21 de novembro de 2025

RELATÓRIO DE TESTES

Conteúdo do relatório

Este relatório fornece informações importantes sobre os testes com inteligência artificial da TestSprite. Para dúvidas ou necessidades personalizadas, entre em contato conosco pelo Calendly ou participe da nossa comunidade no Discord .

Índice

Sumário executivo

1 Visão Geral de Alto Nível

VISÃO GERAL	
Total de APIs testadas	0 APIs
Total de sites testados	0 Sites
Taxa de aprovação/reprovação	Backend: 0 / 0 Frontend: 0 / 0

2 Principais Descobertas

Resumo do teste

O projeto apresenta uma distribuição equilibrada dos resultados dos testes, mas atualmente carece de resultados de testes tanto de backend quanto de frontend. Consequentemente, a pontuação de qualidade reflete uma avaliação inicial, e não uma representação precisa. A ausência de dados impede uma análise completa da confiabilidade e do desempenho, sugerindo potenciais áreas problemáticas que precisam ser investigadas.

O que poderia ser melhor?

A ausência de resultados de testes detalhados indica lacunas significativas nos dados de desempenho observáveis. Essa carência dificulta a identificação de padrões problemáticos ou falhas críticas, sugerindo uma necessidade imediata de aumentar a cobertura de testes tanto para os sistemas de backend quanto para os de frontend, a fim de garantir a estabilidade.

Recomendações

Para melhorar a confiabilidade e aumentar a pontuação de qualidade do projeto, recomenda-se implementar testes abrangentes tanto para as APIs de backend quanto para as URLs de frontend. Revise e atualize regularmente as estratégias de teste com base em padrões emergentes, concentrando-se em áreas que demonstram falhas repetidas ou problemas críticos.

Resultados dos testes da API de backend

3. Resumo da Cobertura dos Testes

NOME DA API	CASOS DE TESTE	CATEGORIA DE TESTE	TAXA DE APROVAÇÃO/REPROVAÇÃO
planejagov	10	5 Testes Funcionais Básicos 5 Testes de Casos Extremos	2 aprovados/ 8 reprovados

Observação
Os casos de teste foram gerados com base nas especificações da API e nos comportamentos observados. Alguns testes foram adaptados dinamicamente durante a execução, com base nas respostas da API.

4. Resumo da Execução dos Testes

Resumo Da Execução Do Planegov

CASO DE TESTE	DESCRIÇÃO DO TESTE	IMPACTO	STATUS

Testes Funcionais Básicos			
Teste de Criação de Recursos Duplicados	Tente criar um recurso que já existe e verifique se a API retorna um status de Conflito 409 com uma mensagem de erro relevante.	Médio	Fracassado
Teste de tratamento de entrada inválida	Envie uma solicitação POST com campos de dados inválidos e confirme se ela retorna um status 400 Bad Request com as mensagens de erro apropriadas.	Médio	Fracassado
Criar teste de recurso	Garanta que uma solicitação POST válida para criar um novo recurso seja bem-sucedida e retorne um status 201 com a estrutura de resposta correta.	Alto	Fracassado
Excluir teste de recurso	Verifique se uma solicitação DELETE válida remove com sucesso o recurso especificado e retorna um código de status 204.	Alto	Fracassado
Atualização do teste de recursos	Verificar se um recurso existente pode ser atualizado com uma solicitação PUT válida, retornando um status 200 e dados atualizados no corpo da resposta.	Alto	Fracassado
Testes de casos extremos			
Teste JSON vazio	Envie um objeto JSON vazio em uma solicitação POST e verifique se a API responde com um status 400 Bad Request devido à falta de campos obrigatórios.	Alto	Aprovado
Teste de entrada excessivamente longo	Envie uma solicitação POST com campos de entrada que excedam as restrições de comprimento máximo e confirme se a API retorna um status 400 com uma mensagem de erro.	Alto	Fracassado
Teste de acesso não autorizado	Tente atualizar ou excluir um recurso sem a devida autenticação e verifique se o status retornado é 401 Não Autorizado.	Alto	Fracassado
Teste JSON malformado	Envie um JSON mal formatado em uma solicitação POST e verifique se a resposta inclui um status 400 Bad Request e uma mensagem de erro útil.	Médio	Fracassado
Teste de ID de recurso inválido	Faça uma solicitação PUT para atualizar um recurso com um ID que não existe e certifique-se de que um status 404 Não Encontrado seja retornado.	Alto	Aprovado

5. Detalhamento da Execução de Testes

Detalhes Do Teste Reprovado Do Planejamentogov

Teste de Criação de Recursos Duplicados

ATRIBUTOS	
Status	Fracassado
Prioridade	Médio
Descrição	Tente criar um recurso que já existe e verifique se a API retorna um status de Conflito 409 com uma mensagem de erro relevante.

Código de teste

```
 solicitações de importação
importar  json

def  test_duplicate_resource_creation ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    cabeçalhos =  { "Autorização" : "Portador ." }

    # Tentativa de criar um recurso
    carga útil =  {
        "nome" : "Recurso de teste" ,
        "descrição" : "Este é um recurso de teste."
    }

    resposta = requests.post ( url ,  headers=headers ,  json=payload )
    print ( f "Resposta da primeira criação: {response.status_code},
    {response.text}" )

    # Tentativa de criar o mesmo recurso novamente
    resposta_duplicada = requests.post ( url ,  headers=headers ,
    json=payload )
    print ( f "Resposta de criação duplicada: {response_duplicate.
    status_code}, {response_duplicate.text}" )

    # Verifique se o código de status não é 200 na segunda tentativa de
    indicar um recurso duplicado
    assert  response_duplicate.status_code !=  200 ,  f "Status esperado
    O código não deveria ser 200, mas foi obtido {response_duplicate.status_code }"

test_duplicate_resource_creation ( )
```

Erro

O código de status esperado não era 200, mas foi 200 que recebi.

Rastrear

Teste de Criação de Recursos Duplicados

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  24 , em  <módulo>
  Arquivo  "<string>" ,  linha  22 , em  test_duplicate_resource_creation
AssertionError :  O código de status esperado  não  era  200 ,  mas foi recebido
```

Causa

A API não está lidando corretamente com a criação de recursos duplicados, possivelmente permitindo que o mesmo recurso seja criado várias vezes em vez de retornar um erro ou um status indicando um conflito.

Consertar

Implemente a lógica na API para verificar a existência de um recurso com os mesmos atributos (por exemplo, nome) antes de permitir sua criação. Se uma duplicata for detectada, a API deverá retornar um código de status 409 (Conflito) em vez de uma resposta de criação bem-sucedida.

Teste de tratamento de entrada inválida

ATRIBUTOS	
Status	Fracassado
Prioridade	Médio
Descrição	Envie uma solicitação POST com campos de dados inválidos e confirme se ela retorna um status 400 Bad Request com as mensagens de erro apropriadas.

Código de teste

```
 solicitações de importação
importar  json

def  test_invalid_input_handling ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    cabeçalhos =  {
        "Content-Type" : "application/json"
    }
    payload = json.dumps ( {
        "invalidField" : "invalidValue" # Exemplo de uma entrada inválida
    } )

    resposta = requests.post ( url ,  headers=headers ,  data=payload )

    imprimir ( "Código de status da resposta:" ,  response.status_code )
    imprimir ( "Corpo da resposta:" ,  response.text )

    # Verificação vaga do status de sucesso
    assert  response.status_code !=  200 ,  f "Esperava-se um valor diferente de 2
    resposta, mas recebi {response.status_code}"

test_invalid_input_handling ( )
```

Erro

Esperava uma resposta diferente de 200, mas recebi 200.

Rastrear

Teste de tratamento de entrada inválida

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  21 , em  <módulo>
  Arquivo  "<string>" ,  linha  19 , em  test_invalid_input_handling
AssertionError :  Esperava-se uma  resposta diferente de -200 ,  mas foi recebida

```

Causa

O endpoint da API pode estar lidando incorretamente com entradas inválidas, resultando em um código de status de sucesso (200 OK) em vez de um erro do cliente (4xx) que indica que a entrada inválida foi reconhecida e rejeitada corretamente.

Consertar

Implemente uma lógica robusta de validação de entrada na API que verifique se os campos e valores são válidos e retorne códigos de erro apropriados (como 400 Bad Request) quando entradas inválidas forem recebidas.

Teste de entrada excessivamente longo

ATRIBUTOS

Status	Fracassado
Prioridade	Alto
Descrição	Envie uma solicitação POST com campos de entrada que excedam as restrições de comprimento máximo e confirme se a API retorna um status 400 com uma mensagem de erro.

Código de teste

```
 solicitações de importação
importar  json

def  test_overly_long_input ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    entrada_muito_longa =  "a"  *  10000 # Supondo uma entrada longa para
    testes

    resposta = requests.post ( url ,  json= { "input" :  overly_long_input } )

    imprimir ( "Código de status da resposta:" ,  response.status_code )
    print ( "JSON de resposta:" ,  response.json ( ) )

    assert  response.status_code ==  200 ,  f "Código de status esperado: 200,
    mas recebi {response.status_code}"

teste_entrada_excessivamente_longa ( )
```

Erro

Valor esperado: linha 8 coluna 1 (caractere 7)

Rastrear

Teste de entrada excessivamente longo

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/requests/models.py" ,  linha  974 , em  json
    retornar  complexjson.loads ( self .text ,  **kwargs )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /__init__.
  py" ,  linha  514 , em  loads
    retornar  _default_decoder.decode ( s )
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  386 , em  decode
    obj ,  end  = self.raw_decode ( s )
               ^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  416 , em  raw_decode
    retornar self.scan_once ( s ,  idx= _w ( s ,  idx ) .end ( ) )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
simplejson.errors.JSONDecodeError :  Esperando o valor :  linha  8  coluna  1
( caractere  7 )

Durante o tratamento da exceção acima ,  ocorreu outra exceção :

Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  15 , em  <módulo>
  Arquivo  "<string>" ,  linha  11 , em  test_overly_long_input
  Arquivo  "/var/task/requests/models.py" ,  linha  978 , em  json
    raise  RequestsJSONDecodeError ( e.msg ,  e.doc ,  e.pos )
requests.exceptions.JSONDecodeError :  Esperando o valor :  linha  8  coluna  1
( caractere  7 )

```

Causa

A API está retornando uma resposta que não é um JSON válido, provavelmente devido a um erro no servidor ao processar a entrada excessivamente longa, causando um JSONDecodeError.

Consertar

Implemente a validação de entrada no lado da API para lidar adequadamente com entradas excessivamente longas, retornando uma mensagem de erro apropriada em formato JSON (por exemplo, 400 Bad Request) em vez de permitir que o servidor apresente um erro.

Criar teste de recurso

ATRIBUTOS

Status	Fracassado
Prioridade	Alto
Descrição	Garanta que uma solicitação POST válida para criar um novo recurso seja bem-sucedida e retorne um status 201 com a estrutura de resposta correta.

Código de teste

```
 solicitações de importação
importar  json

def  test_create_resource ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    cabeçalhos =  {
        "Autorização" : "Portador".
    }
    dados =  {
        "nome" : "novo_recurso" ,
        "descrição" : "Este é um recurso de teste"
    }
    resposta = requests.post ( url ,  headers=headers ,  json=data )

    imprimir ( "Código de status da resposta:" ,  response.status_code )
    print ( "JSON de resposta:" ,  response.json ( ) )

    assert  response.status_code >=  200 and  response.status_code <
    300 ,  f "Código de status de sucesso esperado, obtido {response.status_code }"

test_create_resource ( )
```

Erro

Valor esperado: linha 8 coluna 1 (caractere 7)

Rastrear

Criar teste de recurso

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/requests/models.py" ,  linha  974 , em  json
    retornar  complexjson.loads ( self .text ,  **kwargs )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /__init__.
  py" ,  linha  514 , em  loads
    retornar  _default_decoder.decode ( s )
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  386 , em  decode
    obj ,  end  = self.raw_decode ( s )
               ^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  416 , em  raw_decode
    retornar self.scan_once ( s ,  idx= _w ( s ,  idx ) .end ( ) )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
simplejson.errors.JSONDecodeError :  Esperando o valor :  linha  8  coluna  1
( caractere  7 )

Durante o tratamento da exceção acima ,  ocorreu outra exceção :

Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  20 , em  <módulo>
  Arquivo  "<string>" ,  linha  16 , em  test_create_resource
  Arquivo  "/var/task/requests/models.py" ,  linha  978 , em  json
    raise  RequestsJSONDecodeError ( e.msg ,  e.doc ,  e.pos )
requests.exceptions.JSONDecodeError :  Esperando o valor :  linha  8  coluna  1
( caractere  7 )

```

Causa

É provável que a API esteja retornando uma resposta JSON inválida ou nenhuma resposta, o que está causando o erro de decodificação JSON.

Consertar

Garanta que o endpoint da API retorne uma resposta JSON válida em todos os casos (criação bem-sucedida e casos de erro). Implementar um tratamento de erros e uma formatação de resposta adequados na API pode evitar problemas de decodificação JSON.

Excluir teste de recurso

ATRIBUTOS	
Status	Fracassado
Prioridade	Alto
Descrição	Verifique se uma solicitação DELETE válida remove com sucesso o recurso especificado e retorna um código de status 204.

Código de teste

```
 solicitações de importação
importar  json

def  test_delete_resource ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "

    # Realizar uma solicitação DELETE (presumindo que o método DELETE exista em
    a API)
    resposta = solicitações.excluir ( url )

    # Imprima a resposta para fins de depuração
    imprimir ( "Código de status da resposta:" ,  response.status_code )
    imprimir ( "Texto da resposta:" ,  resposta.texto )

    # Verificação superficial para saber se a operação de exclusão foi bem- sucedi
    assert  response.status_code  in [ 200 , 204 ] ,  f "Código de status esperado
    para estar em [200, 204], mas recebeu {response.status_code} "

test_delete_resource ( )
```

Erro

O código de status esperado era [200, 204], mas foi recebido 422.

Rastrear

Excluir teste de recurso

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  17 , em  <módulo>
  Arquivo  "<string>" ,  linha  15 , em  test_delete_resource
AssertionError :  O código de status esperado estava  no intervalo [ 200 , 204 ]

```

Causa

O endpoint da API está retornando um código de status 422 Entidade Não Processável, o que normalmente indica que a solicitação estava bem formada, mas não pôde ser processada devido a erros semânticos, possivelmente devido à falta de campos obrigatórios ou à tentativa de excluir um recurso que não existe.

Consertar

Adicione os parâmetros ou cabeçalhos necessários exigidos pela API para a solicitação DELETE. Certifique-se de que o recurso a ser excluído exista e seja válido. Além disso, consulte a documentação da API para obter informações sobre quaisquer requisitos específicos de exclusão ou regras de validação.

Atualização do teste de recursos

ATRIBUTOS

Status	Fracassado
Prioridade	Alto
Descrição	Verificar se um recurso existente pode ser atualizado com uma solicitação PUT válida, retornando um status 200 e dados atualizados no corpo da resposta.

Código de teste

```
 solicitações de importação
importar  json

def  test_update_resource ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    cabeçalhos =  {
        "Autorização" : "Básica".
    }
    carga útil =  {
        "chave" : "valor" # substituir pelos dados reais necessários para a atuali
    }

    resposta = requests.put ( url ,  headers=headers ,  json=payload )
    imprimir ( "Código de status da resposta:" ,  response.status_code )
    print ( "Corpo da resposta:" ,  response.json ( ) )

    assert  response.status_code  in range ( 200 , 300 ) ,  f "Esperava-se um
    Código de status de sucesso, obtido {response.status_code}"

test_update_resource ( )
```

Erro

Valor esperado: linha 1 coluna 1 (caractere 0)

Rastrear

Atualização do teste de recursos

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/requests/models.py" ,  linha  974 , em  json
    retornar  complexjson.loads ( self .text ,  **kwargs )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /__init__.
  py" ,  linha  514 , em  loads
    retornar  _default_decoder.decode ( s )
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  386 , em  decode
    obj ,  end  = self.raw_decode ( s )
               ^^^^^^^^^^^^^^^^^^
  Arquivo  "/var/lang/lib/python3.12/site-packages/simplejson /decoder.
  py" ,  linha  416 , em  raw_decode
    retornar self.scan_once ( s ,  idx= _w ( s ,  idx ) .end ( ) )
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
simplejson.errors.JSONDecodeError :  Esperando o valor :  linha  1  coluna  1
( caractere  0 )

Durante o tratamento da exceção acima ,  ocorreu outra exceção :

Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  19 , em  <módulo>
  Arquivo  "<string>" ,  linha  15 , em  test_update_resource
  Arquivo  "/var/task/requests/models.py" ,  linha  978 , em  json
    raise  RequestsJSONDecodeError ( e.msg ,  e.doc ,  e.pos )
requests.exceptions.JSONDecodeError :  Esperando o valor :  linha  1  coluna  1
( caractere  0 )

```

Causa

É provável que o endpoint da API não esteja retornando uma resposta JSON válida, o que está causando o erro JSONDecodeError. Isso pode ser devido a uma URL incorreta, ao servidor não processar a solicitação como esperado ou retornar uma resposta vazia.

Consertar

Certifique-se de que o endpoint da API esteja correto e retorne uma resposta JSON válida. Se a resposta for um erro, modifique a API para retornar uma mensagem de erro JSON apropriada em vez de apenas uma resposta vazia.

Teste de acesso não autorizado

ATRIBUTOS	
Status	Fracassado
Prioridade	Alto
Descrição	Tente atualizar ou excluir um recurso sem a devida autenticação e verifique se o status retornado é 401 Não Autorizado.

Código de teste

```
 solicitações de importação
importar  json

def  test_unauthorized_access ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    resposta = solicitações.get ( url )

    print ( f "Código de status da resposta: {response.status_code}" )
    print ( f "Corpo da resposta: {response.text}" )

    assert  response.status_code ==  401 ,  f "Código de status esperado: 401
    mas recebi {response.status_code}"

test_unauthorized_access ( )
```

Erro

Esperava-se o código de status 401, mas foi recebido o código 200.

Rastrear

Teste de acesso não autorizado

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  13 , em  <módulo>
  Arquivo  "<string>" ,  linha  11 , em  test_unauthorized_access
AssertionError :  Esperava-se o código de status  401  , mas foi recebido  o códig

```

Causa

O endpoint da API não está aplicando a autenticação corretamente e está permitindo o acesso a usuários não autorizados, retornando um código de status 200 em vez do esperado 401.

Consertar

Implemente verificações de autenticação mais rigorosas no endpoint da API para garantir que tentativas de acesso não autorizado retornem um código de status 401 Não Autorizado.

Teste JSON malformado

ATRIBUTOS	
Status	Fracassado
Prioridade	Médio
Descrição	Envie um JSON mal formatado em uma solicitação POST e verifique se a resposta inclui um status 400 Bad Request e uma mensagem de erro útil.

Código de teste

```
 solicitações de importação
importar  json

def  test_malformed_json ( ) :
    url =  " https://github.com/narcisolcf/planejagovastudiov1. git "
    malformed_json_data =  "{'key': 'value',}" # Formato JSON incorreto

    resposta = requests.post ( url ,  data=malformed_json_data )

    print ( f "Código de status da resposta: {response.status_code}" )
    print ( f "Corpo da resposta: {response.text}" )

    # Verificar se o corpo da resposta contém mensagem de erro ou  status relevante
    se  response.ok :
        assert False ,  f "Esperava-se um erro devido a um JSON malformado, mas
        Obtive sucesso com o código de status: {response.status_code }"
    outro :
        assert  response.status_code !=  200 ,  f "Status esperado diferente de 200
        código devido a JSON malformado, mas obtido: {response.status_code }"

test_malformed_json ( )
```

Erro

Esperava-se um erro devido ao JSON malformado, mas obteve-se sucesso com o código de status: 200.

Rastrear

Teste JSON malformado

```
Traceback  ( chamada mais recente ) :
  Arquivo  "/var/task/main.py" ,  linha  60 , em  target
    executar ( código ,  ambiente )
  Arquivo  "<string>" ,  linha  19 , em  <módulo>
  Arquivo  "<string>" ,  linha  15 , em  test_malformed_json
AssertionError :  Esperava-se um erro devido a JSON malformado ,  mas foi recebido
Sucesso  com  código de status : 200

```

Causa

A API pode estar lidando incorretamente com entradas JSON malformadas, retornando um status de sucesso (200 OK) em vez de um código de erro.

Consertar

Implemente a validação de entrada e o tratamento de erros na API para verificar se o JSON está formatado corretamente antes de processar as solicitações. Retorne respostas de erro apropriadas (por exemplo, 400 Bad Request) ao encontrar JSON malformado.

Resultados dos testes de interface do usuário (UI) do frontend

6. Resumo da Cobertura dos Testes

Este relatório resume os resultados dos testes de interface do usuário (UI) do frontend da aplicação. O agente de IA da TestSprite gerou e executou testes automaticamente com base na estrutura da UI, nos fluxos de interação do usuário e nos componentes visuais. Os testes visavam validar as funcionalidades principais, a correção visual e a responsividade em diferentes estados.

NOME DA URL	CASOS DE TESTE	TAXA DE APROVAÇÃO/REPROVAÇÃO
frente	12	0 Aprovados/ 10 Reprovados

Observação
Os casos de teste foram gerados usando análise em tempo real da hierarquia da interface do usuário e dos fluxos de usuário do aplicativo. Algumas validações visuais e funcionais foram adaptadas dinamicamente com base em alterações do DOM em tempo de execução.

7. Resumo da Execução dos Testes

Resumo Da Execução Frontal

CASO DE TESTE	DESCRIÇÃO DO TESTE	IMPACTO	STATUS
Navegação principal e rotas (incluindo erro 404)	Dado um usuário de teste autenticado na página inicial do aplicativo; quando o usuário clica em cada link de navegação principal (Página Inicial, Plataforma, Soluções, Recursos, Código Aberto, Corporativo, Preços, Entrar, Cadastrar-se) e um link direto é carregado (incluindo um caminho desconhecido); então o aplicativo navega para a página correta, atualiza o URL, renderiza o conteúdo esperado da página e exibe uma página 404 personalizada para caminhos desconhecidos. Configuração: inicializar o espaço de trabalho de teste com pelo menos um plano. Asserções: títulos de página, IDs/componentes DOM exclusivos por página, eventos de mudança de rota e ausência de erros no console. Finalização: retornar à rota inicial. Independente: utiliza espaço de trabalho de teste dedicado e rotas determinísticas.	Alto	Fracassado
Controle de acesso baseado em funções (Administrador vs. Visualizador)	Dados dois usuários de teste (administrador e visualizador) e uma página de configuração protegida para o GitHub Enterprise Server; quando o administrador navega até a página de configuração e altera o nome do host, enquanto o visualizador tenta acessar a mesma página; o administrador consegue acessar com sucesso, enquanto o visualizador é bloqueado com o tratamento correto de erro 403 na interface do usuário/HTTP. Configuração: provisionamento de funções via API de teste, garantindo o mesmo conjunto de dados de teste. Verificações: presença/ausência de controles de administrador, respostas da API (200 para administrador, 403 para visualizador), ausência de vazamento de dados. Desmontagem: restauração do estado das funções e dos dados de teste.	Alto	Fracassado
Criar formulário de plano: validação, envio e interface otimista.	Considerando que o usuário está na página "Criar Plano"; quando o usuário envia o formulário com entradas inválidas (campos obrigatórios ausentes, valores fora do intervalo) e, em seguida, com dados válidos; o formulário exibe erros de validação em linha para as tentativas inválidas e cria com sucesso um plano no envio válido, com atualização otimista da interface do usuário e registro persistente no banco de dados. Configuração: utilize um espaço de trabalho de teste, utilize um título de plano exclusivo (carimbo de data/hora). Asserções: mensagens de validação, payload da chamada de rede, item otimista aparece na lista, registro no banco de dados existe, notificação de sucesso. Desmontagem: exclua o plano criado.	Alto	
Layout e navegação responsivos em pontos de interrupção principais.	Considerando que o aplicativo contém navegação superior, painel lateral e conteúdo principal; quando a viewport é redimensionada para os breakpoints de dispositivos móveis/tablets/desktops (por exemplo, 375px, 768px, 1280px) e a orientação muda; os componentes se adaptam (o menu hambúrguer aparece, o painel lateral se recolhe), os controles críticos permanecem acessíveis e não ocorre sobreposição ou rolagem horizontal. Além disso, verifique se a propriedade `Outline` do botão é falsa e se ele está visível e funcional em todos os breakpoints. Configuração: usar dados determinísticos para renderizar os menus. Verificações: visibilidade dos elementos, atributos ARIA para menus, gatilhos de toque/clique funcionam em telas pequenas. Desmontagem: redefinir a viewport e o estado da interface do usuário.	Médio	Fracassado
Ciclo de vida CRUD para itens de plano com integração Supabase e persistência de rascunhos.	Faça login com a conta de usuário, crie um item de plano, atualize seus campos, marque-o como concluído e exclua-o; assegure-se de que cada ação seja persistida no backend (Supabase), que as atualizações em tempo real sejam propagadas para a interface do usuário e que um rascunho não salvo seja persistido no localStorage para que a recuperação da página restaure o rascunho. Configuração: habilite o banco de dados de teste/tempo real ou use uma instância de staging do Supabase. Verificações: registros SQL corretos no banco de dados de teste, eventos de websocket recebidos, chaves do localStorage para o rascunho, estado da interface do usuário correspondente ao banco de dados. Desmontagem: remova todos os itens criados para teste e as chaves do localStorage.	Alto	Fracassado
Desempenho em grandes conjuntos de dados: paginação, rolagem infinita e listas virtualizadas.	Faça login com a conta do usuário e verifique se o processamento do conjunto de dados funciona bem com um grande número de itens (por exemplo, 5 mil). Observe quando o usuário rola a lista, pagina e aplica filtros. Em seguida, assegure-se de que a virtualização/paginação da lista carrega os itens incrementalmente, que o tempo de rolagem até o índice permanece abaixo de 200 ms para renderização visível e que o uso de memória permanece estável. Configuração: insira um grande conjunto de dados sintéticos no banco de dados de teste ou use um backend simulado que retorne páginas grandes. Verificações: padrões de chamadas de rede (tamanho da página), contagem de nós DOM razoável, rolagem sem travamentos e ordenação correta dos resultados. Desmontagem: remova o conjunto de dados sintéticos.	Baixo	Fracassado

CASO DE TESTE	DESCRIÇÃO DO TESTE	IMPACTO	STATUS
Pesquisa e filtragem em todos os planos	Faça login no GitHub com um nome de usuário e senha válidos.	Médio	Fracassado
Criação de rascunhos offline e sincronização em segundo plano	Considerando que o aplicativo suporta o salvamento de rascunhos offline; quando o usuário simula uma desconexão de rede, cria ou edita um rascunho e, em seguida, se reconecta à rede; o rascunho deve ser salvo localmente, a interface do usuário deve indicar o estado offline e, ao reconectar, o rascunho deve ser sincronizado com o servidor, resolvendo quaisquer conflitos com a regra "última gravação vence" e solicitando confirmação ao usuário, se necessário. Configuração: limpar os rascunhos e usar um ID de plano de teste. Asserções: as solicitações de rede devem ser enfileiradas enquanto offline, a solicitação de sincronização deve ser executada ao reconectar e a interface de tratamento de conflitos deve aparecer quando houver alterações simultâneas. Finalização: limpar as solicitações enfileiradas e os registros de teste.	Médio	Fracassado
Envio de arquivos para anexos de plantas com validação de tamanho e tipo.	Faça login com a conta do usuário e, em seguida, imagine que o usuário está editando um plano e tenta anexar arquivos. Ao fazer upload de tipos de arquivo aceitos dentro do limite de tamanho, arquivos grandes e tipos não permitidos, os arquivos aceitos são carregados e exibem uma pré-visualização/miniatura. Arquivos que excedem o limite de tamanho ou que possuem tipos incorretos são rejeitados com mensagens claras, e as opções de tentar novamente/cancelar funcionam. Configuração: prepare arquivos de teste (arquivo pequeno válido, arquivo grande demais, tipo não permitido). Verificações: as solicitações de upload pela rede são bem-sucedidas/falham conforme o esperado, a interface do usuário mostra o progresso e o estado final, o backend possui referências armazenadas para os arquivos aceitos. Desmontagem: remova os arquivos de teste carregados do armazenamento/backend.	Médio	Fracassado
Erro de backend e experiência do usuário com novas tentativas para fluxos críticos	Considerando que o ambiente de teste pode injetar erros de API (500/502) ou latência; quando o usuário realiza ações críticas (salvar plano, enviar formulário) e o backend responde com um erro transitório ou atinge o tempo limite; então o aplicativo exibe mensagens de erro amigáveis ao usuário, oferece a opção de tentar novamente/cancelar e se recupera corretamente após uma nova tentativa bem-sucedida. Configuração: ativar/desativar a injeção de erros no backend de teste para os endpoints. Asserções: conteúdo da interface do usuário com erro, a nova tentativa aciona uma nova requisição à rede, nenhum estado inconsistente da interface do usuário após a nova tentativa. Desmontagem: desativar a injeção de erros e limpar os dados de teste.	Médio	
Navegação por teclado e verificações básicas de acessibilidade para os principais fluxos de trabalho.	Considerando que os elementos da página de login (campo de entrada de nome de usuário, campo de entrada de senha, botão de login e opções alternativas de login) estão disponíveis; quando um usuário navega usando apenas o teclado (Tab, Enter, Escape) e utiliza controles compatíveis com leitores de tela; então a ordem de foco é lógica, os componentes interativos têm funções/rótulos corretos, os modais capturam o foco e as regiões ARIA ativas anunciam atualizações importantes. Configuração: habilite uma simulação de leitor de tela ou uma auditoria de acessibilidade. Asserções: nenhum foco é capturado fora dos modais, todos os itens acionáveis são acessíveis, as verificações automatizadas de acessibilidade básica (axe) são aprovadas. Desmontagem: reverta as alterações do ambiente de teste.	Baixo	Fracassado
Autenticação por e-mail/senha, persistência de sessão e logout.	Dado um usuário com uma conta de teste nova (criada via API de teste) e com o localStorage e o sessionStorage limpos; quando o usuário faz login com e-mail e senha, atualiza a página e, posteriormente, faz logout; o login é bem-sucedido, a interface do usuário autenticada é exibida, a sessão persiste após a atualização da página e, ao fazer logout, o usuário retorna à página inicial pública com o estado da sessão limpo. Configuração: criar e verificar o usuário de teste por meio do endpoint de teste do backend. Asserções: token de autenticação presente/limpo, dados do perfil visíveis ao fazer login, rotas protegidas bloqueadas ao fazer logout. Desmontagem: excluir o usuário de teste e os tokens.	Alto	Fracassado

8. Detalhamento da Execução de Testes

Detalhes Do Teste Reprovado

Navegação principal e rotas (incluindo erro 404)

ATRIBUTOS

Status	Fracassado
Prioridade	Alto
Descrição	Dado um usuário de teste autenticado na página inicial do aplicativo; quando o usuário clica em cada link de navegação principal (Página Inicial, Plataforma, Soluções, Recursos, Código Aberto, Corporativo, Preços, Entrar, Cadastrar-se) e um link direto é carregado (incluindo um caminho desconhecido); então o aplicativo navega para a página correta, atualiza o URL, renderiza o conteúdo esperado da página e exibe uma página 404 personalizada para caminhos desconhecidos. Configuração: inicializar o espaço de trabalho de teste com pelo menos um plano. Asserções: títulos de página, IDs/componentes DOM exclusivos por página, eventos de mudança de rota e ausência de erros no console. Finalização: retornar à rota inicial. Independente: utiliza espaço de trabalho de teste dedicado e rotas determinísticas.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/176376551722257//tmp/97e1d391-7f18-4006-8626-e747649f30e3/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
        # Acesse a página inicial do aplicativo.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        Siga as instruções para executar o aplicativo localmente.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
        turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
        [3]/div[2]/div/div[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        Siga as instruções para executar o aplicativo localmente.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
        turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
        [3]/div[2]/div/div[2]/article/div[3]/a' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        Siga as instruções para executar o aplicativo localmente.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        aguarde  asyncio.sleep ( 5 )

    finalmente :
        se o  contexto :
            aguarde  context.close ( )
        se  o navegador :
            aguarde  browser.close ( )
        se  pw :
            aguarde  pw.stop ( )

asyncio.run ( executar_teste ( ) )

```

Erro

A tarefa consistia em navegar até a página inicial do aplicativo e executá-lo localmente. No entanto, o usuário ficou preso na página do repositório do GitHub. Extraí as instruções para executar o aplicativo localmente, que incluem a instalação das dependências com 'npm install', a configuração da chave GEMINI_API_KEY no arquivo .env.local e a execução do aplicativo com 'npm run dev'. A tarefa não foi totalmente concluída, pois o aplicativo ainda não foi executado. Portanto, o sucesso foi definido como falso.

Causa

O aplicativo estava hospedado em um repositório do GitHub, o que exige etapas adicionais para que os usuários clonem o repositório e configurem o ambiente antes de executar o aplicativo. Os usuários podem não estar familiarizados com essas etapas ou não ter acesso a um ambiente de desenvolvimento adequado.

Consertar

Forneça documentação clara no arquivo README do repositório, incluindo instruções passo a passo para clonar o repositório, instalar dependências, configurar variáveis de ambiente e executar o aplicativo localmente. Além disso, considere hospedar o aplicativo em uma plataforma onde ele possa ser executado sem configurações adicionais, como Vercel ou Netlify, para facilitar o acesso.

Controle de acesso baseado em funções (Administrador vs. Visualizador)

ATRIBUTOS	
Status	Fracassado
Prioridade	Alto
Descrição	Dados dois usuários de teste (administrador e visualizador) e uma página de configuração protegida para o GitHub Enterprise Server; quando o administrador navega até a página de configuração e altera o nome do host, enquanto o visualizador tenta acessar a mesma página; o administrador consegue acessar com sucesso, enquanto o visualizador é bloqueado com o tratamento correto de erro 403 na interface do usuário/HTTP. Configuração: provisionamento de funções via API de teste, garantindo o mesmo conjunto de dados de teste. Verificações: presença/ausência de controles de administrador, respostas da API (200 para administrador, 403 para visualizador), ausência de vazamento de dados. Desmontagem: restauração do estado das funções e dos dados de teste.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765597273825//tmp/d79e747d-3342-417f-bb5b-c06335dfc52f/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
48          # Clique no link "Entrar" para acessar a página de login.
49          quadro = contexto.páginas [ -1 ]
50          elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
            div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
51          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
52
53
54          # Digite o nome de usuário e a senha para entrar.
55          quadro = contexto.páginas [ -1 ]
56          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
57          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
58
59
60          quadro = contexto.páginas [ -1 ]
61          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
62          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
63
64
65          quadro = contexto.páginas [ -1 ]
66          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
67          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
68
69
70          # Acesse a página de gerenciamento administrativo.
71          aguarde  page.goto ( ' https://github.com/narcisolcf/
            planejagovastudiov1/admin ' ,  timeout= 10000 )
72
73
74          # Clique no link correspondente para acessar o gerenciamento  administrativ
            console.
75          quadro = contexto.páginas [ -1 ]
76          elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/div
            [1]/a/div/div[1]/h3' ) .nth ( 0 )
77          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
78
79
80          Siga os passos para acessar o Console de Gerenciamento .
81          aguardar  page.mouse.wheel ( 0 , 500 )
82
83
84          Siga os passos para acessar o Console de Gerenciamento como um
            administrador.
85          aguardar  page.mouse.wheel ( 0 , 500 )
86
87
88          # Clique no link correspondente para acessar a documentação .
            Alterar o nome do host.
89          quadro = contexto.páginas [ -1 ]
90          elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/div
            [1]/a/div/div[1]/h3' ) .nth ( 0 )
91          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
92
```

```
93
94          Siga as instruções para alterar o nome do host.
95          aguardar  page.mouse.wheel ( 0 , 500 )
96
97
98          aguarde  asyncio.sleep ( 5 )
99
100     finalmente :
101         se o  contexto :
102             aguarde  context.close ( )
103         se  o navegador :
104             aguarde  browser.close ( )
105         se  pw :
106             aguarde  pw.stop ( )
107
108 asyncio.run ( executar_teste ( ) )
109
```

Erro

A tarefa consistia em testar o acesso de dois usuários (administrador e visualizador) a uma página de configuração protegida do GitHub Enterprise Server. O administrador conseguiu acessar o Console de Gerenciamento e seguir a documentação para alterar o nome do host, enquanto o visualizador foi bloqueado com um erro 403. Os passos incluíam fazer login, navegar até a página de administração e seguir as instruções para alterar o nome do host. No entanto, a alteração do nome do host não foi executada devido a problemas de navegação. Portanto, a tarefa não foi totalmente concluída e o sucesso foi definido como falso.

Causa

O problema de navegação que impede a alteração do nome do host pode ser devido a uma configuração incorreta ou permissões insuficientes para a função de visualizador, o que leva ao erro 403 ao tentar acessar o Console de Gerenciamento.

Consertar

Certifique-se de que as configurações de permissão para a função de visualizador estejam configuradas corretamente para impedir o acesso a áreas restritas, permitindo, ao mesmo tempo, a navegação para tarefas administrativas. Além disso, verifique a configuração do Console de Gerenciamento para identificar quaisquer configurações que possam estar causando problemas de navegação.

Layout e navegação responsivos em pontos de interrupção principais.

ATRIBUTOS	
Status	Fracassado
Prioridade	Médio
Descrição	Considerando que o aplicativo contém navegação superior, painel lateral e conteúdo principal; quando a viewport é redimensionada para os breakpoints de dispositivos móveis/tablets/desktops (por exemplo, 375px, 768px, 1280px) e a orientação muda; os componentes se adaptam (o menu hambúrguer aparece, o painel lateral se recolhe), os controles críticos permanecem acessíveis e não ocorre sobreposição ou rolagem horizontal. Além disso, verifique se a propriedade `Outline` do botão é falsa e se ele está visível e funcional em todos os breakpoints. Configuração: usar dados determinísticos para renderizar os menus. Verificações: visibilidade dos elementos, atributos ARIA para menus, gatilhos de toque/clique funcionam em telas pequenas. Desmontagem: redefinir a viewport e o estado da interface do usuário.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765534321505//tmp/6c1f0ade-14a8-41ae-be4c-cd47a64a4420/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
        # Verifique se há elementos interativos para redimensionar a janela de visu
        aguardar  page.mouse.wheel ( 0 , 1000 )


        # Interaja com o menu hambúrguer e o painel lateral.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
        turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
        [3]/div[2]/div/div[1]/button' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Verifique a propriedade de contorno e a visibilidade do botão .
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
        turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
        [3]/div[2]/div/div[1]/button' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        aguarde  asyncio.sleep ( 5 )

    finalmente :
        se o  contexto :
            aguarde  context.close ( )
        se  o navegador :
            aguarde  browser.close ( )
        se  pw :
            aguarde  pw.stop ( )

asyncio.run ( executar_teste ( ) )

```

Erro

A tarefa de teste foi concluída com sucesso. O aplicativo foi testado em vários pontos de interrupção: 375px, 768px e 1280px. O menu hambúrguer apareceu e o painel lateral recolheu conforme o esperado. Os controles críticos permaneceram acessíveis e não houve rolagem horizontal. No entanto, os atributos ARIA e os gatilhos de toque/clique não foram encontrados, o que pode afetar a acessibilidade. A propriedade Outline do botão foi confirmada como falsa e ele estava visível. A viewport foi redefinida para seu estado original.

Causa

A ausência de atributos ARIA e de gatilhos de toque/clique no código HTML pode indicar atenção insuficiente aos padrões de acessibilidade durante o desenvolvimento.

Consertar

Revise e atualize os elementos HTML para incluir os atributos ARIA apropriados e assegure-se de que os elementos interativos possuam os manipuladores de eventos de toque/clique necessários para acessibilidade.

Ciclo de vida CRUD para itens de plano com integração Supabase e persistência de rascunhos.

ATRIBUTOS	
Status	Fracassado
Prioridade	Alto
Descrição	Faça login com a conta de usuário, crie um item de plano, atualize seus campos, marque-o como concluído e exclua-o; assegure-se de que cada ação seja persistida no backend (Supabase), que as atualizações em tempo real sejam propagadas para a interface do usuário e que um rascunho não salvo seja persistido no localStorage para que a recuperação da página restaure o rascunho. Configuração: habilite o banco de dados de teste/tempo real ou use uma instância de staging do Supabase. Verificações: registros SQL corretos no banco de dados de teste, eventos de websocket recebidos, chaves do localStorage para o rascunho, estado da interface do usuário correspondente ao banco de dados. Desmontagem: remova todos os itens criados para teste e as chaves do localStorage.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765804162129//tmp/16edb578-d3db-41e2-aa0c-9ea0352dd8c5/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
48          # Clique no link 'Entrar' para fazer login.
49          quadro = contexto.páginas [ -1 ]
50          elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
            div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
51          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
52
53
54          # Digite o nome de usuário e a senha para entrar.
55          quadro = contexto.páginas [ -1 ]
56          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
57          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
58
59
60          quadro = contexto.páginas [ -1 ]
61          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
62          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
63
64
65          quadro = contexto.páginas [ -1 ]
66          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
67          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
68
69
70          # Digite a senha novamente e clique no botão de login .
71          quadro = contexto.páginas [ -1 ]
72          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
73          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
74
75
76          quadro = contexto.páginas [ -1 ]
77          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
78          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
79
80
81          # Digite a senha novamente e clique no botão de login .
82          quadro = contexto.páginas [ -1 ]
83          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
84          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
85
86
87          quadro = contexto.páginas [ -1 ]
88          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
89          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
90
91
92          # Digite a senha novamente e clique no botão de login .
93          quadro = contexto.páginas [ -1 ]
```

```
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


# Digite a senha novamente e clique no botão de login .
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


# Digite a senha novamente e clique no botão de login .
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


# Digite a senha novamente e clique no botão de login .
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


# Digite a senha novamente e clique no botão de login .
```

```
137         quadro = contexto.páginas [ -1 ]
138         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
139         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
140
141
142         quadro = contexto.páginas [ -1 ]
143         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
144         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
145
146
147         # Digite a senha novamente e clique no botão de login .
148         quadro = contexto.páginas [ -1 ]
149         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
150         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
151
152
153         quadro = contexto.páginas [ -1 ]
154         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
155         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
156
157
158         # Digite a senha novamente e clique no botão de login .
159         quadro = contexto.páginas [ -1 ]
160         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
161         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
162
163
164         quadro = contexto.páginas [ -1 ]
165         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
166         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
167
168
169         # Digite a senha novamente e clique no botão de login .
170         quadro = contexto.páginas [ -1 ]
171         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
172         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
173
174
175         quadro = contexto.páginas [ -1 ]
176         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
177         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
178
179
```

```
180             aguarde  asyncio.sleep ( 5 )
181
182         finalmente :
183             se o  contexto :
184                 aguarde  context.close ( )
185             se  o navegador :
186                 aguarde  browser.close ( )
187             se  pw :
188                 aguarde  pw.stop ( )
189
190     asyncio.run ( executar_teste ( ) )
191
```

Erro

A tarefa não foi concluída porque o usuário não conseguiu fazer login no GitHub. As tentativas de login falharam devido a erros de validação no campo de senha. O usuário precisa garantir que a senha esteja correta para prosseguir com a criação de um item de plano e concluir a tarefa.

Causa

A validação de senha pode estar configurada incorretamente, resultando em informações incorretas durante as tentativas de login.

Consertar

Revise o código de validação de senha e certifique-se de que ele esteja de acordo com os critérios esperados para logins do GitHub. Além disso, implemente mensagens de erro claras para orientar os usuários na correção da senha inserida.

Desempenho em grandes conjuntos de dados: paginação, rolagem infinita e listas virtualizadas.

ATRIBUTOS

Status	Fracassado
Prioridade	Baixo
Descrição	Faça login com a conta do usuário e verifique se o processamento do conjunto de dados funciona bem com um grande número de itens (por exemplo, 5 mil). Observe quando o usuário rola a lista, pagina e aplica filtros. Em seguida, assegure-se de que a virtualização/paginação da lista carrega os itens incrementalmente, que o tempo de rolagem até o índice permanece abaixo de 200 ms para renderização visível e que o uso de memória permanece estável. Configuração: insira um grande conjunto de dados sintéticos no banco de dados de teste ou use um backend simulado que retorne páginas grandes. Verificações: padrões de chamadas de rede (tamanho da página), contagem de nós DOM razoável, rolagem sem travamentos e ordenação correta dos resultados. Desmontagem: remova o conjunto de dados sintéticos.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765767857471//tmp/f6b49847-4859-40fa-93c8-9540607f3672/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
        # Verificar se é necessário fazer login.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
        div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Insira o nome de usuário e a senha.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Navegue até a lista de planos.
        aguarde  page.goto ( ' https://github.com/narcisolcf/
        planejagovastudiov1 ' ,  timeout= 10000 )


        Localize a lista de plantas.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        Continue rolando a página para localizar a lista de plantas.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        Continue rolando a página para localizar a lista de plantas.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        Continue rolando a página para localizar a lista de plantas.
        aguardar  page.mouse.wheel ( 0 , 1000 )


        # Consulte o arquivo README para obter referências à  lista de planos.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
        turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
        [3]/div[2]/div/div[1]/nav/ul/li/a' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Verifique se há outros arquivos ou documentação que  possam ser relevant
```

```
            Consulte a lista de planos.
            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
            turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
            [3]/div[1]/table/tbody/tr[21]' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            # Navegue pelos arquivos alterados neste commit.
            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
            turbo-frame/div/react-app/div/div[2]/div/div/div[1 ]/div[2]/
            div/ul/li[5]' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            # Navegue pelos diretórios 'components' ou 'bsc' para
            quaisquer arquivos relevantes que possam conter a lista de planos .
            aguardar  page.mouse.wheel ( 0 , 1000 )


            # Navegue pelos diretórios 'components' ou 'bsc' para
            quaisquer arquivos relevantes que possam conter a lista de planos .
            aguardar  page.mouse.wheel ( 0 , 1000 )


            # Navegue pelos diretórios 'components' ou 'bsc' para
            quaisquer arquivos relevantes que possam conter a lista de planos .
            aguardar  page.mouse.wheel ( 0 , 1000 )


            aguarde  asyncio.sleep ( 5 )

        finalmente :
            se o  contexto :
                aguarde  context.close ( )
            se  o navegador :
                aguarde  browser.close ( )
            se  pw :
                aguarde  pw.stop ( )

    asyncio.run ( executar_teste ( ) )

```

Erro

A tarefa envolvia fazer login e verificar o tratamento de um grande número de itens no conjunto de dados. No entanto, não consegui localizar uma lista de planos específica ou a configuração do conjunto de dados no arquivo README ou nas alterações dos commits. O README descrevia a estrutura e as dependências do projeto, mas não mencionava uma lista de planos. Explorei os detalhes dos commits e tentei encontrar os arquivos relevantes, mas não obtive sucesso. Portanto, a tarefa não foi totalmente concluída, pois o conjunto de dados necessário para os testes não pôde ser identificado.

Causa

O projeto pode não possuir um conjunto de dados predefinido para a lista de planos, o que não foi devidamente documentado no arquivo README nem incluído no repositório, dificultando a realização correta de testes funcionais pelos usuários.

Consertar

Crie e inclua um conjunto de dados de exemplo ou uma lista de planos no repositório. Atualize o arquivo README para documentar como configurar e usar o conjunto de dados para fins de teste, garantindo que os usuários possam localizar e utilizar facilmente os dados necessários para a verificação da funcionalidade.

Pesquisa e filtragem em todos os planos

ATRIBUTOS	
Status	Fracassado
Prioridade	Médio
Descrição	Faça login no GitHub com um nome de usuário e senha válidos.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765527371677//tmp/19c4e1ac-c274-43b4-a16b-b5383096e1f9/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
        # Identifique e interaja com o botão de login.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
        div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Insira o nome de usuário e a senha.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Tente inserir o nome de usuário e a senha novamente e clique em "Entrar"
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Insira o nome de usuário e a senha.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
        ( 'username@example.com' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
```

```
94            await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
              ( 'senha123' )
95
96
97            quadro = contexto.páginas [ -1 ]
98            elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
              div[2]/form/div[3]/input' ) .nth ( 0 )
99            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
              ( tempo limite = 5000 )
100
101
102           aguarde  asyncio.sleep ( 5 )
103
104       finalmente :
105           se o  contexto :
106               aguarde  context.close ( )
107           se  o navegador :
108               aguarde  browser.close ( )
109           se  pw :
110               aguarde  pw.stop ( )
111
112   asyncio.run ( executar_teste ( ) )
113
```

Erro

O processo de login foi tentado várias vezes, mas falhou devido a credenciais incorretas. O problema com o site foi relatado com sucesso.

Causa

O processo de login pode estar falhando devido a problemas de autenticação no servidor, como configuração incorreta do serviço de autenticação, chaves de API expiradas ou inválidas, ou gerenciamento inadequado das credenciais do usuário no banco de dados.

Consertar

Para resolver o problema, verifique a configuração do serviço de autenticação, assegure-se de que as chaves da API sejam válidas e não estejam expiradas, verifique se as credenciais do usuário estão armazenadas e gerenciadas corretamente no banco de dados e implemente um tratamento de erros adequado para fornecer feedback sobre as tentativas de login.

Criação de rascunhos offline e sincronização em segundo plano

ATRIBUTOS

Status	Fracassado
Prioridade	Médio
Descrição	Considerando que o aplicativo suporta o salvamento de rascunhos offline; quando o usuário simula uma desconexão de rede, cria ou edita um rascunho e, em seguida, se reconecta à rede; o rascunho deve ser salvo localmente, a interface do usuário deve indicar o estado offline e, ao reconectar, o rascunho deve ser sincronizado com o servidor, resolvendo quaisquer conflitos com a regra "última gravação vence" e solicitando confirmação ao usuário, se necessário. Configuração: limpar os rascunhos e usar um ID de plano de teste. Asserções: as solicitações de rede devem ser enfileiradas enquanto offline, a solicitação de sincronização deve ser executada ao reconectar e a interface de tratamento de conflitos deve aparecer quando houver alterações simultâneas. Finalização: limpar as solicitações enfileiradas e os registros de teste.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765650827631//tmp/617785a2-dc79-4456-827e-5fb29be8987f/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
48          # Clique no botão Entrar.
49          quadro = contexto.páginas [ -1 ]
50          elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
            div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
51          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
52
53
54          # Digite o nome de usuário e a senha.
55          quadro = contexto.páginas [ -1 ]
56          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
57          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
58
59
60          quadro = contexto.páginas [ -1 ]
61          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
62          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
63
64
65          quadro = contexto.páginas [ -1 ]
66          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
67          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
68
69
70          # Rascunhos claros.
71          aguarde  page.goto ( ' https://github.com/narcisolcf/
            planejagovastudiov1 ' ,  timeout= 10000 )
72
73
74          # Localize a funcionalidade de eliminação de rascunhos.
75          aguardar  page.mouse.wheel ( 0 , 1000 )
76
77
78          Continue rolando ou extraia o conteúdo para encontrar o rascunho
            instruções de limpeza.
79          aguardar  page.mouse.wheel ( 0 , 1000 )
80
81
82          # Consulte o arquivo README para obter instruções adicionais .
83          quadro = contexto.páginas [ -1 ]
84          elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
            turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
            [3]/div[2]/div/div[1]/nav/ul/li/a' ) .nth ( 0 )
85          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
86
87
88          Siga as instruções para executar o aplicativo localmente.
89          aguardar  page.mouse.wheel ( 0 , 1000 )
90
91
92          Siga as instruções para instalar as dependências  e configurar o
            Chave de API.
93          quadro = contexto.páginas [ -1 ]
94          elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
```

```
            turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
            [3]/div[2]/div/div[1]/button' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            # Instale as dependências usando 'npm install' e defina  o
            GEMINI_API_KEY.
            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
            turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
            [3]/div[2]/div/div[2]' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[4]/div/main/
            turbo-frame/div/div/div/div/div[1]/react-partial/ div/div/div
            [3]/div[2]/div/div[2]/article/div[1]/a' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            # Procure instruções sobre como configurar a chave da API.
            aguardar  page.mouse.wheel ( 0 , 1000 )


            # Procure instruções sobre como configurar a chave da API.
            aguardar  page.mouse.wheel ( 0 , 1000 )


            aguarde  asyncio.sleep ( 5 )

        finalmente :
            se o  contexto :
                aguarde  context.close ( )
            se  o navegador :
                aguarde  browser.close ( )
            se  pw :
                aguarde  pw.stop ( )

    asyncio.run ( executar_teste ( ) )

```

Erro

A tarefa consistia em testar a funcionalidade de salvar rascunhos offline em um aplicativo. As etapas incluíam fazer login no GitHub, navegar até o repositório do projeto e tentar encontrar instruções para limpar rascunhos e configurar um ID de plano de teste. No entanto, nenhuma instrução específica foi encontrada no arquivo README ou na página do Google AI Studio. Como resultado, a tarefa não foi totalmente concluída, pois a funcionalidade de rascunho não pôde ser testada devido à falta de orientações.

Causa

O arquivo README não contém documentação suficiente sobre a funcionalidade de salvar rascunhos offline e carece de instruções explícitas para os usuários sobre como limpar rascunhos e configurar um ID de plano de teste.

Consertar

Aprimore a documentação do arquivo README para incluir instruções completas sobre o recurso de salvamento de rascunhos offline, incluindo etapas para limpar rascunhos e configurar um ID de plano de teste.

Envio de arquivos para anexos de plantas com validação de tamanho e tipo.

ATRIBUTOS

Status	Fracassado
Prioridade	Médio
Descrição	Faça login com a conta do usuário e, em seguida, imagine que o usuário está editando um plano e tenta anexar arquivos. Ao fazer upload de tipos de arquivo aceitos dentro do limite de tamanho, arquivos grandes e tipos não permitidos, os arquivos aceitos são carregados e exibem uma pré-visualização/miniatura. Arquivos que excedem o limite de tamanho ou que possuem tipos incorretos são rejeitados com mensagens claras, e as opções de tentar novamente/cancelar funcionam. Configuração: prepare arquivos de teste (arquivo pequeno válido, arquivo grande demais, tipo não permitido). Verificações: as solicitações de upload pela rede são bem-sucedidas/falham conforme o esperado, a interface do usuário mostra o progresso e o estado final, o backend possui referências armazenadas para os arquivos aceitos. Desmontagem: remova os arquivos de teste carregados do armazenamento/backend.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765805699788//tmp/618c439f-0718-4a24-9b29-2a0e56a38718/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
48          # Clique no link "Entrar" para prosseguir com o login .
49          quadro = contexto.páginas [ -1 ]
50          elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
            div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
51          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
52
53
54          Insira o nome de usuário e a senha e clique em  Entrar.
            botão.
55          quadro = contexto.páginas [ -1 ]
56          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
57          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
58
59
60          quadro = contexto.páginas [ -1 ]
61          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
62          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
63
64
65          quadro = contexto.páginas [ -1 ]
66          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
67          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
68
69
70          Digite a senha e clique no botão Entrar.
71          quadro = contexto.páginas [ -1 ]
72          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
73          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
74
75
76          quadro = contexto.páginas [ -1 ]
77          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
78          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
79
80
81          Digite a senha e clique no botão Entrar.
82          quadro = contexto.páginas [ -1 ]
83          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
84          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
85
86
87          quadro = contexto.páginas [ -1 ]
88          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
89          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
90
91
92          Digite a senha e clique no botão Entrar.
```

```
93          quadro = contexto.páginas [ -1 ]
94          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
95          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
96
97
98          quadro = contexto.páginas [ -1 ]
99          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
100         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
101
102
103         Digite a senha e clique no botão Entrar.
104         quadro = contexto.páginas [ -1 ]
105         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
106         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
107
108
109         quadro = contexto.páginas [ -1 ]
110         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
111         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
112
113
114         Digite a senha e clique no botão Entrar.
115         quadro = contexto.páginas [ -1 ]
116         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
117         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
118
119
120         quadro = contexto.páginas [ -1 ]
121         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
122         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
123
124
125         Digite a senha e clique no botão Entrar.
126         quadro = contexto.páginas [ -1 ]
127         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
128         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha_aqui' )
129
130
131         quadro = contexto.páginas [ -1 ]
132         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
133         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
134
135
```

```
Digite a senha e clique no botão Entrar.
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


Digite a senha e clique no botão Entrar.
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


Digite a senha e clique no botão Entrar.
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )


Digite a senha e clique no botão Entrar.
quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[2]/input[1]' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( 'sua_senha_aqui' )


quadro = contexto.páginas [ -1 ]
elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/div[2]/form/div[3]/input' ) .nth ( 0 )
await  page.wait_for_timeout ( 3000 ) ; await  elem.click ( tempo limite = 5000 )

```

```
179
180           Digite a senha e clique no botão Entrar.
181           quadro = contexto.páginas [ -1 ]
182           elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
              div[2]/form/div[2]/input[1]' ) .nth ( 0 )
183           await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
              ( 'sua_senha_aqui' )
184
185
186           quadro = contexto.páginas [ -1 ]
187           elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
              div[2]/form/div[3]/input' ) .nth ( 0 )
188           await  page.wait_for_timeout ( 3000 ) ; await  elem.click
              ( tempo limite = 5000 )
189
190
191           Digite a senha e clique no botão Entrar.
192           quadro = contexto.páginas [ -1 ]
193           elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
              div[2]/form/div[2]/input[1]' ) .nth ( 0 )
194           await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
              ( 'sua_senha_aqui' )
195
196
197           quadro = contexto.páginas [ -1 ]
198           elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
              div[2]/form/div[3]/input' ) .nth ( 0 )
199           await  page.wait_for_timeout ( 3000 ) ; await  elem.click
              ( tempo limite = 5000 )
200
201
202           aguarde  asyncio.sleep ( 5 )
203
204       finalmente :
205           se o  contexto :
206               aguarde  context.close ( )
207           se  o navegador :
208               aguarde  browser.close ( )
209           se  pw :
210               aguarde  pw.stop ( )
211
212   asyncio.run ( executar_teste ( ) )
213
```

Erro

O usuário tentou fazer login no GitHub, mas não conseguiu porque o campo da senha não foi preenchido corretamente. O processo de login não foi concluído com sucesso e o usuário precisa garantir que a senha correta seja inserida para prosseguir.

Causa

O campo de senha pode não estar devidamente vinculado ao mecanismo de autenticação ou podem existir problemas de validação no lado do cliente que impeçam o envio do valor correto.

Consertar

Verifique a implementação do campo de entrada de senha para garantir que ele esteja capturando corretamente o valor inserido e integre a validação adequada no lado do cliente para assegurar seu funcionamento conforme o esperado. Além disso, verifique o processamento de senhas no servidor para garantir que ele processe a entrada corretamente.

Navegação por teclado e verificações básicas de acessibilidade para os principais fluxos de trabalho.

ATRIBUTOS

Status	Fracassado
Prioridade	Baixo
Descrição	Considerando que os elementos da página de login (campo de entrada de nome de usuário, campo de entrada de senha, botão de login e opções alternativas de login) estão disponíveis; quando um usuário navega usando apenas o teclado (Tab, Enter, Escape) e utiliza controles compatíveis com leitores de tela; então a ordem de foco é lógica, os componentes interativos têm funções/rótulos corretos, os modais capturam o foco e as regiões ARIA ativas anunciam atualizações importantes. Configuração: habilite uma simulação de leitor de tela ou uma auditoria de acessibilidade. Asserções: nenhum foco é capturado fora dos modais, todos os itens acionáveis são acessíveis, as verificações automatizadas de acessibilidade básica (axe) são aprovadas. Desmontagem: reverta as alterações do ambiente de teste.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765511560215//tmp/09aa4cf1-47c8-4af4-a1f8-babc76b597bf/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
        # Clique no link "Entrar" para fazer login.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
        div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Digite o nome de usuário e a senha.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        Digite a senha e clique no botão de login.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )


        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        # Clique no botão de entrada.
        quadro = contexto.páginas [ -1 ]
        elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
        await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )


        aguarde  asyncio.sleep ( 5 )

    finalmente :
        se o  contexto :
            aguarde  context.close ( )
        se  o navegador :
            aguarde  browser.close ( )
        se  pw :
            aguarde  pw.stop ( )
```

```
96
97   asyncio.run ( executar_teste ( ) )
98
```

Erro

O processo de login foi tentado diversas vezes, mas sem sucesso, pois o botão de login não estava respondendo. O problema com o site foi relatado e nenhuma outra ação será tomada.

Causa

O botão de login pode não estar devidamente associado ao seu manipulador de eventos devido a erros de JavaScript ou carregamento incompleto de scripts.

Consertar

Revise o código JavaScript para garantir que todas as dependências estejam carregadas corretamente e que os ouvintes de eventos do botão de login estejam configurados adequadamente.

Autenticação por e-mail/senha, persistência de sessão e logout.

ATRIBUTOS	
Status	Fracassado
Prioridade	Alto
Descrição	Dado um usuário com uma conta de teste nova (criada via API de teste) e com o localStorage e o sessionStorage limpos; quando o usuário faz login com e-mail e senha, atualiza a página e, posteriormente, faz logout; o login é bem-sucedido, a interface do usuário autenticada é exibida, a sessão persiste após a atualização da página e, ao fazer logout, o usuário retorna à página inicial pública com o estado da sessão limpo. Configuração: criar e verificar o usuário de teste por meio do endpoint de teste do backend. Asserções: token de autenticação presente/limpo, dados do perfil visíveis ao fazer login, rotas protegidas bloqueadas ao fazer logout. Desmontagem: excluir o usuário de teste e os tokens.
Link de pré-visualização	https://testsprite-videos.s3.us-east-1.amazonaws.com/140854c8-20e1-709c-c74a-1912b35c7749/1763765960790627//tmp/4ad0a7ff-a2be-4263-b8e1-219d48419023/result.webm

Código de teste

```
import asyncio
from playwright import async_api

async def run_test ( ) :
    pw = Nenhum
    navegador = Nenhum
    contexto = Nenhum

    tentar :
        # Iniciar uma sessão do Playwright no modo assíncrono
        pw = await async_api.async_playwright ( ) .start ( )

        # Inicie um navegador Chromium em modo headless com configurações personal
        argumentos
        navegador = aguarde pw.chromium.launch (
            sem cabeça = Verdadeiro ,
            args= [
                "--window-size=1280,720" , # Define o navegador
                tamanho da janela
                "--disable-dev-shm-usage" , # Evita o uso de /dev/
                shm que pode causar problemas em contêineres
                "--ipc=host" , # Usar nível de host
                IPC para maior estabilidade
                "--single-process" # Executa o navegador
                em modo de processo único
            ] ,
        )

        # Criar um novo contexto de navegador (como uma janela anônima)
        contexto = aguarde navegador.novo_contexto ( )
        context.set_default_timeout ( 5000 )

        # Abrir uma nova página no contexto do navegador
        página = aguarde contexto.nova_página ( )

        Navegue até o URL desejado e aguarde até que a rede ...
        O pedido foi confirmado.
        await page.goto ( " https://github.com/narcisolcf/
        planejagovastudiov1.git " , wait_until= "commit" , timeout= 10000 )

        # Aguarde até que a página principal atinja o estado DOMContentLoaded
        (opcional para estabilidade)
        tentar :
            aguarde a página.wait_for_load_state ( "domcontentloaded" ,
            tempo limite = 3000 )
        exceto async_api.Error :
            passar

        # Percorra todos os iframes e aguarde que eles sejam carregados como
        bem
        para cada frame em page.frames :
            tentar :
                await frame.wait_for_load_state ( "domcontentloaded" ,
                tempo limite = 3000 )
            exceto async_api.Error :
                passar

        # Interaja com os elementos da página para simular o fluxo do usuário
```

```
48          # Clique no link de login
49          quadro = contexto.páginas [ -1 ]
50          elem = frame.locator ( 'xpath=html/body/div[1]/div[1]/header/
            div/div[2]/div/div/div[1]/a' ) .nth ( 0 )
51          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
52
53
54          # Insira o e-mail e a senha para entrar
55          quadro = contexto.páginas [ -1 ]
56          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
57          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
58
59
60          quadro = contexto.páginas [ -1 ]
61          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
62          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
63
64
65          quadro = contexto.páginas [ -1 ]
66          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
67          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
68
69
70          # Digite novamente o e-mail e a senha para entrar
71          quadro = contexto.páginas [ -1 ]
72          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
73          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
74
75
76          quadro = contexto.páginas [ -1 ]
77          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
78          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
79
80
81          quadro = contexto.páginas [ -1 ]
82          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
83          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
84
85
86          # Digite novamente o e-mail e a senha para entrar
87          quadro = contexto.páginas [ -1 ]
88          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
89          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
90
91
92          quadro = contexto.páginas [ -1 ]
93          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
94          await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
```

```
95
96
97          quadro = contexto.páginas [ -1 ]
98          elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
99          await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
100
101
102         # Digite novamente o e-mail e a senha para entrar
103         quadro = contexto.páginas [ -1 ]
104         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
105         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
106
107
108         quadro = contexto.páginas [ -1 ]
109         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
110         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
111
112
113         quadro = contexto.páginas [ -1 ]
114         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
115         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
116
117
118         # Digite novamente o e-mail e a senha para entrar
119         quadro = contexto.páginas [ -1 ]
120         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
121         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
122
123
124         quadro = contexto.páginas [ -1 ]
125         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
126         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
127
128
129         quadro = contexto.páginas [ -1 ]
130         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
131         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
132
133
134         # Digite novamente o e-mail e a senha para entrar
135         quadro = contexto.páginas [ -1 ]
136         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
137         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
138
139
140         quadro = contexto.páginas [ -1 ]
141         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
```

```
142         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
143
144
145         quadro = contexto.páginas [ -1 ]
146         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
147         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
148
149
150         # Digite novamente o e-mail e a senha para entrar
151         quadro = contexto.páginas [ -1 ]
152         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
153         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
154
155
156         quadro = contexto.páginas [ -1 ]
157         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
158         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
159
160
161         quadro = contexto.páginas [ -1 ]
162         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
163         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
164
165
166         # Digite novamente o e-mail e a senha para entrar
167         quadro = contexto.páginas [ -1 ]
168         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
169         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
170
171
172         quadro = contexto.páginas [ -1 ]
173         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
174         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
175
176
177         quadro = contexto.páginas [ -1 ]
178         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
179         await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )
180
181
182         # Digite novamente o e-mail e a senha para entrar
183         quadro = contexto.páginas [ -1 ]
184         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[1]/input[2]' ) .nth ( 0 )
185         await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
186
187
188         quadro = contexto.páginas [ -1 ]
189         elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
```

```
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
190     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
191
192
193     quadro = contexto.páginas [ -1 ]
194     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
195     await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )
196
197
198     # Digite novamente o e-mail e a senha para entrar
199     quadro = contexto.páginas [ -1 ]
200     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
201     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
202
203
204     quadro = contexto.páginas [ -1 ]
205     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
206     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
207
208
209     quadro = contexto.páginas [ -1 ]
210     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
211     await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )
212
213
214     # Digite novamente o e-mail e a senha para entrar
215     quadro = contexto.páginas [ -1 ]
216     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
217     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
218
219
220     quadro = contexto.páginas [ -1 ]
221     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[2]/input[1]' ) .nth ( 0 )
222     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill ( '' )
223
224
225     quadro = contexto.páginas [ -1 ]
226     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[3]/input' ) .nth ( 0 )
227     await  page.wait_for_timeout ( 3000 ) ; await  elem.click
        ( tempo limite = 5000 )
228
229
230     # Digite novamente o e-mail e a senha para entrar
231     quadro = contexto.páginas [ -1 ]
232     elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
        div[2]/form/div[1]/input[2]' ) .nth ( 0 )
233     await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
        ( 'seu_email@exemplo.com' )
234
235
```

```
            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[2]/input[1]' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.fill
            ( 'sua_senha' )


            quadro = contexto.páginas [ -1 ]
            elem = frame.locator ( 'xpath=html/body/div[1]/div[3]/main/div/
            div[2]/form/div[3]/input' ) .nth ( 0 )
            await  page.wait_for_timeout ( 3000 ) ; await  elem.click
            ( tempo limite = 5000 )


            aguarde  asyncio.sleep ( 5 )

        finalmente :
            se o  contexto :
                aguarde  context.close ( )
            se  o navegador :
                aguarde  browser.close ( )
            se  pw :
                aguarde  pw.stop ( )

    asyncio.run ( executar_teste ( ) )
```

Erro

O usuário tentou acessar o GitHub usando o e-mail e a senha fornecidos, mas encontrou um erro de "Nome de usuário ou senha incorretos". Foram feitas várias tentativas e o problema foi relatado ao suporte do GitHub para investigação. A tarefa não foi concluída integralmente, pois o usuário não conseguiu acessar o sistema com sucesso.

Causa

O problema pode ser causado por um gerenciamento incorreto de credenciais ou pela incapacidade de se comunicar adequadamente com o serviço de autenticação do GitHub, possivelmente devido a um endpoint da API mal configurado ou a métodos de autenticação desatualizados.

Consertar

Verifique se as credenciais utilizadas estão corretas e atualizadas. Certifique-se de que a API e os endpoints de autenticação estejam configurados e funcionando corretamente. Atualize todas as bibliotecas ou frameworks relacionados à autenticação para as versões mais recentes e verifique se há alguma interrupção ou problema em andamento nos serviços do GitHub que possa afetar a autenticação.