

Faculdade Impacta Tecnologia

Projeto de Software (5º semestre)

Curso: Análise e Desenvolvimento de Sistemas

Turma: 5A - EAD

Matéria: Software Product: Analysis, Specification, Project & Implementation

Professor: Antonio de Oliveira Dias

ALUNOS PARTICIPANTES

Ana Carollyne Guimarães de Souza	RA: 2301929
Luana Ramos de Almeida	RA: 2302018
Natã Pedro	RA: 2301849
Pedro Pescarole	RA: 2301635

SOBRE O PROJETO

Nome: Moveat

Resumo: Esse projeto busca oferecer uma experiência prática para aqueles que praticam atividades físicas. A plataforma permite que pessoais trainers e nutricionistas cadastrem seus usuários e associem planos de treino e alimentação personalizados.

Ao fazer login, os usuários encontram tudo o que precisam para seguir suas rotinas de treino e alimentação. O objetivo da plataforma é trazer praticidade, unindo tudo em um lugar só.

AC 01 – FUNCIONALIDADE AUTENTICAÇÃO (CADASTRO E LOGIN)

Objetivo: A primeira Sprint do projeto Moveat teve como objetivo a implementação das funcionalidades de autenticação (registro de novos usuários e login).

Link do board do projeto:

<https://github.com/orgs/Moveat-Fit/projects/4>

Link do vídeo apresentando a funcionalidade:

<https://youtu.be/ke7ivC1ypzA>

Link do código-fonte (GitHub):

<https://github.com/orgs/Moveat-Fit/repositories>

RELATÓRIO DE TESTES: SPRINT 1

Responsável QA: Ana Carollyne Guimarães de Souza

Data: 10/03/2025

1. INTRODUÇÃO

Este relatório tem como objetivo documentar os testes funcionais e de API realizados na aplicação **Moveat**. Os testes foram executados para garantir que o sistema atenda aos requisitos funcionais e que a API funcione conforme o esperado.

2. TESTES FUNCIONAIS DO SISTEMA

Ao clicar no ID, você será redirecionado para a documentação completa do Caso de Teste no GitHub.

ID Caso de Teste	Descrição	Status	Prioridade	Resultado Obtido
CT-001	Login com credenciais válidas	APROVADO	Alta	Figura 1 – Resultado CT001
CT-002	Redefinição de senha bem-sucedida	Não implementado	Média	-
CT-003	Tentativa de redefinição com e-mail não cadastrado	Não implementado	Média	-
CT-004	Tentativa de login com e-mail não cadastrado	APROVADO*	Alta	Figura 2 – Resultado CT004
CT-005	Tentativa de login com senha incorreta	APROVADO*	Alta	Figura 3 – Resultado CT005
CT-006	Cadastro com dados válidos	APROVADO	Alta	Figura 4 – Resultado CT006
CT-007	Tentativa de cadastro com e-mail já existente	APROVADO*	Alta	Figura 5 – Resultado CT007
CT-008	Tentativa de cadastro com campos obrigatórios em branco	APROVADO	Média	Figura 8 – Resultado CT008

3. TESTES DE API

Os testes de API foram realizados com o Postman. Abaixo está a estrutura elaborada:

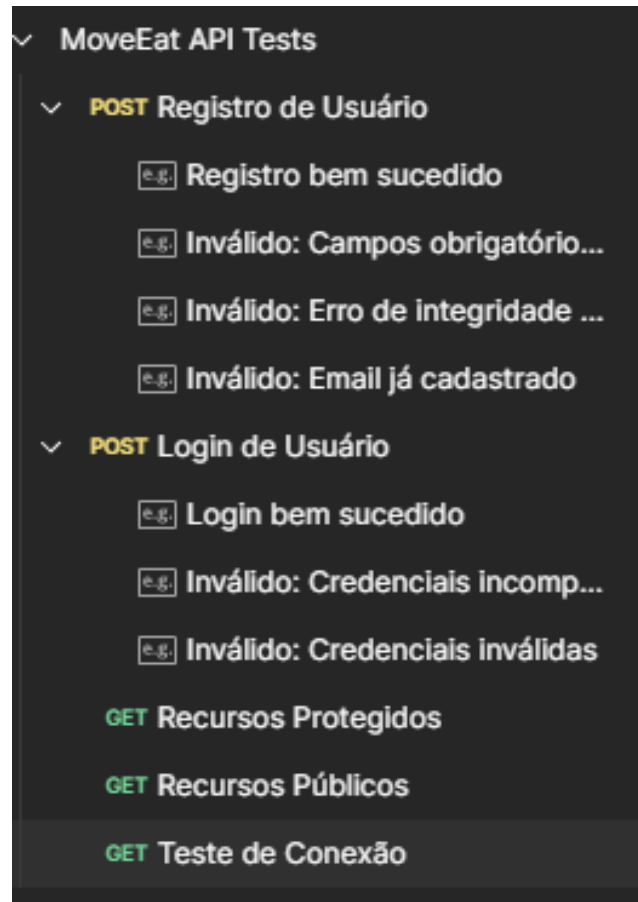
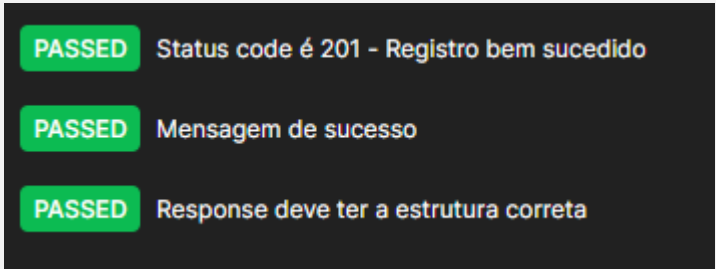


Imagem 1 - Estrutura dos testes (Postman)

Cada requisição segue o mesmo padrão de script de teste, com testes funcionais e testes de contrato. As validações foram feitas para cenários positivos e também cenários negativos.

3.1. REQUISIÇÃO: REGISTRO DE USUÁRIO (POST)

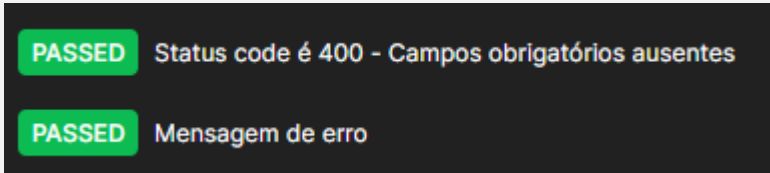
3.1.1. Cenário: Registro bem-sucedido

Status Code esperado	201
Resultado esperado	Mensagem: “Usuário registrado com sucesso”
Script de teste (Postman)	<pre>1 const jsonData = pm.response.json(); 2 3 /* TESTES PARA REGISTRO BEM SUCEDIDO */ 4 if (pm.response.code === 201) { 5 pm.test("Status code é 201 - Registro bem sucedido", function() { 6 pm.response.to.have.status(201); 7 }); 8 pm.test("Mensagem de sucesso", function() { 9 pm.expect(jsonData.message).to.eql("Usuário registrado com sucesso"); 10 }); 11 pm.test("Response deve ter a estrutura correta", function() { 12 pm.expect(jsonData).to.have.property("message"); 13 pm.expect(typeof jsonData.message).to.eql("string"); 14 }); 15 }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

3.1.2. Cenário: Email já cadastrado

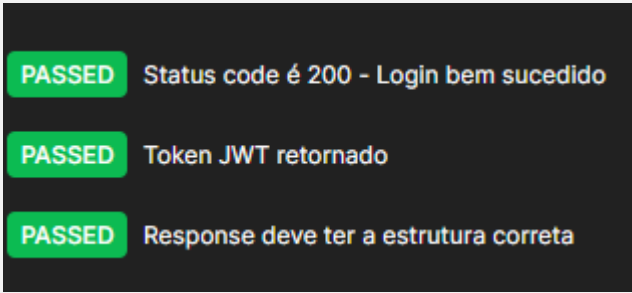
Status Code esperado	409
Resultado esperado	Mensagem: “Email já cadastrado”
Script de teste (Postman)	<pre>24 25 /* TESTES PARA EMAIL JÁ EM USO */ 26 } else if (pm.response.code === 409) { 27 pm.test("Status code é 409 - Email já está em uso", function() { 28 pm.response.to.have.status(409); 29 }); 30 pm.test("Mensagem de erro", function() { 31 pm.expect(jsonData.message).to.eql("Email já cadastrado"); 32 });</pre>
Resultado obtido	<div><div>PASSED</div> Status code é 409 - Email já está em uso</div> <div><div>FAILED</div> Mensagem de erro AssertionError: expected 'Erro de integridade de dados: (\'2300...' to deeply equal 'Email já cadastrado'</div> <div><pre>"message": "Erro de integridade de dados: ('23000', \"[23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]Violation of UNIQUE KEY constraint 'UQ_tb_Users__A9D18634B585820B'. Cannot insert duplicate key in object 'dbo.tb_Users'. The duplicate key value is (sabrina@nutricionista.com). (2627) (SQLExecDirectW); [23000] [Microsoft][ODBC Driver 18 for SQL Server][SQL Server]The statement has been terminated. (3621)\")"</pre></div>
Implementação	Manual
Status do teste	REPROVADO
Observações	Falha: não há tratamento de erro para o cenário

3.1.3. Cenário: Campos não preenchidos

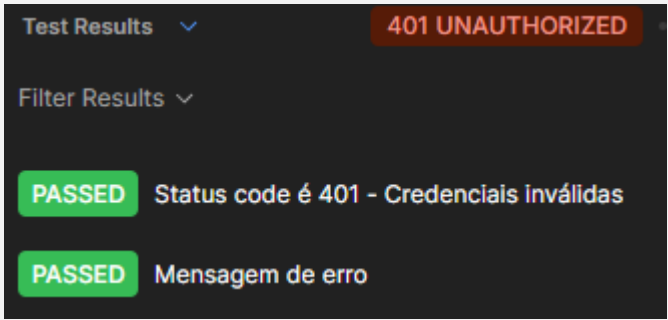
Status Code esperado	400
Resultado esperado	Mensagem: “Todos os campos obrigatórios devem ser preenchidos”
Script de teste (Postman)	<pre>/* TESTES PARA CAMPOS OBRIGATÓRIOS AUSENTES */ } else if (pm.response.code === 400) { pm.test("Status code é 400 - Campos obrigatórios ausentes", function() { pm.response.to.have.status(400); }); pm.test("Mensagem de erro", function() { pm.expect(jsonData.message).to.eql("Todos os campos obrigatórios devem ser preenchidos"); }); };</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

3.2. REQUISIÇÃO: LOGIN DE USUÁRIO (POST)

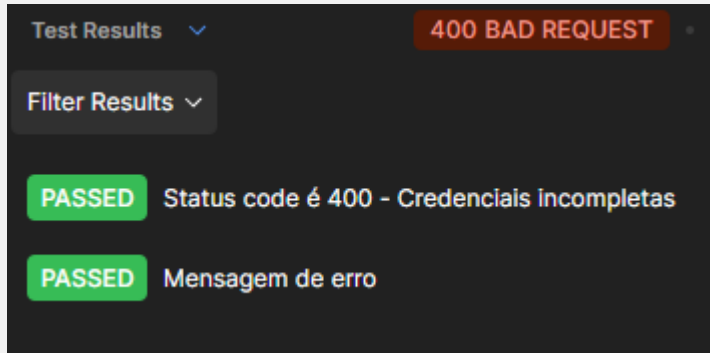
3.2.1. Cenário: Login bem-sucedido

Status Code esperado	200
Resultado esperado	Token JWT retornado
Script de teste (Postman)	<pre>const jsonData = pm.response.json(); /* TESTES PARA LOGIN BEM SUCESSADO */ if (pm.response.code === 200) { pm.test("Status code é 200 - Login bem sucedido", function() { pm.response.to.have.status(200); }); pm.test("Token JWT retornado", function() { pm.expect(jsonData.access_token).to.be.a("string"); }); pm.test("Response deve ter a estrutura correta", function() { pm.expect(jsonData).to.have.property("access_token"); pm.expect(typeof jsonData.access_token).to.eql("string"); }); pm.globals.set("jwt_token", jsonData.access_token); }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

3.2.2. Cenário: Credenciais inválidas

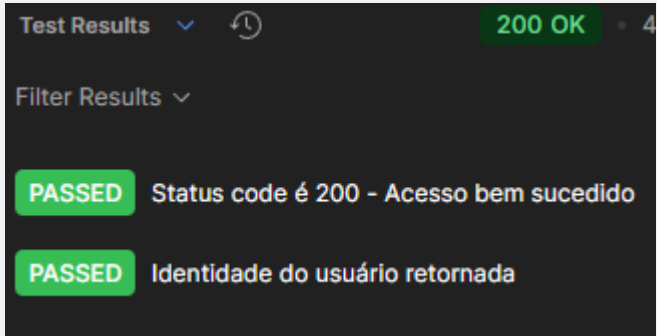
Status Code esperado	401
Resultado esperado	Mensagem: “Credenciais inválidas”
Script de teste (Postman)	<pre>/* TESTES PARA CREDENCIAIS INVÁLIDAS */ } else if (pm.response.code === 401) { pm.test("Status code é 401 - Credenciais inválidas", function() { pm.response.to.have.status(401); }); pm.test("Mensagem de erro", function() { pm.expect(jsonData.message).to.eql("Credenciais inválidas"); }); }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

3.2.3. Cenário: Campos não preenchidos

Status Code esperado	400
Resultado esperado	Mensagem: “Login e senha são obrigatórios”
Script de teste (Postman)	<pre>/* TESTES PARA CREDENCIAIS INCOMPLETAS */ } else if (pm.response.code === 400) { pm.test("Status code é 400 - Credenciais incompletas", function() { pm.response.to.have.status(400); }); pm.test("Mensagem de erro", function() { pm.expect(jsonData.message).to.eql("Login e senha são obrigatórios"); }); }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

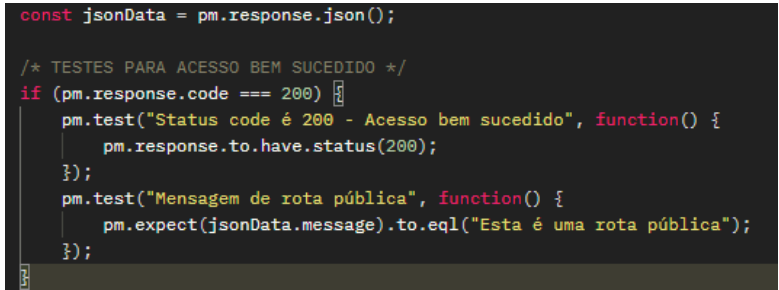
3.3. REQUISIÇÃO: RECURSOS PROTEGIDOS (GET)

3.3.1. Cenário: Acesso bem-sucedido

Status Code esperado	200
Resultado esperado	Email do usuário logado
Script de teste (Postman)	<pre>const jsonData = pm.response.json(); /* TESTES PARA ACESSO BEM SUCEDIDO */ if (pm.response.code === 200) { pm.test("Status code é 200 - Acesso bem sucedido", function() { pm.response.to.have.status(200); }); pm.test("Identidade do usuário retornada", function() { pm.expect(jsonData.logged_in_as).to.be.a("string"); }); }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

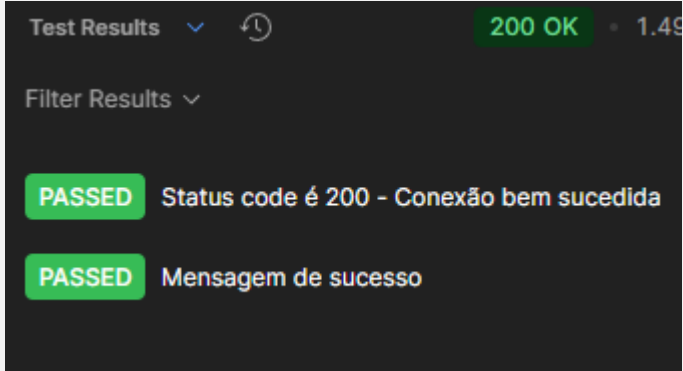
3.4. REQUISIÇÃO: RECURSOS PÚBLICOS (GET)

3.4.1. Cenário: Acesso bem-sucedido

Status Code esperado	200
Resultado esperado	Mensagem: “Esta é uma rota pública”
Script de teste (Postman)	 <pre>const jsonData = pm.response.json(); /* TESTES PARA ACESSO BEM SUCEDIDO */ if (pm.response.code === 200) { pm.test("Status code é 200 - Acesso bem sucedido", function() { pm.response.to.have.status(200); }); pm.test("Mensagem de rota pública", function() { pm.expect(jsonData.message).to.eql("Esta é uma rota pública"); }); }</pre>
Resultado obtido	 <p>Test Results 200 OK</p> <p>Filter Results</p> <p>PASSED Status code é 200 - Acesso bem sucedido</p> <p>PASSED Mensagem de rota pública</p>
Implementação	Manual
Status do teste	APROVADO

3.5. REQUISIÇÃO: TESTE DE CONEXÃO

3.5.1. Cenário: Conexão bem-sucedida

Status Code esperado	200
Resultado esperado	Mensagem: “Conexão com backend estabelecida com sucesso!”
Script de teste (Postman)	<pre>const jsonData = pm.response.json(); /* TESTES PARA CONEXÃO BEM SUCEDIDA */ if (pm.response.code === 200) { pm.test("Status code é 200 - Conexão bem sucedida", function() { pm.response.to.have.status(200); }); pm.test("Mensagem de sucesso", function() { pm.expect(jsonData.message).to.eql("Conexão com o backend estabelecida com sucesso!"); }); }</pre>
Resultado obtido	
Implementação	Manual
Status do teste	APROVADO

4. MELHORIAS IDENTIFICADAS

Durante a fase de testes, algumas melhorias na tela de cadastro foram encontradas e documentadas. Elas não impedem que o sistema funcione, mas são indispensáveis para um usuário.

ID Melhoria	Descrição
UPGRADE-CAD-001	Limitar caracteres do campo “CPF”
UPGRADE-CAD-002	Destacar email não preenchido corretamente
UPGRADE-CAD-003	Destacar senha fora do padrão
UPGRADE-CAD-004	Validação de senhas diferentes
UPGRADE-CAD-005	Destacar campos obrigatórios não preenchidos
UPGRADE-CAD-006	Bloquear letras e caracteres especiais em “telefone” e “CPF”

5. EVIDÊNCIAS

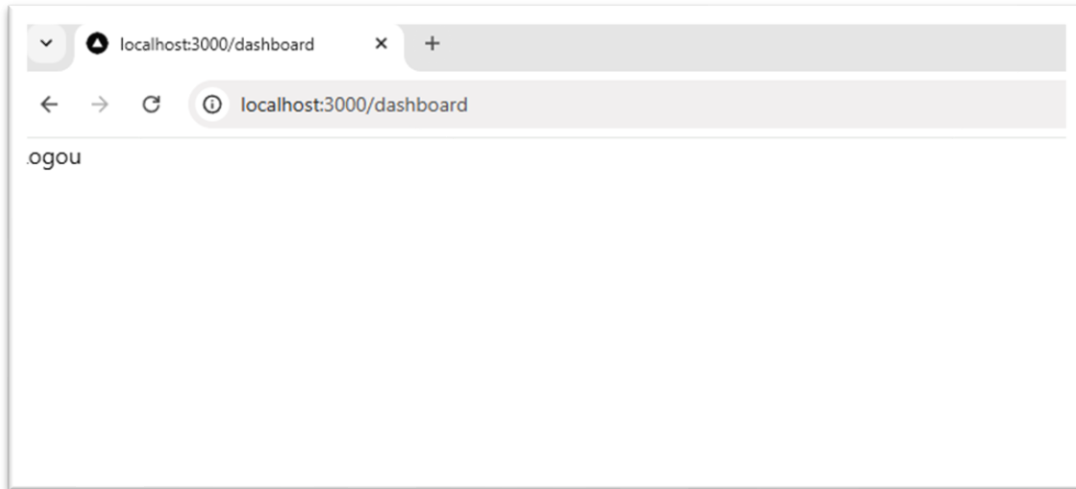


Figura 1 - Resultado CT001

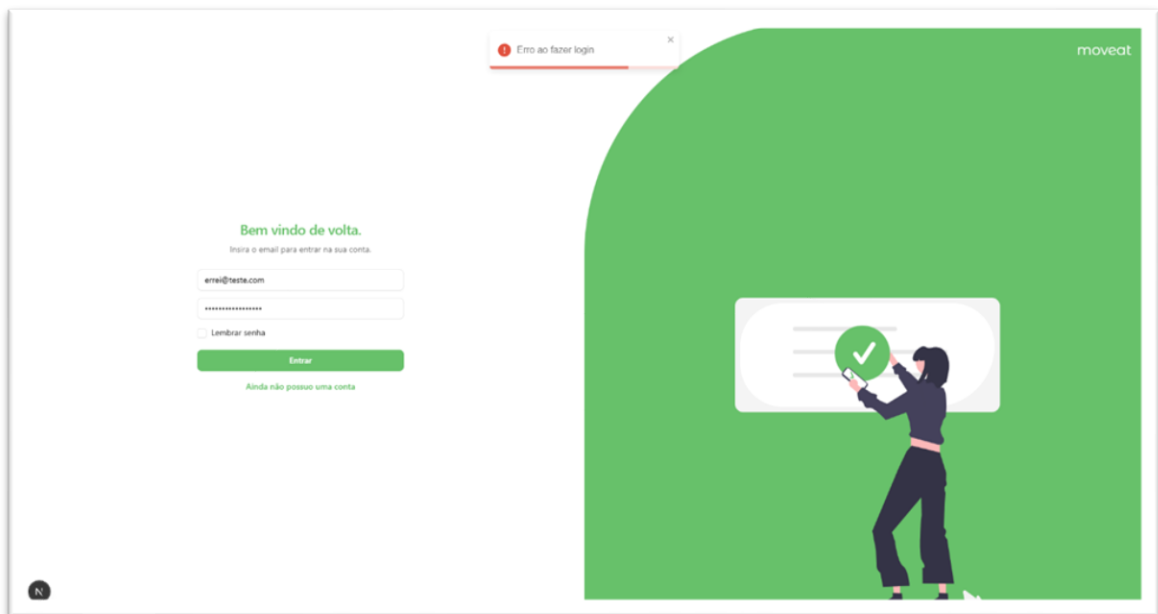


Figura 2 - Resultado CT004

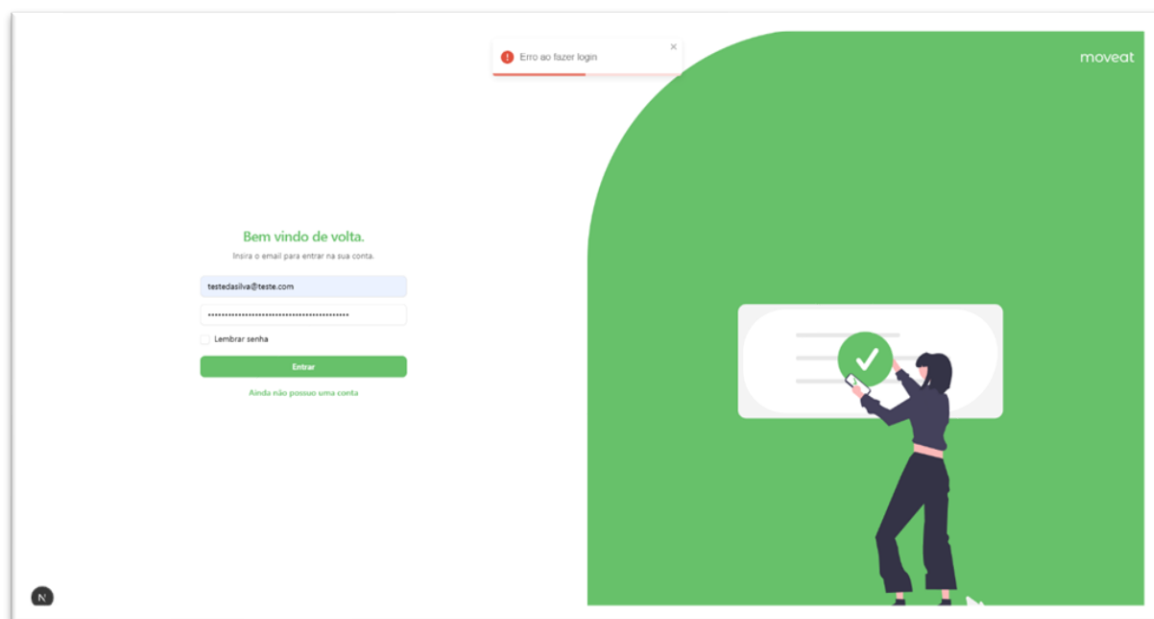


Figura 3 – Resultado CT005

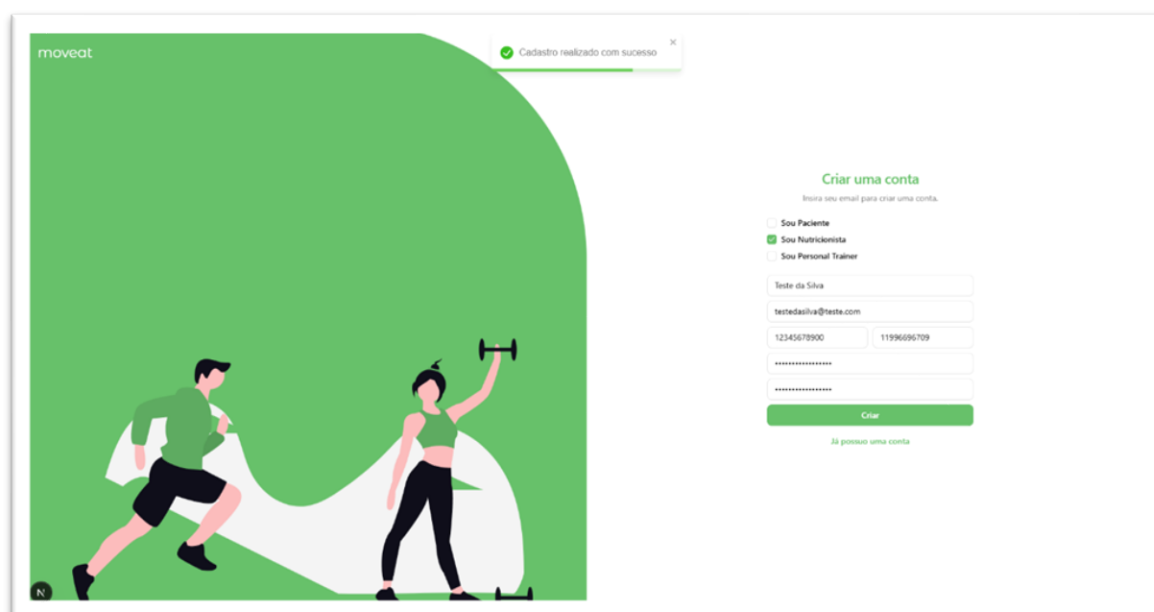


Figura 4 – Resultado CT006

moveat

Erro ao se registrar

Criar uma conta

Inicie seu email para criar uma conta.

☒ Sou Paciente
☐ Sou Nutricionista
☐ Sou Personal Trainer

Teste dos Santos

teste@live@teste.com

40570096322 11996096705

Criar

Já possui uma conta

Figura 5 – Resultado CT007

moveat

Erro ao se registrar

Criar uma conta

Inicie seu email para criar uma conta.

☐ Sou Paciente
☐ Sou Nutricionista
☒ Sou Personal Trainer

Teste dos Santos

umemail@email.com

CPF: 11960094748

Criar

Já possui uma conta

Figura 6 – Resultado CT008