Função: Converter vetor $1xN$ em matriz $Nx9$

Descrição: Converte um vetor $1xN$ em uma matriz $Nx9$. A última linha da matriz poderá ser truncada no caso em que N não for um múltiplo de 9.

Parâmetros:

vetor - Vetor heterogêneo $1xN$

Valor retornado

Matriz $Nx9$ contendo os valores do vetor com a última linha truncada no caso de N não ser um múltiplo de 9.

Assertiva de entrada

$N \in \aleph: N > 0$

Assertiva de saída

Se $N > 0$

Então

O retorno será uma matriz $Nx9$, cujas linhas derivam do vetor de entrada seccionado em vetores $1xM_i$, $1 \leq i \leq N$, tal que

$i < N \Rightarrow M = 9 \wedge i = N \Rightarrow 1 \leq M \leq 9$.

Senão

O retorno será um vetor vazio.

Função: Validar senha e confirmar senha.

Descrição:

Valida a entrada de um usuário em campos reservados para senha e confirmação de senha em formulários.

Parâmetros:

$password$

String referente à entrada em um campo para senha.

$confirm$

Booleano indicando a validação conjunta dos campos para senhas e confirmação de senhas.

$repeat$

String referente à entrada em um campo para confirmação da senha.

Valor retornado:

Booleano indicando o status da validação (*false* para inválido, *true* para válido)

Assertiva de entrada:

$typeof\ password\ =\ string$

$typeof\ confirm\ =\ boolean$

$typeof\ repeat\ =\ string$

(*typeof* é uma palavra-chave em Javascript que retorna o tipo de dado de uma variável)

Assertiva de saída:

Se $confirm$

Então

Retorna $password\ \neq\ ''\ \wedge\ repeat\ \neq\ ''\ \wedge\ password\ =\ repeat$

Senão

Retorna $password\ \neq\ ''$

(' ' denota uma *string* vazia)

Função: Jwt Guard

Descrição: A função JwtGuard é responsável por verificar e validar um token de autorização JWT em uma requisição HTTP.

Parâmetros:

enviado = Objeto representando a requisição HTTP recebida.

resposta = Objeto representando a resposta HTTP a ser enviada.

next = Função de callback a ser chamada quando a validação do token for bem-sucedida.

Valor retornado:

A função não retorna um valor diretamente, mas pode enviar uma resposta HTTP com diferentes códigos de status e mensagens dependendo do resultado da validação do token.

Assertiva de entrada:

enviado: Deve ser um objeto contendo informações sobre a requisição HTTP, incluindo o cabeçalho authorization que deve conter o token JWT no formato "Bearer <token>".

resposta: Deve ser um objeto usado para enviar respostas HTTP.

next: Deve ser uma função de callback a ser chamada após a validação do token.

Assertiva de saída:

Se o token JWT estiver presente e for válido, o código de status retornado será 200 ou outro código de sucesso configurado na função next(), e a mensagem será indicando sucesso na validação.

Se não houver token no cabeçalho authorization ou se o token for inválido, a função retornará um código de status 401 (Não Autorizado) com uma mensagem de erro indicando o motivo da falha na validação do token.

Função: Cadastrar uma loja

Descrição:

Cadastra uma nova Loja no banco de dados se o email já não estiver cadastrado em outra loja.

Parâmetros:

$email$

String referente à entrada em um campo para email.

$nome$

String referente à entrada em um campo para nome.

$senha$

String referente à entrada em um campo para senha.

$imagem$

Rota para imagem

$longitudeFixa$

String referente à longitude obtida através de outra função.

$latitudeFixa$

String referente à latitude obtida através de outra função.

Valor retornado:

Object do erro se houver um lançamento de exceção.

Assertiva de entrada:

Assertivas são realizadas através do front-end.

Assertiva de saída:

Se o email já estiver cadastrado em outra loja, chama outra função para renderizar o erro no front-end.

Função: Enviar Localização Para o Backend

Descrição: Realiza o envio das coordenadas de localização obtidas pelo navegador para o backend da aplicação.

Parâmetros:

Coordenadas: Objeto contendo as coordenadas de latitude e longitude do usuário.

Valor retornado:

Promessa (Promise) indicando o sucesso ou falha do envio.

Assertiva de entrada:

Coordenadas deve ser um objeto não nulo contendo as propriedades "latitude" e "longitude".

Assertiva de saída:

Se *coordenadas são fornecidas*

Então

A função realiza uma requisição assíncrona (como uma Promise) para o backend, enviando as coordenadas do usuário.

Senão

O retorno será uma Promise rejeitada indicando que as coordenadas não foram fornecidas.

Função: Atualizar Localização Inicial do Usuário

Descrição: Atualiza a localização inicial do usuário no banco de dados com base nas coordenadas fornecidas.

Parâmetros:

'req' - Objeto de requisição contendo dados do usuário.

'res' - Objeto de resposta utilizado para enviar a resposta da requisição.

Valor retornado:

Nenhum valor retornado diretamente, mas a resposta da requisição é enviada para o cliente.

Assertiva de entrada:

‘userId’ deve ser fornecido no corpo da requisição (req.body.userId).

Assertiva de saída:

Localiza a latitude e longitude do usuário

Então

Atualiza a localização do usuário no banco de dados usando o Prisma.

Senão

Em caso de erro, loga o erro e envia uma resposta de erro ao cliente.

Função: Validar nome

Descrição:

Valida um nome de acordo com os critérios especificados.

Parâmetros:

name

String referente ao nome a ser validado.

Valor retornado:

Retorna verdadeiro se o name for uma string não vazia e seu comprimento for menor ou igual a 50 caracteres.

Assertiva de entrada:

typeof name = 'string'

Assertiva de saída:

A função retorna um valor booleano (verdadeiro ou falso) de acordo com os critérios especificados.

Função: Retornar loja

Descrição:

Retorna informações de uma loja específicas com base no ID fornecido.

Parâmetros:

id

ID único da loja a ser recuperada.

Valor retornado:

Retorna um objeto contendo informações da loja.

Exceção lançada em caso de erro durante a busca.

Assertiva de entrada:

typeof id === 'number'

Assertiva de saída:

A função retorna um objeto que contém as informações da loja.

Função: Cadastrar produto

Descrição:

Cria um novo registro na tabela de produtos no banco de dados, vinculando-o a uma loja.

Parâmetros:

$nome$

String referente ao nome do produto a ser cadastrado.

$descrição$

String referente à descrição do produto a ser cadastrado.

$imagem$

String referente ao caminho da imagem vinculado ao produto a ser cadastrado.

$idLoja$

Número Inteiro referente ao identificador (ID) único da loja a ser vinculada ao produto a ser cadastrado.

Valor retornado:

Objeto contendo as informações do produto criado ou, em caso de lançamento de exceção, *erro* encontrado durante a criação do registro.

Assertiva de entrada:

D: $Tabela - Loja \mid idLoja \in D$

Assertiva de saída:

Se $idLoja \in D$

Então

O retorno será um objeto (instância) do novo produto criado.

Senão

O retorno será um erro.

Função: Excluir produto

Descrição:

Remove um registro específico de produto vinculado a uma loja identificada no bando de dados.

Parâmetros:

$nome$

String referente ao nome do produto a ser cadastrado.

$idLoja$

Número Inteiro referente ao identificador (ID) único da loja a ser vinculada ao produto a ser excluído.

$idProduto$

Número Inteiro referente ao identificador (ID) único do produto a ser excluído.

Valor retornado:

Confirmação da exclusão do produto ou, em caso de erro, detalhes do erro encontrado durante a tentativa de exclusão.

Assertiva de entrada:

D: $Tabela - Produto \mid idProduto \in D$

D: $Tabela - Loja \mid idLoja \in D$

R: $Relação\ Produto - Loja \mid (idLoja, idProduto) \in R$

Assertiva de saída:

Se $idLoja \in D$, idProduto $\in D$, e (idProduto,idLoja) $\in R$

Então

O produto correspondente será excluído do banco de dados.

Senão

O retorno será um erro.