


Code review

login_tests.robot

- **Revisor:** Squad 5 - Thaís do Amaral
- **Data:** 18/06/2025
- **Autor do código:** Squad 6 - Cavalheiros da Qualidade
- **API:** ServeRest
- **Link do repositório:**  [GitHub - Rodrigo-Matuz/robot_framework_serverest](#)
- **Branch:** main

Pontos Positivos

1. **Documentação clara** - Explica bem o propósito do arquivo
2. **Organização de recursos** - Separação adequada em arquivos distintos para variáveis e keywords
3. **Setup/Teardown** - Configuração padrão para abrir/fechar o navegador
4. **Fluxo completo** - Testa desde cadastro até login, o que é bom para fluxo E2E
5. **Reutilização de keywords** - Utiliza keywords de outros recursos de forma adequada
6. **Variáveis** - Armazena credenciais em variáveis de forma correta
7. **Cenário negativo** - Bom teste de caso negativo
8. **Verificação** - Verifica a mensagem de erro corretamente
9. **Validação de campos obrigatórios** - Importante teste de validação
10. **Múltiplas verificações** - Verifica ambos os campos

Pontos de Melhoria

1. **Adicionar tags** - Poderia incluir tags para categorizar os testes (ex: `[Login]` , `[Regressão]`)
2. **Timeout global** - Poderia adicionar `Suite Setup/Teardown` se necessário para a suíte toda
3. **Dependência do cadastro** - O teste de login depende do sucesso do cadastro, o que pode ser problemático se o cadastro falhar
4. **Dados dinâmicos** - Não está claro se o email é gerado dinamicamente (deve-se evitar dados fixos)
5. **Verificação** - Poderia incluir mais verificações após o login (ex: redirecionamento para página correta)
6. **Seletor CSS** - Poderia usar um localizador mais descritivo (ex: `css=[data-test='error-message']`)
7. **Dados inválidos** - Não está claro como o email inválido é gerado (deveria ser explícito no teste)
8. **Consistência** - Um dos checks usa `Get Text` enquanto outros usam `Wait For Elements State` (cenário login com campo vazio)
9. **Mensagens hardcoded** - As mensagens deveriam vir de variáveis para facilitar manutenção
10. **Variável não utilizada** - `${erro_senha}` é capturada mas não usada

Recomendações Gerais

1. **Padronização de Locators:**
 - Usar estratégias consistentes para localizar elementos (`data-test-id`, `text`, `css`)
 - Considerar criar variáveis para os locators mais usados
2. **Dados de Teste:**
 - Garantir que emails sejam únicos em cada execução (usar timestamp ou random)

- Criar um arquivo de variáveis para dados de teste

3. **Asserções:**

- Adicionar mais verificações após ações críticas
- Verificar não apenas mensagens mas também estados da aplicação

4. **Organização:**

- Adicionar tags para categorizar testes
- Considerar separar testes positivos e negativos em suites diferentes

5. **Tratamento de Erros:**

- Adicionar tratamento para falhas nas keywords chamadas
- Considerar screenshots em caso de falha

6. **Documentação:**

- Adicionar documentação para cada caso de teste explicando o cenário
- Documentar pré-condições quando aplicável

Conclusão

O arquivo está bem estruturado e cobre os principais cenários de login. As principais melhorias seriam em padronização, tratamento de dados dinâmicos e documentação mais detalhada. A separação em arquivos de recursos está adequada e facilita a manutenção. Parabéns pela excelente entrega!!!


Evidência da execução do teste

```
$ robot -d results tests/login_tests.robot
[ WARN ] Error in file 'C:\Users\thais\robot_framework_serverest\resources\casas
tro_keywords.resource' on line 39: The '[Return]' setting is deprecated. Use the
'RETURN' statement instead.

=====
Login Tests :: Casos de teste para o Login de usuários
=====
Login com sucesso | PASS |
Login com email invalido | PASS |
Login com senha inválida | PASS |
Login com campos vazios | PASS |
Login Tests :: Casos de teste para o Login de usuários | PASS |
4 tests, 4 passed, 0 failed
=====
```

Observação: Utilizei DeepSeek para consulta nesse code review.

Product_test

- **Revisor:** Squad 5 - Raique Alfredo
- **Data:** 18/06/2025
- **Autor do código:** Squad 6 - Cavalheiros da Qualidade
- **API:** ServeRest
- **Link do repositório:**  [GitHub - Rodrigo-Matuz/robot_framework_serverest](https://github.com/Rodrigo-Matuz/robot_framework_serverest)
- **Branch:** main
- **Test Cases analisados:**
 - Cadastrar produto no sistema
 - Pesquisar produto e verificar resultados

Pontos Positivos [🔗](#)

Organização: [🔗](#)

- A estrutura dos arquivos está muito bem organizada. Separar os **test cases** em `tests` e as **keywords** em `resources` reflete um entendimento sólido das melhores práticas para modularidade e manutenção de código.

Boas Práticas: [🔗](#)

- A utilização de um arquivo central de variáveis (`env_variables.resource`) foi uma escolha excelente. Isso facilita a reutilização e a alteração de dados sensíveis ou mutáveis.
- A abordagem de criar uma `common_keywords.resource` para funções compartilhadas, como abrir o navegador, demonstra um bom uso de abstração e reaproveitamento de código.

Funcionalidade e Fluxo: [🔗](#)

- O código cobre cenários importantes para a aplicação ServeRest, como o cadastro e a busca de produtos, utilizando dados dinâmicos das variáveis.
- As keywords são bem nomeadas, claras e seguem um padrão consistente, o que ajuda na leitura e entendimento do propósito de cada uma.

Pontos de Melhoria [🔗](#)

Cobertura de Testes: [🔗](#)

1. **Validações Explícitas:** Na keyword "**Criar Produto com sucesso como administrador**", seria interessante validar a resposta ou o estado final da aplicação (por exemplo, verificar a presença do produto na lista após o cadastro).

Padrões e Consistência: [🔗](#)

1. **Timeouts:** Os timeouts utilizados nas ações (`Wait For Elements State`) poderiam ser extraídos para uma variável global, garantindo consistência e facilidade de ajuste futuro.
2. **Mensagens de Erro:** Adicionar assertivas ou verificações que retornem mensagens personalizadas quando algo não ocorre como esperado ajudaria a identificar problemas rapidamente.

Pontos de Melhoria Relacionados aos Erros [🔗](#)

Timeout nos Locators: [🔗](#)

1. Descrição do Problema:

Nos testes que utilizam a keyword `Pesquisar produto`, foi reportado um `TimeoutError` ao tentar localizar o elemento `[data-testid="pesquisar"]`. Isso indica que o elemento não ficou visível no tempo esperado.

2. Possíveis Causas:

- O seletor CSS `[data-testid="pesquisar"]` pode estar incorreto ou não corresponde a um elemento carregado na página atual.
- O tempo de espera configurado (`timeout=3s`) pode ser insuficiente, especialmente em ambientes onde o carregamento da página pode ser mais lento.
- O elemento pode não estar sendo renderizado devido a erros na navegação para a URL correta ou a problemas com a aplicação ServeRest.

3. Recomendações:

- **Verifique o seletor CSS:** Confirme se `[data-testid="pesquisar"]` é o identificador correto para o campo de busca na aplicação ServeRest.
- **Aumente o tempo de espera:** Ajuste o `timeout` para um valor maior, como `timeout=10s`, para garantir que o elemento tenha tempo suficiente para ser carregado.

- **Adicione validação de navegação:** Inclua uma validação para garantir que a página correta foi carregada antes de executar os testes, utilizando um elemento único da página como referência.
- **Teste manualmente a aplicação:** Verifique se o campo de busca está presente e funcional na aplicação ServeRest, descartando problemas de infraestrutura ou configuração local.

Sugestões de Melhorias Adicionais

1. Refatoração das Keywords:

- **Pesquisar Produto:** Adicionar validações ao final da keyword para garantir que os resultados retornados são condizentes com o termo de busca.


Conclusão:

- O código está muito bem estruturado e segue boas práticas de organização e reutilização. A separação de responsabilidades entre **test cases**, **keywords** e **variáveis** está clara e consistente, refletindo um entendimento sólido de desenvolvimento com Robot Framework.
- Os problemas identificados parecem ser majoritariamente relacionados à visibilidade dos elementos e ao tempo de espera. Com as correções sugeridas, como a validação dos seletores, ajustes nos timeouts e a inclusão de mensagens personalizadas nos logs, é provável que os testes sejam executados com mais consistência e eficiência.
- As sugestões apresentadas têm como objetivo melhorar a clareza, a manutenção, a cobertura do projeto e a rastreabilidade dos testes. Parabéns pelo excelente trabalho e dedicação ao projeto!

Evidência da execução do teste:

```
=====
Product Tests :: Testes de busca de produtos e adição ao carrinho
=====
Cadastrar produto no sistema                                     | PASS |
-----
Pesquisar produto e verificar resultados                         | FAIL |
TimeoutError: locator.waitFor: Timeout 3000ms exceeded.
Call log:
  - waiting for locator('[data-testid="pesquisar"]') to be visible
=====
```

Observação: Utilizei ChatGPT para consulta nesse code review.

- **Revisor:** Squad 5 - Izadora Santos
- **Data:** 18/06/2025
- **Autor do código:** Squad 6 - Cavalheiros da Qualidade
- **API:** ServeRest
- **Link do repositório:**  [GitHub - Rodrigo-Matuz/robot_framework_serverest](https://github.com/Rodrigo-Matuz/robot_framework_serverest)
- **Branch:** main

Observações Gerais

O código analisado refere-se a testes automatizados desenvolvidos com Robot Framework para validação da funcionalidade de busca, adição e exclusão de produtos na aplicação. Os cenários abordam a verificação da exclusão de produtos e a adição ao carrinho com a exibição de uma mensagem de funcionalidade em construção.

As keywords estão organizadas em um resource separado e os testes estão descritos de forma clara, porém há alguns pontos de instabilidade e melhorias

Erros: Não consegue encontrar o input de pesquisa

```
Fill Text      css=[data-testid="pesquisar"]      ${search_term}
```

```
=====
Pesquisar produto e verificar resultados                               | FAIL |
TimeoutError: locator.waitFor: Timeout 3000ms exceeded.
Call log:
  - waiting for locator('[data-testid="pesquisar"]') to be visible
=====
Verificar se o produto foi excluído com sucesso                       | FAIL |
TimeoutError: locator.waitFor: Timeout 3000ms exceeded.
Call log:
  - waiting for locator('[data-testid="pesquisar"]') to be visible
=====
Adicionar produto ao carrinho e verificar mensagem em construção    | FAIL |
TimeoutError: locator.waitFor: Timeout 3000ms exceeded.
Call log:
  - waiting for locator('[data-testid="pesquisar"]') to be visible
=====
Tests.Product Tests :: Testes de busca de produtos e adição ao car... | FAIL |
4 tests, 1 passed, 3 failed
=====
Tests                                                                | FAIL |
11 tests, 7 passed, 4 failed
=====
```

Talvez não consegue achar o input de pesquisa porque esta fazendo o login como adm.

Observação: Utilizei ChatGPT para ajuda nesse code review.

cadastro_tests.robot [🔗](#)

- **Revisor:** Squad 5 - Gabriel Lobo
- **Data:** 19/06/2025
- **Autor do código:** Squad 6 - Cavalheiros da Qualidade
- **API:** ServeRest
- **Link do repositório:** [GitHub - Rodrigo-Matuz/robot_framework_serverest](#)
- **Branch:** main

Pontos Positivos [🔗](#)

1. A estrutura do arquivo `test.cadastro.robot` foi bem organizada, com separação clara de recursos e keywords e o uso correto de `Test Setup` e `Teardown`.
2. Os testes cobriram cenários muito importantes relacionados ao cadastro de usuários, sendo um cenário feliz e um cenário triste.
3. Utilização de keywords externas, incentivando o reuso e garantindo uma melhor legibilidade do código.

Sugestão de Melhoria [🔗](#)

1. Considerar a adição de tags nos testes para facilitar o agrupamento e execução por tipo.

Conclusão [🔗](#)

O código está funcional e atende aos cenários propostos, contém uma boa estruturação e clareza. Apresentou um excelente conhecimento sobre o uso da tecnologia Robot Framework e fez a utilização de boas práticas na elaboração e escrita do teste. Ótimo trabalho da Squad 6!

Evidência da execução do teste [↗](#)

```
=====
Cadastro Tests :: Casos de teste para o Cadastro de usuários
=====
Cadastro com sucesso | PASS |
=====
Tentar cadastrar usuário com e-mail já existente | PASS |
=====
Cadastro Tests :: Casos de teste para o Cadastro de usuários | PASS |
2 tests, 2 passed, 0 failed
=====
Output: C:\Users\vitor\Desktop\robot_framework_serverest\logs\output.xml
Log: C:\Users\vitor\Desktop\robot_framework_serverest\logs\log.html
Report: C:\Users\vitor\Desktop\robot_framework_serverest\logs\report.html
PS C:\Users\vitor\Desktop\robot_framework_serverest> 
```

Observação: Foi realizado o uso da ferramenta Chat GPT para realização do Code Review.