Documentação dos Testes da API de Autenticação

Visão Geral

Esta documentação descreve a estrutura e os detalhes dos testes automatizados para a API de autenticação, utilizando Mocha, Chai e Supertest. Os testes cobrem os endpoints de cadastro (`/signup`) e login (`/login`) de usuários, garantindo que a aplicação funcione corretamente conforme esperado.

Ferramentas Utilizadas

- **Mocha:** Framework de teste para JavaScript, utilizado para estruturar e organizar os testes.
- **Chai:** Biblioteca de asserções para Node.js, utilizada para validar os resultados dos testes.
- **Supertest:** Biblioteca que facilita a realização de requisições HTTP para testar endpoints de APIs.

Estrutura dos Testes

1. Testes para o Endpoint `/signup`

Os testes para o endpoint `/signup` verificam se o processo de cadastro de novos usuários funciona corretamente.

a. Cadastro bem-sucedido

Teste: Deve cadastrar um usuário com sucesso.

```
it('Deve cadastrar um usuario com sucesso', async function() {

  const response = await request(app)
    .post('/signup')
    .send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'TestPassword123',
      confirmPassword: 'TestPassword123'
    });

  expect(response.status).to.equal(201);  // Verifica se o status da
resposta é 201 (Criado)
  expect(response.text).to.equal('Usuário cadastrado com sucesso');
// Verifica se a mensagem de resposta é a esperada
});
```

b. Cadastro com e-mail já existente

Teste: Não deveria cadastrar um usuário com um e-mail já cadastrado.

```
it('Nao deveria cadastrar um usuario com um email ja cadastrado',
async function() {
  await request(app)
    .post('/signup')
    .send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'TestPassword123',
      confirmPassword: 'TestPassword123'
    });

  const response = await request(app)
    .post('/signup')
    .send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'TestPassword123',
      confirmPassword: 'TestPassword123'
    });

  expect(response.status).to.equal(400);  // Verifica se o status da
resposta é 400 (Bad Request)
  expect(response.text).to.equal('E-mail já cadastrado');  // Verifica
se a mensagem de resposta é a esperada
});
```

2. Testes para o Endpoint `/login`

Os testes para o endpoint `/login` verificam se o processo de login de usuários funciona corretamente.

a. Login bem-sucedido

Pré-condição: Certifica-se de que há um usuário cadastrado para fazer o login.

```
before(async function() {
  await request(app)
    .post('/signup')
    .send({
      name: 'Test User',
      email: 'test@example.com',
      password: 'TestPassword123',
      confirmPassword: 'TestPassword123'
    });
});
```

Teste: Deveria logar com sucesso utilizando credenciais válidas.

```
it('deveria logar com sucesso utiizando credenciais validas', async
function() {
  const response = await request(app)
    .post('/login')
    .send({
      email: 'test@example.com',
      password: 'TestPassword123'
    });

  expect(response.status).to.equal(200);  // Verifica se o status da
resposta é 200 (OK)
  expect(response.text).to.include('Bem-vindo(a), Test User');  //
Verifica se a mensagem de resposta inclui o texto esperado
});
```

b. Login com credenciais inválidas

Teste: Não deveria logar com credenciais inválidas.

```
it('Nao deveria logar com credenciais invalidas', async function() {
  const response = await request(app)
    .post('/login')
    .send({
      email: 'test@example.com',
      password: 'WrongPassword123'
    });

  expect(response.status).to.equal(400);  // Verifica se o status da
resposta é 400 (Bad Request)
  expect(response.text).to.equal('E-mail ou senha incorretos');  //
Verifica se a mensagem de resposta é a esperada
});
```

Como Executar os Testes

1. Certifique-se de que as dependências necessárias estão instaladas:

   ```
   npm install mocha chai supertest
   ```

2. Execute os testes utilizando o comando:

   ```
   npx mocha
   ```

Os testes serão executados e os resultados serão exibidos no terminal, indicando quais testes passaram e quais falharam.

Conclusão

Esta documentação fornece uma visão detalhada dos testes automatizados para a API de autenticação, cobrindo os principais cenários de cadastro e login de usuários. Com esses testes, é possível garantir que a funcionalidade de autenticação esteja funcionando corretamente e que erros comuns, como tentativas de cadastro com e-mails duplicados ou logins com credenciais inválidas, sejam tratados adequadamente.