Testes candidatos a automação

Testes Candidatos à Automação	Descrição	Sugestões de Testes no Postman e Passo a Passo
Login com sucesso e falha	Testar o login com credenciais válidas e inválidas. Garantir que a resposta do sistema seja válida, com retorno correto de status e mensagem (código 200 para sucesso e 401 para falha).	**Sugestão de Automação:** 1. Crie uma requisição POST para a rota `/login` com dados de login válidos e inválidos. 2. Verifique o status code esperado (200 para sucesso e 401 para falha). 3. Utilize scripts no Postman para validar as mensagens de erro (ex.: `pm.response.to.have.status(200)` e `pm.response.to.have.status(401)`).
Cadastro de usuário com dados válidos e inválidos	Testar o cadastro de usuários com dados válidos, como e-mail válido e senha dentro dos critérios. Testar o cadastro com dados inválidos, como e-mail duplicado, senha fraca ou campos obrigatórios faltando. Validar as respostas do sistema, garantindo o comportamento esperado para cada cenário (códigos 200, 400, 409, etc.).	**Sugestão de Automação:** 1. Crie requisições POST para a rota `/users` com dados válidos e inválidos (como e-mail duplicado). 2. Para dados válidos, verifique o status 200 e o conteúdo da resposta. 3. Para dados inválidos (e.g., e-mail já registrado), valide o status 400 ou 409 e mensagens de erro usando `pm.expect(pm.response.json().message).to.eql('E-mail já cadastrado')`.
Cadastro e edição de produtos (autenticado)	Testar a adição e edição de produtos para usuários autenticados, incluindo: Produtos com dados válidos (nome, preço, descrição). Produtos com dados inválidos (como valores de preço negativos ou campos obrigatórios ausentes). Verificar que as mudanças sejam refletidas corretamente no banco de dados e no sistema.	**Sugestão de Automação:** 1. Crie requisições POST e PUT para a rota `/products`, uma para adicionar um produto e outra para editar. 2. Para dados válidos, verifique o status 201 e 200, respectivamente. 3. Para dados inválidos (e.g., preço negativo), valide o status 400 e a mensagem de erro adequada. 4. Utilize o `pm.expect()` para garantir que os produtos sejam corretamente criados e atualizados, validando o banco de dados se possível.
Adição, remoção e atualização de produtos no carrinho	Testar cenários relacionados ao carrinho, com foco na adicionar, remover e atualizar produtos:	**Sugestão de Automação:** 1. Crie requisições POST para a rota `/cart` para adicionar produtos, PUT

	• Testar a adição de produtos ao carrinho com dados válidos. • Testar a atualização da quantidade de um produto no carrinho. • Testar a remoção de produtos do carrinho (garantindo que o produto seja removido corretamente, e o carrinho seja atualizado). Validar que as respostas de status e mensagens estão corretas (200, 400, 404, etc.).	para atualizar e DELETE para remover. 2. Teste a adição de um produto, validando o status 200 e a resposta de sucesso. 3. Para atualização de quantidade, valide o status 200 e a nova quantidade. 4. Para remoção de produto, verifique se o produto foi removido corretamente e o status 200 ou 404 se o produto não existir.
Validações de resposta (status code, mensagens) via aba "Tests" do Postman	Automatizar validações de código de status (200, 400, 404, etc.) e mensagens de resposta para as rotas que envolvem carrinho e produtos. Verificar que as mensagens sejam claras e informativas, como: "Produto adicionado ao carrinho com sucesso", "Produto não encontrado", "Quantidade inválida", entre outras.	**Sugestão de Automação:** 1. Em todas as requisições do Postman, adicione validações na aba "Tests". 2. Verifique se o código de status é o esperado (ex.: `pm.response.to.have.status(200)` ou `pm.response.to.have.status(400)`). 3. Use o `pm.response.to.have.jsonBody()` para validar o conteúdo das mensagens e garantir que elas sejam claras e adequadas.
Geração de token e tempo de validade (extra com script)	Automatizar a geração de token para autenticação e validar seu tempo de validade. Garantir que o token seja validado corretamente para ações que envolvem produtos no carrinho e outros recursos que exigem autenticação. Validar cenários em que o token expira ou é inválido, retornando o código e mensagem adequados (401 - "Token inválido ou expirado").	**Sugestão de Automação:** 1. Crie uma requisição POST para a rota `/login` para gerar um token. 2. Armazene o token usando `pm.environment.set('authToken', pm.response.json().token)` para reutilizá-lo em outras requisições. 3. Verifique a expiração do token, criando um teste para que o sistema retorne o status 401 se o token estiver expirado. 4. Teste a resposta com um token inválido, verificando o código de erro e a mensagem.