Compass UOL

Relatório de Teste

SERVEREST

Relatório de planejamento de teste, apresentado ao Evangelista Matheus Domingos Locatelli, como parte das exigências do challenge.

Recife
2022

Sumário

Relatório de testes da API SERVEREST

1. INTRODUCÃO

1.1. Objetivo

Este Relatório de Avaliação do Teste descreve os resultados dos testes da API SERVEREST em termos de cobertura de teste e análise de possíveis falhas, bugs e tratamentos.

1.2. Escopo

Este Relatório de Avaliação do Teste é aplicado a API SERVEREST. Os testes conduzidos estão descritos no Plano de Teste previamente explanados e segue em anexo junto este. Este Relatório de Avaliação deve ser utilizado para o seguinte:

- avaliar a capacidade de aceitação e apropriação do(s) comportamento(s) de desempenho da API;
- avaliar a capacidade de aceitação dos testes;
- identificar aprimoramentos para aumentar a cobertura do teste e/ou sua qualidade.

2. Resumo de Resultados de Testes

Os casos de teste definidos no Conjunto de Teste para a API foram executados seguindo a estratégia de teste definida no Plano de Teste.

A Cobertura de teste (consulte a seção 4.0 do plano de teste) em termos de cobertura dos casos de uso e requisitos de teste definidos no Plano de Teste foi concluída.

A análise dos testes (conforme mostrado no postman) indica que há problemas **significativos de bugs e tratamentos não cobertos na API**. Os testes foram baseados inicialmente na documentação fornecida e moldados a necessidade da regra de negócios, com isso fora detectado em alguns pontos, como na rota login falhas de tratamento e bugs relacionado à aspectos não tratados. A Equipe de desenvolvimento

gerenciara os recursos para avaliar adicionalmente esses resultados de teste e determinar alternativas.

3. Cobertura do Teste

Os testes a serem executados na API estão definidos na seção 4 do Plano de Teste e no ServeRest postman collection (aplicado ao Postman), juntamente com seus critérios de conclusão. Os resultados da cobertura do teste são as seguintes:

Taxa de Casos de Teste Executados = 58/58 = 100%

Taxa de validações executadas = 368/381 = 88,18%

Taxa de Casos de Teste com Êxito = 38/58 = 65,51%

4. Análise de testes

Esta seção resume os resultados da análise gerada utilizando o Postman, conforme o documento ServeRest.postman_test_run anexado. A seção 5 recomenda ações para tratar as descobertas da análise.

Como resultado da aplicação dos testes conseguimos constatar algumas falhas, bugs e aspectos não tratados na API e/ou no Swagger, gerando um total de 2,39% em erros, conforme abaixo.

4.1. /login

- CT-03 - **Realizar login com email e senha vazia**: Esperava-se status code 422 com tratamento apropriado, mas retorna 400 com tratamento: “email e senha em branco.”
- CT-04 - **Realizar login com email em formato invalido**: Modificar na API o tratamento para email inválido, conforme documentação.
- CT-05 - **Realizar login com login e/ou senha errada**: Ao tentar realizar login no servidor com email e senha errada obtivemos o status code 401 e não o esperado conforme documentação Swagger, status code 400;

- CT-06 - **Realizar login com request diferente do padrão**: Modificar na API o tratamento para a requisição diferente do esperado, conforme documentação. Esperava-se o status code 400, mas retornou o 500.
- CT-07 - **Realizar login sem os campos obrigatórios**: Modificar na API o tratamento para requisições sem um body enviado.

4.2. /usuários

- CT-17 - **Cadastrar usuário - sem email**: Falta acrescentar o tratamento ao Swagger no caso de ausência de email na requisição, ao status code 400;
- CT-18 - **Cadastrar usuário - email invalido**: Falta acrescentar o tratamento ao Swagger no caso de email invalido na requisição ao status code 400;
- CT-19 - **Cadastrar usuário - sem body**: Esperava-se “Os campos são obrigatórios.” como resposta.
- CT-20 - **Buscar usuário por ID - Com ID errado**: Esperava-se o status code 404, mas retorna o 400.
- CT-21 - **Editar usuário - email já cadastrado**: Esperava-se o status code 422, mas retorna o 400.
- CT-30 – **Cadastrar usuário – usuário já cadastrado**: Esperava-se encontrar erro 422, mas retorno 400.
- CT-23– **Excluir usuário – Usuário não encontrado**: Esperava-se encontrar o status code 404, mas retornou 200.

4.3. /produtos

- CT-33 - **Cadastrar produto - Quantidade negativa**: Falta acrescentar o tratamento ao Swagger no caso de criar produto com quantidade negativa, ao status code 400.
- CT-34 - **Cadastrar produto - preço sendo menor que zero**: Falta acrescentar o tratamento ao Swagger no caso de criar produto com valor inferior a zero, no status code 400;
- CT-39 – E**xcluir produto – Nenhum excluído**: Esperava-se o status code 404, mas retornou 200.
- CT-30 – **Cadastrar produto – produto já cadastrado**: Esperava-se encontrar erro 422, mas retorno 400.
- CT-35 - **Buscar produto - Não encontrado**: Esperava-se o status code 404, mas retornou 400.

4.4. /carrinhos

- CT-53 – **Cadastrar carrinho - quantidade de produtos negativa**: Falta acrescentar ao Swagger e modificar o tratamento na API para ser mais informativo;
- CT-50 - **Cadastrar carrinho - Produto não encontrado**: Esperava-se o status code 404, mas o retornado foi 400.
- CT-49 - **Cadastrar carrinho - Mais de 1 carrinho**: Esperava-se o status code 422, mas retornou 400.
- CT-54 - **Buscar carrinho por ID - Não encontrado**: Esperava-se o status code 404, mas o retornado foi 400.
- CT-57 - **Excluir carrinho (concluir-compra) - Carrinho não encontrado**: Esperava-se o status code 404, mas o retornado foi 200
- CT-58 - **Excluir carrinho e retornar produtos para estoque - Não foi encontrado carrinho**: Esperava-se o status code 404, mas o retornado foi 200

5. Ações Sugeridas

As ações recomendadas são as seguintes:

5.1. /login

- CT-03 – Modificar na API e acrescentar ao Swagger o tratamento do status code para 422, pois os campos email e senha são obrigatórios na API, logo o servidor compreende a solicitação, que não há erro na sintaxe, mas não pode processá-la devido a erros semânticos;
- CT-04 – Modificar na API o tratamento para “Email e/ou senha inválidos”, conforme swagger;
- CT-05 – Acrescentar no Swagger o tratamento do status code 401 ao digitar login e/ou senha errada e modificar o status code na API de 400 para 401;
- CT-06 – Corrigir status code para 400. O fato é que o usuário da sua API tentou fazer uma requisição e não conseguiu completá-la por falta de aspas, erro de sintaxe. O tratamento também terá como retorno a resposta: “Adicione aspas em todos os valores.”. Este ponto também serve para tratamento nos demais endpoints;

- CT-07 – Modificar na API o tratamento para “Email e/ou senha são obrigatórios na requisição.”

5.2. /usuário

- CT-17 – Adicionar o tratamento já presente na API ao Swagger: Email não pode ficar em branco ao status code 400;
- CT-18 – Adicionar o tratamento já presente na API ao Swagger: Email deve ser email válido;
- CT-19 - Modificar o tratamento presente na API para: “Os campos são obrigatórios.” e acrescentar ao Swagger no status code 400;
- CT-20 – Modificar o tratamento presente na API para o status code 404 e acrescentar no swagger, pois a busca por um id não cadastrado remeta a ideia de não encontrar, logo o status code 404 é mais coerente.
- CT-21 – Modificar o tratamento presente na API para o status code 422 e acrescentar no swagger, pois o servidor compreender a solicitação, mas não pode executá-la.
- CT-30 – Modificar tratamento na API e acrescentar ao Swagger.

  O código 422 descreve que o servidor compreende a solicitação, que não há erro na sintaxe, mas não pode processá-la. Ou seja, acredito que o código de status supramencionado deva ser o mais indicado para responder à questão porque se encaixa nos seguintes pontos:

  - não há um erro do usuário explicito na sintaxe da requisição (ou seu conteúdo)
  - o objetivo do pedido não é modificar o estado de um recurso existente
  - o servidor compreende o que lhe foi enviado, mas não pode concluir a operação
  - o uso deste código não tem "obrigatoriedade" com WebDAV ou qualquer sistema de versão

- CT-23 – Modificar tratamento na API e acrescentar ao Swagger. O status esperado deveria ser 404, pois a solicitação não foi encontrada para ser atendida.

5.3. /produtos

- CT-33 – Acrescentar ao Swagger no status code 400 o tratamento: “Quantidade deve ser maior ou igual a 0.”;
- CT-34 – Acrescentar ao Swagger no status code 400 o tratamento: “Preço deve ser um número positivo.”.
- CT-39 – Modificar tratamento na API e acrescentar ao Swagger. A solicitação em termos sintáticos está certa, mas não foi encontrado no sistema o item solicitado, logo o status seria 404.

- CT-30 - Modificar tratamento na API e acrescentar ao Swagger.

  O código 422 descreve que o servidor compreende a solicitação, que não há erro na sintaxe, mas não pode processá-la. Ou seja, acredito que o código de status supramencionado deva ser o mais indicado para responder à questão porque se encaixa nos seguintes pontos:

  - não há um erro do usuário explicito na sintaxe da requisição (ou seu conteúdo)
  - o objetivo do pedido não é modificar o estado de um recurso existente
  - o servidor compreende o que lhe foi enviado, mas não pode concluir a operação
  - o uso deste código não tem "obrigatoriedade" com WebDAV ou qualquer sistema de versão
- CT-35 - Modificar tratamento na API e acrescentar ao Swagger. A solicitação em termos sintáticos está certa, mas não foi encontrado no sistema o item solicitado, logo o status seria 404.

5.4. /carrinhos

- CT-53 – Acrescentar o tratamento ao status code 400 no Swagger: “Quantidade de produtos deve ser um número positivo.” E modificar o tratamento presente na API para: “Quantidade de produtos deve ser um número positivo.”.
- CT-50 - Modificar tratamento na API e acrescentar ao Swagger. A solicitação em termos sintáticos está certa, mas não foi encontrado no sistema o item solicitado, logo o status seria 404.
- CT-49 - Modificar tratamento na API e acrescentar ao Swagger o status 422.

  Vale salientar para este ponto que deveria ter um modo de merge dos carrinhos ou informar ao cliente que deveria ser acrescentado itens ao carrinho atual.

- CT-54 - Modificar o tratamento presente na API para o status code 404 e acrescentar no swagger, pois a busca por um id não cadastrado remeta a ideia de não encontrar, logo o status code 404 é mais coerente.
- CT-57 e CT-58 – Ambos casos precisam das mesmas modificações para tratamento na API e acrescentar ao Swagger. A solicitação em termos sintáticos está certa, mas não foi encontrado no sistema o item solicitado, logo o status seria 404.

6. Pontos críticos

- CT-49 - **Cadastrar carrinho - Mais de 1 carrinho:** Podemos considerar esta validação como pronto crítico, pois podemos perder clientes que não podem querer refazer seu carrinho, devido ao tratamento que temos de não dar merge ou simplesmente ser mais informativo em seu tratamento de direcionar o cliente a

acrescentar mais itens ao carrinho, caso seja essa vontade.

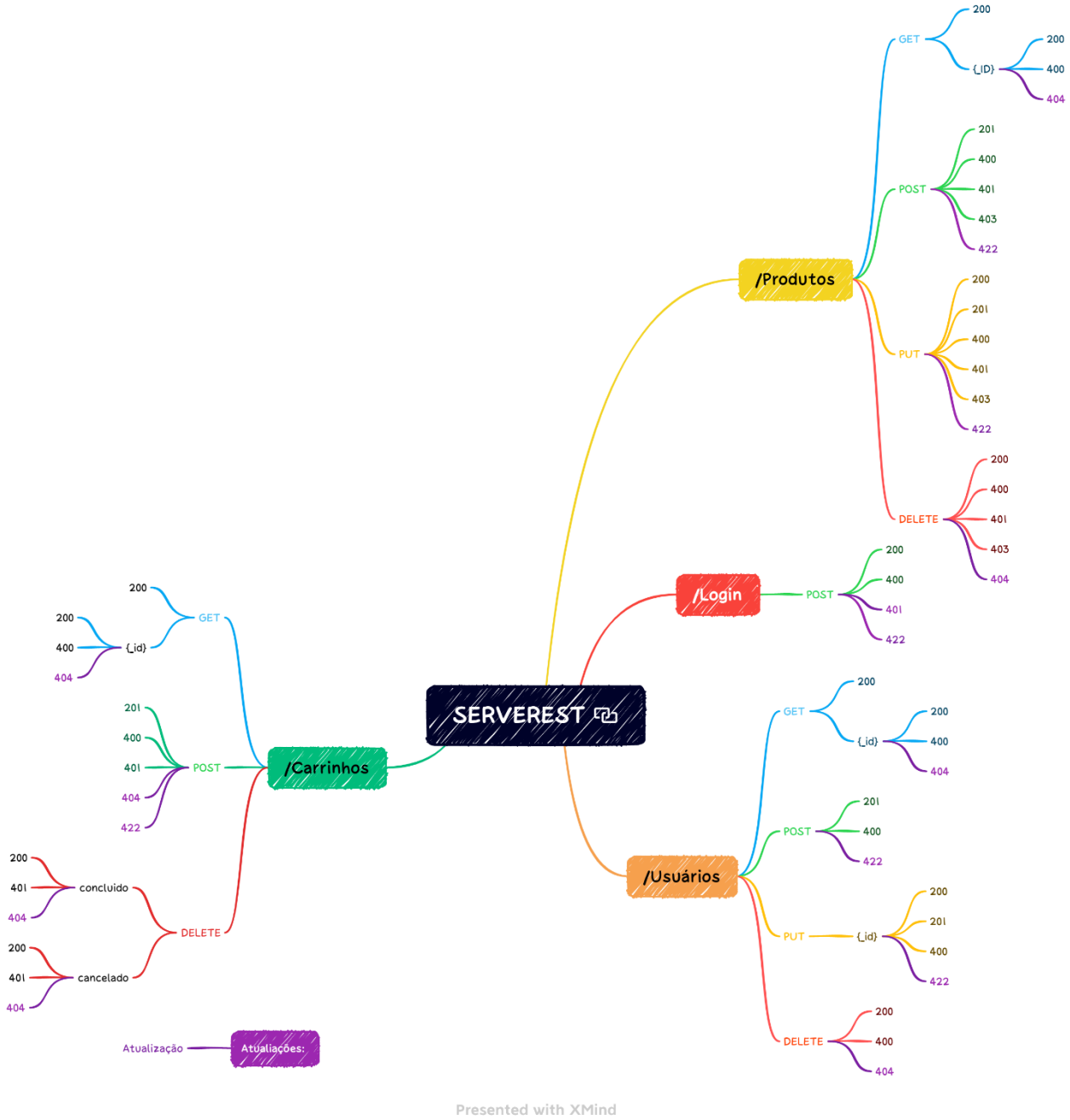
- **Falta de padrão em senhas e ausência de limites de tentativas de login**: Não possui um padrão de segurança aceitável para criação de senhas. O fato é que, podemos criar a senha de “N” maneiras, mas não tem presente um molde básico, por exemplo de 6 dígitos com limite de tentativas para dificultar a quebra de senha por cracks.

  Por tanto a recomendação seria a criação de senha ser em uma padrão de 8 dígitos contendo números, símbolos, letras maiúsculas e minúsculas e o login com limite de tentativas de login de 5 vezes com tempo de espera progressivo e acumulativo de 30 segundos com bloqueio de tentativas de 24hrs após as 5 tentativas.
- **Cadastro e login obrigatório antes de cadastrar itens ao carrinho**: Ao analisar alguns sites e-commerce conseguimos constatar que em sua maioria os produtos podem ser adicionados aos carrinhos antes mesmo de logar ou precisar cadastrar no site, diga-se de passagem, causa uma sensação mais satisfatória, por quanto pode aumentar as chances de vendas dos produtos.

  A recomendação nesse ponto seria rever a regra de negócios para tentar adequar aos padrões atuais visando a lucratividade.
- **Tempos de resposta**: Na seção 8 deste material você conseguira ter um vislumbre de teste de performance efetuado com mil usuários ao mesmo tempo em simulação de interações simultâneas, através do Apache JMeter. O ponto a ser visado aqui seria o tempo de resposta para alguns endpoints, pois apresentaram elevações significativas onde podem ser futuros pontos de retrabalho.

  Para entendimento mais detalhado visite a seção 8.

7. Mapa mental atualizado



8. Teste de performance

Esta seção resume os resultados do teste de performance gerados no Apache JMeter, conforme o documento Performance JMeter anexado.

Label	# Samples	Median	90% Line	Min	Max	Error %	Received KB/sec	Sent KB/sec
Cadastrar usuário	1000	396	578	86	685	869	6042	3274
Realizar login	1000	388	573	364	580	0	8148	2532
Editar usuário	1000	550	615	369	659	1000	5731	3212
Buscar usuários	1000	643	801	371	845	0	843453	1605
Buscar usuários por ID	1000	723	808	400	848	0	6695	1807
Buscar produtos	1000	2	5	0	30	0	257835	1736
Cadastrar produto	1000	735	1231	474	1499	997	5753	5414
Editar produto	1000	903	1210	470	1311	998	5916	5730
Buscar produtos por ID	1000	53	121	0	142	0	7337	2031
Buscar carrinhos	1000	2	13	1	58	0	391544	1833
Cadastrar carrinho	1000	1025	1237	648	1287	999	6668	5411
Buscar carrinhos por ID	1000	29	44	0	69	0	9263	2140
Excluir carrinho	1000	571	718	389	1027	0	6438	4808
Excluir produto	1000	502	596	298	620	220	6792	5046
Excluir usuário	1000	312	330	104	358	0	6981	2465
TOTAL	15000	495	908	0	1499	338,87	974294	29132

A tabela acima conseguimos identificar uma incursão de mil usuários simulados simultaneamente em interação com a API, resultando em traços significativos performando na casa dos milissegundos.

Com as interações conseguimos constatar alguns pontos com “0” milissegundos de reposta a uma taxa de 0% de erro, ao exemplo da rota buscar produtos por ID, contudo podemos ver o contrates em cadastrar produto com seus 1499 milissegundos de resposta a uma taxa de erro de 997 milissegundos, o que, diga-se de passagem, não é muito grande, mas o aceitável seria algo entre 0.1 segundo (100 milissegundos) e 1 segundo (1000 milissegundos).

A variação de tempo pode causar diferentes reações, por exemplo de 0.1 segundo podemos sentir a sensação de resposta instantânea e de 1 segundo conseguirmos manter a mesma linha de raciocínio. Veja mais abaixo:

1. **De 0 a 0.1s = Perfeito.** O usuário se sente no comando e a experiência é gratificante.

2. **De 0.1 a 1.0s = Aceitável,** desde que haja uma indicação do que está ocorrendo.

3. **De 1.0 a 10s = Problemático.** Aqui sim o feedback é vital. Mas provavelmente o usuário tentará outra atividade.

4. **mais de 10s = Inviável.** A não ser que seja um fator físico ou algo do gênero e mesmo assim não pode interromper a navegação.

Por fim a recomendação a ser feita seria uma revisão dos endpoints:

1. **Cadastrar produto**;

2. **Editar produto**;

3. **Cadastrar Carrinho**.