**Cenários de Teste para Cadastro de Usuário**

Cenário de Teste 1: Cadastro de Usuário com Sucesso

**Objetivo**: Garantir que um novo usuário pode ser registrado com sucesso quando todos os campos obrigatórios são fornecidos corretamente.

**Pré-condições**:

* Nenhum usuário existente com o mesmo email ou nome de usuário.

**Passos**:

1. Execute uma solicitação POST para o endpoint /register\_user com o seguinte payload JSON:
2. {
3. "first\_name": "João",
4. "last\_name": "Silva",
5. "email": "joao.silva@example.com",
6. "username": "joao\_silva",
7. "password": "senha\_segura"
8. }

Copiar

**Resultados Esperados**:

* O status de resposta deve ser 201 (Criado).
* O JSON de resposta deve incluir uma mensagem de sucesso: {"message": "Usuário cadastrado com sucesso!"}.
* Um novo registro de usuário deve ser criado no banco de dados com os detalhes acima.

Cenário de Teste 2: Tentativa de Cadastro com Nome de Usuário Duplicado

**Objetivo**: Garantir que a tentativa de cadastro falha se o nome de usuário já estiver registrado no banco de dados.

**Pré-condições**:

* Um usuário já existe com o nome de usuário "joao\_silva".

**Passos**:

1. Execute uma solicitação POST para o endpoint /register\_user com payload JSON onde apenas o email é diferente:
2. {
3. "first\_name": "Maria",
4. "last\_name": "Oliveira",
5. "email": "maria.oliveira@example.com",
6. "username": "joao\_silva",
7. "password": "outra\_senha"
8. }

Copiar

**Resultados Esperados**:

* O status de resposta deve ser 400 (Bad Request).
* O JSON de resposta deve conter uma mensagem de erro: {"message": "Username já existe!"}.

Cenário de Teste 3: Tentativa de Cadastro com Email Duplicado

**Objetivo**: Garantir que a tentativa de cadastro falha se o email já estiver registrado no banco de dados.

**Pré-condições**:

* Um usuário já existe com o email "joao.silva@example.com".

**Passos**:

1. Execute uma solicitação POST para o endpoint /register\_user com payload JSON onde apenas o nome de usuário é diferente:
2. {
3. "first\_name": "Carlos",
4. "last\_name": "Pereira",
5. "email": "joao.silva@example.com",
6. "username": "carlos\_pereira",
7. "password": "senha\_diferente"
8. }

Copiar

**Resultados Esperados**:

* O status de resposta deve ser 400 (Bad Request).
* O JSON de resposta deve conter uma mensagem de erro: {"message": "Email já cadastrado!"}.

Cenário de Teste 4: Campos Obrigatórios Faltando no Cadastro

**Objetivo**: Validar que o cadastro falha se quaisquer campos obrigatórios estiverem ausentes na solicitação.

**Pré-condições**:

* Sem pré-condições.

**Passos**:

* Execute várias solicitações POST para o endpoint /register\_user omitindo diferentes campos obrigatórios, por exemplo, email:
* {
* "first\_name": "Paulo",
* "last\_name": "Santos",
* "username": "paulo\_santos",
* "password": "senha\_secreta"
* }

Copiar

**Resultados Esperados**:

* O status de resposta deve ser 400 (Bad Request).
* O JSON de resposta deve conter uma mensagem de erro indicando o campo faltando, como: {"message": "<nome do campo> é obrigatório!"}.

**Execução MANUAL**

Pré-requisitos

1. **Postman**: Certifique-se de que o Postman esteja instalado em seu computador.
2. **Serviço Flask**: Certifique-se de que o aplicativo Flask está em execução, com o banco de dados corretamente configurado.

Passo a Passo para Excluir Cenários de Teste no Postman

Cenário de Teste 1: Cadastro de Usuário com Sucesso

**Objetivo**: Validar que um novo usuário pode ser cadastrado com sucesso quando todos os campos obrigatórios são fornecidos.

**Passos**:

1. Abra o Postman.
2. Clique em "New" para criar uma nova requisição.
3. Escolha o método **POST**.
4. Insira a URL da sua API local http://localhost:<porta>/register\_user.
5. No corpo da requisição, escolha **raw** e **JSON** como tipo de conteúdo. Insira o seguinte conteúdo:
6. {
7. "first\_name": "João",
8. "last\_name": "Silva",
9. "email": "joao.silva@example.com",
10. "username": "joao\_silva",
11. "password": "senha\_segura"
12. }

Copiar

1. Clique em "Send" para enviar a requisição.

**Resultados Esperados**:

* A resposta deve ter um status **201 Created**.
* A resposta deve conter: {"message": "Usuário cadastrado com sucesso!"}.

Cenário de Teste 2: Cadastro com Nome de Usuário Duplicado

**Objetivo**: Validar que a tentativa de cadastro falha se o nome de usuário já estiver registrado.

**Passos**:

1. Repita os passos de 1 a 4 do Cenário de Teste 1.
2. No corpo da requisição, use o seguinte conteúdo JSON:
3. {
4. "first\_name": "Maria",
5. "last\_name": "Oliveira",
6. "email": "maria.oliveira@example.com",
7. "username": "joao\_silva",
8. "password": "outra\_senha"
9. }

Copiar

1. Clique em "Send".

**Resultados Esperados**:

* A resposta deve ter um status **400 Bad Request**.
* A resposta deve conter: {"message": "Username já existe!"}.

Cenário de Teste 3: Cadastro com Email Duplicado

**Objetivo**: Garantir que o cadastro falha se o email já estiver registrado.

**Passos**:

1. Repita os passos de 1 a 4 do Cenário de Teste 1.
2. No corpo da requisição, use o JSON:
3. {
4. "first\_name": "Carlos",
5. "last\_name": "Pereira",
6. "email": "joao.silva@example.com",
7. "username": "carlos\_pereira",
8. "password": "senha\_diferente"
9. }

Copiar

1. Clique em "Send".

**Resultados Esperados**:

* A resposta deve ter um status **400 Bad Request**.
* A resposta deve conter: {"message": "Email já cadastrado!"}.

Cenário de Teste 4: Campos Obrigatórios Faltando no Cadastro

**Objetivo**: Validar que o cadastro falha se qualquer campo obrigatório estiver faltando.

**Passos**:

1. Repita os passos de 1 a 4 do Cenário de Teste 1.
2. No corpo da requisição, use um JSON com campos faltando. Por exemplo:
3. {
4. "first\_name": "Paulo",
5. "last\_name": "Santos",
6. "username": "paulo\_santos",
7. "password": "senha\_secreta"
8. }

Copiar

1. Clique em "Send".

**Resultados Esperados**:

* A resposta deve ter um status **400 Bad Request**.
* A resposta deve conter uma mensagem indicando o campo faltando, como: {"message": "email é obrigatório!"}.

Considerações Finais

* **Executar Aplicativo**: Certifique-se de que o servidor Flask esteja em execução ao testar.
* **Porta Correta**: Substitua <porta> pela porta em que a aplicação Flask está executando (geralmente 5000 por padrão).
* **Validação de Resposta**: Para cada cenário, verifique tanto o código de status quanto o corpo de resposta para garantir conformidade com os requisitos.