| desableUser(username:string) | Recebe um username e desativa o acesso dele ao sistema |
| verifyUSer(userName: string) | Verifica se um usuário está cadastrado no sistema |
| registerUser(user:TUserData) | Insere um novo usuário no DB |
| getUserNames() | Pega todos os usernames de usuários do sistema |
| saveMesages(message:TMessage) | Salva as mensagens enviadas nos chats pelo usuários no DB |
| createRoom(data:TUserCreateRoom) | Cria novas salas no DB |
| getRooms() | Seleciona todos os dados sobre as salas criadas |
| getRoomsCreatedBy(username:string) | Seleciona as salas criadas por um usuário administrador especifico |
| veriflyUserIsMemberRoom(username:  string,room:string) | Verifica se um usuário específico faz parte de uma sala |
| getAllUsersMember() | Pega todos os usuários cadastrados nas salas de bate-papo |
| addUsersMember(username:string,  room:string) | Adiciona um usuário a uma sala |
| delMemberRoom(username:string,  room:string | Apaga um usuário de uma sala especifica |
| delMemberAllRoom(username:string) | Apaga um usuário de todas as salas que ele está cadastrado |
| alterNameRoom(alterRoom:TAlterRoom) | Altera o nome de uma sala |
| alterDescriptionRoom(room:string,  newDescription:string) | Altera a descrição de uma sala |
| deleteRoom(room:string) | Deleta uma sala do sistema |

Views(paginas do sistemas)

|  |  |
| --- | --- |
| Nome do arquivo da pagina | Função da pagina |
| lateralMenuHead.ejs | Permite inserir um elemento Head em varias paginas |
| LateralMenu.ejs | Menu usados em varias paginas para criar um padrão de menu |
| lateralMenuJs.ejs | Permite inserir referência do Javascript usado pelo menu |
| Cadastro.ejs | Pagina de cadastro |
| Chat.ejs | Pagina padrão para dos os chats criados |
| createRoom.ejs | Pagina usada para criar salas |
| editProfile.ejs | Pagina para edição de perfil dos usuarios |
| Index.ejs | Pagina de login |
| RoomsManager.ejs | Pagina de gerenciamento de salas |
| socialArea.ejs | Pagina de navegação dos usuarios para salas de bate papo |
| usersManager.ejs | Pagina de gerenciamento de usuarios. |

JavaScrpth usado no front-end

|  |  |  |
| --- | --- | --- |
| Nome do Arquivo | Atribuição | Principais funções |
| clientWS.js | responsável por enviar e receber as mensagens e arquivos enviados pelo servidor websocket | Socket.emit-> enviar as mensagens  Socket.on->recebe as mensagens  captureEnter-> captura a mensagem e arquivo do usuário para depois ser enviada para o websocket  createMessage-> cria a mensagem do usuário e as que são recebidas de outros usuários na tela |
| createRoom.js | Executa um regex para que o usuário não coloque um nome invalido na sala que vai criar |  |
| roomsManager.js | Permite mostrar os usuários dos sistemas para depois serem selecionados pelo administrator dar acesso aos mesmo a sala |  |
| socialArea.js | Responsável por mandar para o servidor a sala que usuário selecionou | clickButtonAction-> captura algum botão que usuário clicou para então enviar uma ação para o servidor |