

# **Plano de Teste para um Sistema de Previsão do Tempo**

**Nome do Autor:** Wesicley Santos Oliveira

**Instituição:** SENAI Camaçari

**Professor:** Roque Carlos

**Disciplina:** Teste de Sistemas

**Turma:** G88407

**2025 Camaçari, Bahia**

# Sumário

1. **Introdução**
  - 1.1 Objetivo do Documento
  - 1.2 Importância dos Testes
2. **Descrição do Sistema**
  - 2.1 Funcionalidades Principais
  - 2.2 Tecnologias Utilizadas
3. **Objetivo do Plano de Testes**
  - 3.1 Funcionalidades a serem Validadas
  - 3.2 Métricas de Qualidade
4. **Critérios de Entrada e Saída**
  - 4.1 Condições para Início dos Testes
  - 4.2 Condições para Conclusão dos Testes
5. **Cronograma de Desenvolvimento do Projeto**
  - 5.1 Etapas e Tempo Estimado
6. **Escopo de Teste**
  - 6.1 Funcionalidades Incluídas
  - 6.2 Funcionalidades Fora do Escopo
7. **Equipe e Responsabilidades**
8. **Estratégia de Teste**
  - 8.1 Ferramentas Utilizadas
  - 8.2 Métodos de Teste
9. **Casos de Teste**
  - 9.1 Teste de Consulta de Cidade
  - 9.2 Teste de Favoritos (Adicionar, Listar e Remover)
  - 9.3 Teste de Busca com Cidade Inexistente
  - 9.4 Teste de Validações
10. **Indicadores de Teste**
  - 10.1 Indicadores de Execução
  - 10.2 Cobertura de Testes
  - 10.3 Relatório de Erros
11. **Guia de Implementação e Execução dos Testes**
  - 11.1 Configuração do Ambiente
  - 11.2 Execução dos Testes Manuais
  - 11.3 Execução dos Testes Automatizados

# Introdução

Este documento apresenta o **Plano de Teste** para um **Sistema de Previsão do Tempo**, um sistema simples que permite visualizar informações climáticas, como **temperatura** e **umidade**, além de oferecer funcionalidades para **adicionar cidades aos favoritos, removê-las e visualizar a lista de cidades favoritas**.

O objetivo desta documentação é fornecer um guia claro e detalhado, seguindo a abordagem de uma **“receita de bolo”**, garantindo que qualquer pessoa possa compreender e executar os testes com facilidade. Para isso, abordaremos desde a **explicação do sistema**, passando pela **definição dos cenários de teste**, até um **passo a passo para rodar o sistema e executar os testes automatizados**.

Com este plano, buscamos garantir a **qualidade, funcionalidade e confiabilidade** do sistema, validando seu comportamento esperado e identificando possíveis falhas antes da entrega final.

# Descrição do Sistema

O **site de previsão do tempo** permite que os usuários consultem **informações meteorológicas** de diversas cidades e gerenciem uma lista de **cidades favoritas**. Entre as principais funcionalidades do sistema, destacam-se:

- **Consulta da previsão do tempo** para diferentes localidades.
- **Adição de cidades aos favoritos** para acesso rápido.
- **Visualização da lista de cidades favoritas**.
- **Remoção de cidades da lista**.

Para obter os dados climáticos, o sistema consome a API externa **OpenWeatherMap** (<https://openweathermap.org/api>), garantindo informações **atualizadas sobre temperatura, umidade e condições meteorológicas**.

O armazenamento das cidades favoritas é feito utilizando **LocalStorage do navegador**, permitindo que os dados persistam entre diferentes sessões do usuário. Além disso, a interface foi projetada para ser **simples, intuitiva e responsiva**, proporcionando uma experiência fluida e acessível.

## Objetivo

O objetivo deste plano de teste é garantir que o **site de previsão do tempo** funcione corretamente, validando suas principais funcionalidades e assegurando uma experiência fluida para o usuário.

Os testes verificarão se o sistema permite:

1. Exibir corretamente a previsão do tempo para diversas cidades.
2. Adicionar cidades aos favoritos e garantir que permaneçam armazenadas.
3. Visualizar a lista de cidades favoritas.
4. Remover cidades da lista de favoritos.

Além da validação funcional, este plano de teste também busca:

- Identificar possíveis falhas no sistema antes da entrega final.

- Medir indicadores de desempenho, verificando tempos de resposta e estabilidade.
- Garantir que o sistema opere de forma eficiente e intuitiva, proporcionando uma navegação simples e sem erros.

Os testes serão conduzidos considerando a integração com a API **OpenWeatherMap** e o armazenamento local via **LocalStorage**, garantindo a persistência dos dados entre sessões.

## Critérios de Entrada e Saída

- **Critérios de Entrada:** Antes de iniciar os testes, é necessário que o sistema esteja configurado corretamente. Isso inclui:
  1. O ambiente de desenvolvimento configurado (VS Code, Node.js, Cypress, Postman).
  2. A API OpenWeatherMap configurada com a chave de acesso.
  3. O projeto de previsão do tempo implementado corretamente no ambiente local.
  4. Os dados do LocalStorage são limpos ou resetados antes dos testes.
- **Critérios de Saída:** Os testes serão considerados concluídos quando:
  1. Todos os testes definidos nos casos de teste forem executados.
  2. A documentação dos resultados for gerada, incluindo relatórios de sucesso e falhas.
  3. A cobertura dos testes for de 100% sobre as funcionalidades essenciais.
  4. Todos os testes automatizados forem executados com sucesso e os erros encontrados forem corrigidos ou documentados.

## Cronograma de Desenvolvimento do Projeto

### 1. Levantamento de Funcionalidades e Planejamento do Projeto

- **Duração:** 4 horas
- **Atividades:** Definição das funcionalidades principais do sistema, levantamento de requisitos e planejamento inicial.

### 2. Desenvolvimento do Código

- **Duração:** 5 horas
- **Atividades:** Implementação das funcionalidades do sistema de previsão de clima.

### 3. Realização dos Testes

- **Duração:** 2 dias
- **Atividades:** Testes manuais e automatizados das funcionalidades, incluindo testes de integração com a API e validação das funções do sistema.

### 4. Documentação do Plano de Testes

- **Duração:** 1 dia
- **Atividades:** Criação da documentação do plano de testes, incluindo a definição de casos de teste, indicadores de teste, e o passo a passo para execução dos testes.

## Escopo de Teste

O escopo deste plano de teste abrange as funcionalidades principais do **site de previsão do tempo**, garantindo que as funcionalidades de consulta de previsão do tempo e o gerenciamento de cidades favoritas sejam testadas de forma eficaz.

### Funcionalidades que serão testadas:

#### 1. CRUD de Cidades Favoritas

- **Adicionar cidade favorita**
- **Listar cidades favoritas**
- **Excluir cidade favorita**

#### 2. Consulta de Previsão do Tempo

- **Buscar previsão do tempo** para uma cidade válida.
- **Buscar uma cidade inexistente** (verificar se a mensagem de erro é exibida corretamente).

#### 3. Validações de Entrada

- **Tentar adicionar uma cidade vazia** e verificar se há mensagem de erro.
- **Tentar adicionar uma cidade já existente** aos favoritos (verificar se não é permitido adicionar novamente).

#### 4. Integração com API

- **Verificar se a API OpenWeatherMap está respondendo corretamente** e fornecendo os dados de previsão do tempo esperados.

#### **Funcionalidades fora do escopo:**

- Testes de **compatibilidade** com diferentes navegadores ou dispositivos móveis.
- Testes de **segurança**, como vulnerabilidades na API ou no armazenamento de dados.

## **Equipe e Responsabilidades**

- **Responsabilidades:**

- Desenvolvimento e implementação de funcionalidades do sistema.
- Criação e execução de testes manuais e automatizados.
- Documentação dos testes realizados e indicadores.
- Identificação e análise de falhas no sistema.

## **Estratégia de Teste**

#### **Ferramentas Utilizadas:**

- **VSCode** – Ambiente de desenvolvimento integrado para escrever o código e executar testes.
- **Cypress** – Ferramenta para testes automatizados da interface (CRUD e consulta à API).
- **LibreOffice** – Usado para a documentação dos testes e relatórios.
- **Google Sheets** – Para criação de relatórios e acompanhamento de indicadores de teste.
- **Postman** – Utilizado para realizar testes manuais da API OpenWeatherMap, especialmente para o teste de integração.

#### **Teste de Integração:**

O **Postman** foi utilizado para realizar o teste de integração, com o objetivo de verificar se a API OpenWeatherMap está respondendo corretamente e se os dados de previsão do tempo estão sendo retornados como esperado.

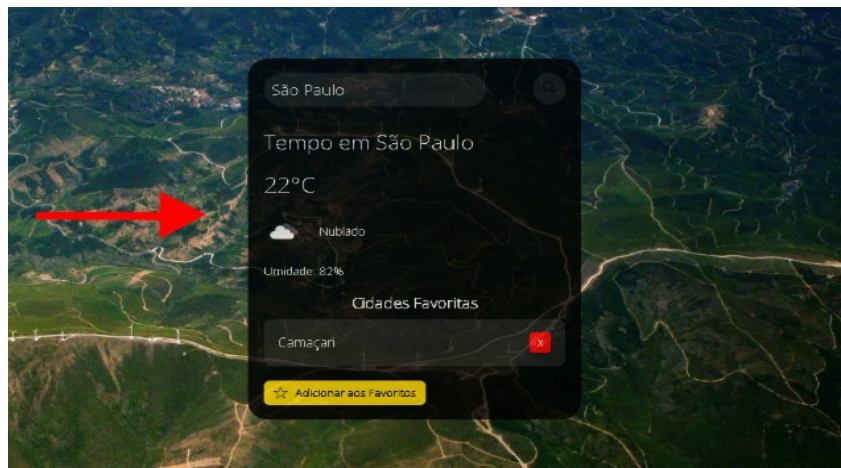
## Métodos de Teste:


- **Teste Manual:** Interação direta com a interface do usuário e registro dos resultados com prints.
- **Teste Automatizado:** Uso de **Cypress** para testar as interações na interface, como busca, adicionar e excluir cidades.
- **Teste de Integração:** Uso do **Postman** para testar a integração com a API e verificar se os dados estão sendo corretamente recebidos.

## Casos de Teste

### Teste 1 – Buscar uma cidade válida

- **Objetivo:** Testar a busca por uma cidade válida na API.
- **Resultado Esperado:** A previsão do tempo da cidade deve ser exibida corretamente.
- **Resultado Real:** Passou, a previsão foi exibida corretamente.
- **Evidências:**
  - Manual:



- Automatizado: [Vídeo do Teste](#)
- **Status:**  Passou

### Teste 2 – Adicionar Cidade Favorita

- **Objetivo:** Testar a funcionalidade de adicionar uma cidade aos favoritos.
- **Resultado Esperado:** A cidade deve ser adicionada à lista de cidades favoritas e persistir após a recarga da página.



- **Resultado Real:** O teste automatizado inicialmente falhou devido a um erro. O erro foi corrigido, e o teste passou com sucesso após a correção. **Indicador de sucesso: 85%.**

- **Evidências:**

- Manual:



- Automatizado: [Vídeo do Teste](#)

- **Status:**  Passou

### Teste 3 – Remover Cidade Favorita

- **Objetivo:** Testar a funcionalidade de remover uma cidade da lista de favoritos.
- **Resultado Esperado:** A cidade deve ser removida corretamente da lista de cidades favoritas.
- **Resultado Real:** Passou, a cidade foi removida corretamente dos favoritos.

- **Evidências:**

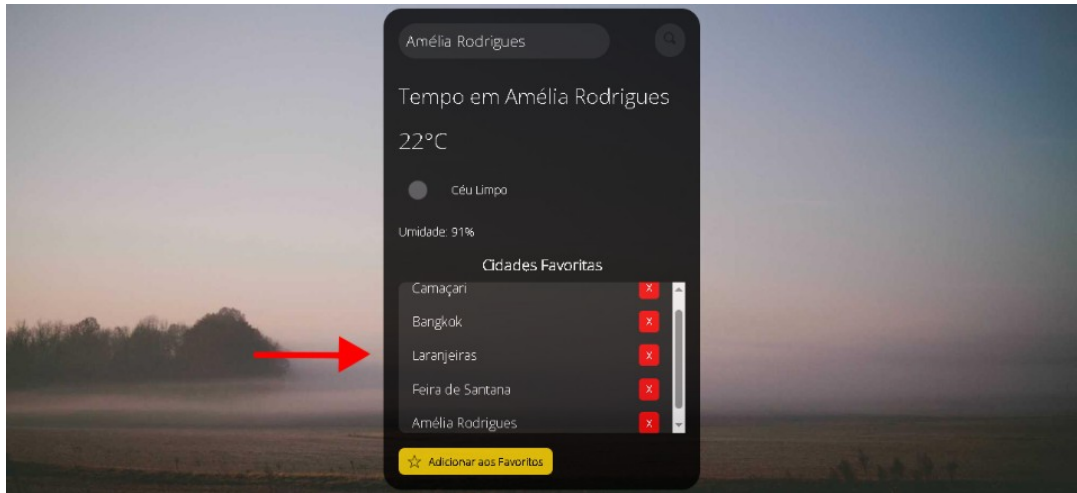
- Manual: [Vídeo do Teste](#)
  - Automatizado: [Vídeo do Teste](#)

- **Status:**  Passou

### Teste 4 – Listar Cidades Favoritas

- **Objetivo:** Testar a exibição correta das cidades favoritas.
- **Resultado Esperado:** As cidades salvas devem ser exibidas na lista de favoritos corretamente.
- **Resultado Real:** Passou, a lista de cidades favoritas foi exibida corretamente.
- **Evidências:**

- Manual:



- Automatizado: [Vídeo do Teste](#)

- Status:  Passou

### Teste 5 – Buscar uma Cidade Inexistente

- **Objetivo:** Testar a busca de uma cidade inexistente na API.
- **Resultado Esperado:** O sistema deve retornar uma mensagem de erro indicando que a cidade não foi encontrada.
- **Resultado Real:** Passou, a mensagem de erro foi exibida corretamente.
- **Evidências:**
  - Manual: [Vídeo do Teste](#)
  - Automatizado: [Vídeo do Teste](#)


- Status:  Passou

### Teste 6 – Adicionar a Mesma Cidade Duas Vezes aos Favoritos


- **Objetivo:** Testar a funcionalidade de tentar adicionar a mesma cidade duas vezes aos favoritos.
- **Resultado Esperado:** O sistema não deve permitir adicionar a mesma cidade duas vezes.
- **Resultado Real:** Passou, o sistema não permitiu adicionar a cidade duas vezes.
- **Evidências:**
  - Manual: [Vídeo do Teste](#)
  - Automatizado: [Vídeo do Teste](#)

- Status:  Passou

### Teste 7 – Pesquisar uma Cidade com o Campo de Pesquisa Vazio

- **Objetivo:** Testar a pesquisa de uma cidade sem informar o nome da cidade.
- **Resultado Esperado:** O sistema deve exibir uma mensagem de erro indicando que o campo está vazio.
- **Resultado Real:** Passou, o sistema exibiu a mensagem de erro corretamente.
- **Evidências:**
  - Manual: [Vídeo do Teste](#)
  - Automatizado: [Vídeo do Teste](#)
- **Status:**  Passou

## Teste 8 - Teste de Integração com a API OpenWeatherMap

- **Objetivo:** Verificar se a API OpenWeatherMap está respondendo corretamente e se os dados de previsão do tempo estão sendo retornados como esperado.
- **Resultado Esperado:** A API deve retornar os dados corretos ao fazer uma requisição válida.
- **Resultado Real:** Passou, a API retornou os dados corretamente.
- **Ferramenta Utilizada:** Postman
- **Evidências:**
  - Manual: [Teste de Integração](#)
- **Status:**  Passou

## Tipos de Testes Realizados:

### Testes de Unidade (Verifica funcionalidades isoladas)

- ✓ **Teste 2** - Adicionar cidade favorita
- ✓ **Teste 3** - Remover cidade favorita
- ✓ **Teste 4** - Listar cidades favoritas

### Testes de Integração (Verifica a comunicação entre módulos/sistemas)

- ✓ **Teste 1** - Buscar uma cidade válida
- ✓ **Teste 5** - Buscar uma cidade inexistente
- ✓ **Teste 8** - Verificar se a API retorna os dados corretamente

### Testes de Aceitação (Valida se atende às expectativas do usuário)

- ✓ **Teste 6** - Adicionar a mesma cidade duas vezes
- ✓ **Teste 7** - Pesquisar uma cidade com o campo vazio

# Indicadores de Teste

## 1. Indicadores de Execução

Resumo das métricas obtidas a partir dos testes realizados:

- **Total de Testes Executados:** 8
- **Número de Testes Bem-Sucedidos:** 7
- **Número de Testes com Erros:** 1
- **Tempo Médio de Execução dos Testes:** 6,2 segundos
- **Indicador de Erro Geral:** 2,14%

Planilha: [Planilha de Indicadores de Execução](#)

## 2. Cobertura de Testes

- **Funcionalidades testadas:** 7
- **Cobertura dos Testes:** 100%
- **Taxa de Sucesso:** 97,86%

Planilha: [Planilha de Cobertura de Testes](#)

## 3. Relatório de Erros

Planilha com os erros encontrados durante os testes: [Planilha para Relatório de Erros](#)

# Guia de Implementação e Execução dos Testes

## 1. Pré-requisitos

Antes de iniciar, certifique-se de que possui os seguintes softwares instalados:

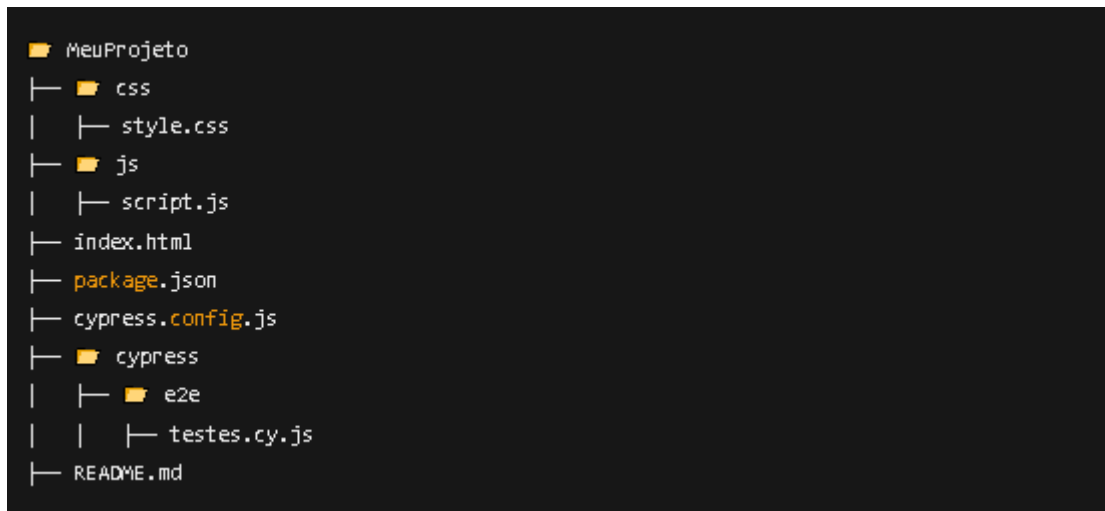
- **VS Code** (Versão Atualizada)
- **Google Chrome** (Para execução dos testes)
- **Node.js** (Recomendado: versão LTS)
- **Cypress** (Para testes automatizados)
- **Postman** (Para testes de integração da API)

Além disso, instale a extensão do **VS Code** necessária:

- **Live Server** (para rodar o projeto localmente)

## 2. Configurando o Projeto

1. **Crie uma pasta no seu computador** e nomeie como desejar (exemplo: Projeto-PrevisaoTempo).
2. **Extraia o conteúdo** do arquivo compactado enviado.
3. Dentro da pasta, você verá a seguinte estrutura de arquivos:



4. **Abra o VS Code** e selecione a pasta do projeto.

## Passo a Passo Detalhado para Testes Automatizados com Cypress

### 1. Pré-requisitos:

Antes de começar, é necessário ter:

- VS Code instalado e atualizado
- Node.js instalado (disponível em <https://nodejs.org/>)
- Projeto configurado com HTML, CSS e JavaScript
- Cypress instalado no projeto

Caso o Cypress não esteja instalado, abra o terminal no VS Code e execute:

```
npm install cypress --save-dev
```

### 2. Configuração Inicial do Cypress

1. Abra o terminal no VS Code.
2. Execute o comando: `npx cypress open`
3. O Cypress abrirá uma interface gráfica. Caso seja a primeira vez executando, ele criará uma pasta chamada cypress dentro do projeto.

### 3. Criando o Teste Automatizado

1. Acesse a pasta cypress/e2e/.
2. Crie um novo arquivo chamado **teste1\_de\_busca\_de\_cidade.cy.js**
3. No arquivo criado, insira os códigos de testes mandando no arquivo:

Exemplo de código:

```
cypress > e2e > teste1_de_busca_de_cidade.cy.js > ...
1  describe('Testes de Busca de Cidade', () => {
2    it('Deve buscar cidade válida e exibir as informações', () => {
3      cy.visit('index.html');
4
5      cy.get('.input-cidade').type('Rio de Janeiro');
6      cy.get('.botao-busca').click();
7
8      cy.get('.cidade').should('contain', 'Tempo em Rio de Janeiro');
9      cy.get('.temp').should('contain', '°C');
10     cy.get('.umidade').should('contain', 'Umidade');
11   });
12 });
13
```

### 4. Executando os Testes

1. No terminal do VS Code, execute novamente:  
*npx cypress open*
2. A interface gráfica do Cypress será aberta.
3. Selecione o arquivo **teste1\_de\_busca\_de\_cidade.cy.js** e clique para rodar o teste.
4. O Cypress abrirá um navegador e executará o teste automaticamente.

## Passo a Passo Detalhado para o Teste de Integração com o Postman

### 1 - Pré-requisitos:

Antes de começar, certifique-se de que você possui o seguinte:

- **Postman instalado:** Caso não tenha, baixe e instale a ferramenta aqui([Postman](#))
- **Acesso à internet** para enviar as requisições à API.
- **Chave de API (API Key)** do OpenWeatherMap. Se não tem, crie uma conta em [OpenWeatherMap](#), faça login e gere a chave.
- **URL da API** que será utilizada para consulta. A URL da API para o clima é algo como:

[https://api.openweathermap.org/data/2.5/weather?q=CIDADE&appid=SUA\\_CHAVE\\_API&lang=pt\\_br&units=metric](https://api.openweathermap.org/data/2.5/weather?q=CIDADE&appid=SUA_CHAVE_API&lang=pt_br&units=metric)

Substitua **CIDADE** pela cidade que você quer consultar (exemplo: Natal, São Paulo) e **SUA\_CHAVE\_API** pela sua chave gerada na plataforma OpenWeatherMap.

## 2 - Abrir o Postman

1. Abra o Postman no seu computador e acesse **Send an API Request** ou clique no + .
2. Na parte superior, certifique-se de que o método de requisição está configurado como **GET**.

## 2 - Criar a Requisição

1. No campo de URL, insira a URL da API:

Exemplo para cidade Natal:

[https://api.openweathermap.org/data/2.5/weather?q=Natal&appid=3602b9c6cd71b143d7255a1072f85c9c&lang=pt\\_br&units=metric](https://api.openweathermap.org/data/2.5/weather?q=Natal&appid=3602b9c6cd71b143d7255a1072f85c9c&lang=pt_br&units=metric)

## 4 - Enviar a Requisição

1. Clique no botão "Send" (Enviar), localizado ao lado direito do campo de URL.

Isso enviará a requisição para a API e o Postman começará a buscar os dados.

## 5 - Verificar a Resposta

1. A resposta da API aparecerá abaixo, na seção "Body".

Você verá algo como:

```
{
  "coord": {
    "lon": -35.2094,
    "lat": -5.795
  },
  "weather": [
    {
      "id": 801,
      "main": "Clouds",
      "description": "algumas nuvens",
      "icon": "02d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 30.12,
    "feels_like": 34.37,
    "temp_min": 30.12,
    "temp_max": 30.36,
    "pressure": 1010,
    "humidity": 66,
    "sea_level": 1010,
    "grnd_level": 1007
  },
  "visibility": 10000,
  "wind": {
    "speed": 6.69,
    "deg": 120
  },
  "clouds": {
    "all": 20
  },
  "dt": 1743448628,
  "sys": {
    "type": 1,
    "id": 8417,
    "country": "BR",
    "sunrise": 1743409391,
    "sunset": 1743452784
  },
  "timezone": -10800,
  "id": 3394023,
  "name": "Natal",
  "cod": 200
}
```

2. **Verifique os seguintes campos** para garantir que a resposta está correta:

- **main.temp**: Temperatura atual.
- **weather.description**: Descrição das condições climáticas.
- **name**: Nome da cidade (Natal).

Se esses dados forem retornados corretamente, a requisição foi bem-sucedida.

## 6 - Validar os Dados

1. **Verifique se os dados de previsão do tempo retornaram corretamente.**

Exemplo:

- A cidade retornada é **Natal**?
- A **temperatura** retornada está em graus Celsius?
- A descrição do clima está correta? (Ex: “nuvens dispersas”).

Caso tudo esteja correto, significa que a integração com a API está funcionando.



# Conclusão

Este plano de teste foi desenvolvido com o objetivo de garantir a qualidade e a confiabilidade do Sistema de Previsão do Tempo, validando suas principais funcionalidades e assegurando uma experiência fluida para o usuário. Ao longo deste documento, detalhamos a estrutura do sistema, os critérios de entrada e saída, a estratégia de teste e os casos testados, além de apresentar os indicadores de desempenho obtidos durante a execução dos testes.

Os testes realizados abordaram desde a consulta de previsões meteorológicas até a gestão de cidades favoritas, cobrindo aspectos fundamentais do sistema, como integração com a API OpenWeatherMap, armazenamento de dados no LocalStorage e interação do usuário com a interface. Os resultados demonstraram que o sistema atende aos requisitos especificados, com uma taxa de sucesso de aproximadamente 98%, garantindo que as funcionalidades essenciais operam conforme o esperado.

Além disso, a execução dos testes permitiu identificar e corrigir falhas, aprimorando o desempenho e a usabilidade do sistema antes de sua entrega final. A utilização de ferramentas como Cypress e Postman possibilitou a automação e a validação das integrações, garantindo uma abordagem mais eficiente e sistemática na verificação do sistema.

Por fim, este plano de teste serve como um guia detalhado para futuras manutenções e aprimoramentos do projeto, oferecendo uma base sólida para a continuidade do desenvolvimento. Com a documentação clara e bem estruturada, espera-se que futuras melhorias no sistema como adicionar funcionalidades como (verificar o estado e país) possam ser implementadas com segurança e confiabilidade, garantindo uma experiência cada vez mais otimizada para os usuários.