Izadora - Plano de teste

Apresentação:

A Api Serverest é uma Api Rest que simula um e-commerce permitindo a realização operações como cadastro de usuários, login, listagem de produtos, gerenciamento de carrinhos e outras operações para fins educacionais.

Objetivo:

O objetivo deste plano de teste é garantir que as funcionalidades da Api Severest estejam funcionando corretamente e com qualidade de acordo com os critérios de aceite definidos nas User Stories e conforme o comportamento esperado na documentação.

Escopo:

Serão testados as principais funcionalidades da API:

- **Login** (`/login`):
  - Autenticação de usuários
- **Usuários** (`/usuarios`):
  - Cadastro de novos usuários
  - Listagem de usuários cadastrados
  - Consulta de usuário por ID
  - Atualização de dados de usuários
  - Exclusão de usuário
- **Produtos** (`/produtos`):
  - Cadastro de produtos
  - Listagem produtos cadastrados
  - Atualização e exclusão de produtos por ID
  - Buscar produto por ID
- **Carrinhos** (`/carrinhos`):
  - Criação de carrinhos
  - Consulta de carrinho por ID
  - Excluir carrinho quando concluir compra
  - Excluir carrinho quando cancelar compra
  - Lista carrinhos cadastrados

Análise

Durante a execução dos testes propostos neste plano, foram aplicadas técnicas específicas de teste de software para validar o comportamento da API Serverest, com foco nas funcionalidades críticas como produtos, usuários e carrinhos.

Técnicas Aplicadas

1. Teste Exploratório

Aplicado para explorar a API sem scripts rígidos, ajudou a identificar comportamentos inesperados e falhas não descritas na documentação oficial. Essa abordagem foi essencial para revelar:

- Falha na exclusão de produto com ID válido;
- Permissão de múltiplas exclusões de usuário sem retorno de erro;
- Restrição incorreta ao atualizar nome de produto já existente.

2. Partição de Equivalência

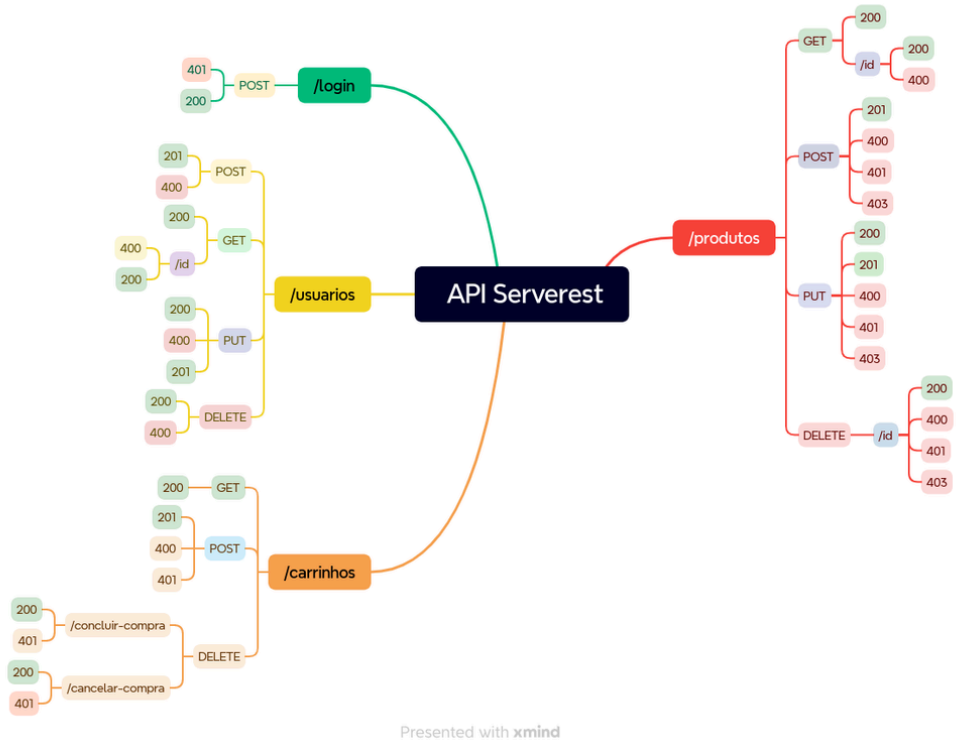
Auxiliou na criação de casos de teste com dados **válidos e inválidos** para identificar como a API trata diferentes tipos de entradas. Por exemplo:

- Envio de nomes de produtos válidos e duplicados;
- Exclusão de usuário existente e usuário já excluído.

3. Análise de Valor Limite

Utilizada para validar os extremos aceitáveis de dados, como quantidade de estoque e comprimento de campos. Embora não tenha resultado diretamente em bugs, ajudou a garantir estabilidade nas bordas da entrada dos dados.

Mapa mental da aplicação:



Cenários de teste

ID	Cenário de Teste	Passo	Resultado Esperado	Prioridade
01	Login com usuário válido	Enviar POST para `/login` com e-mail e senha corretos	Status 200 + Token de autenticação	Alta
02	Cadastro de novo usuário	Enviar POST para `/usuarios` com dados	Status 201 + Mensagem "Cadastro	Alta

		válidos	realizado com sucesso"	
03	Cadastro de novo produto	Enviar POST para `/produtos` autenticado como admin	Status 201 + Mensagem "Cadastro realizado com sucesso"	Alta
04	Criação de carrinho	Enviar POST para `/carrinhos` com produtos válidos e autenticado	Status 201 + Mensagem "Cadastro realizado com sucesso"	Alta
05	Excluir carrinho ao concluir compra	Enviar DELETE para `/carrinhos/concluir-compra` com carrinho finalizado	Status 200	Alta
06	Exclusão de usuário	Enviar DELETE para `/usuarios/{id}` com ID válido	Status 200 + Mensagem "Registro excluído com sucesso"	Alta
07	Consulta de usuário por ID	Enviar GET para `/usuarios/{id}`	Status 200 + Dados do usuário	Média
08	Atualização de dados de usuário	Enviar PUT para `/usuarios/{id}` com novos dados	Status 200 + Mensagem "Registro alterado com sucesso"	Média
09	Buscar produto por ID	Enviar GET para `/produtos/{id}`	Status 200 + Dados do produto	Média
10	Atualizar produto por ID	Enviar PUT para `/produtos/{id}` com novos dados	Status 200 + Mensagem "Registro alterado com sucesso"	Média
11	Excluir produto por ID	Enviar DELETE para `/produtos/{id}` com ID válido	Status 200 + Mensagem "Registro excluído com sucesso"	Média
12	Listagem de usuários cadastrados	Enviar GET para `/usuarios`	Status 200 + Lista de usuários	Baixa
13	Listagem de produtos cadastrados	Enviar GET para `/produtos`	Status 200 + Lista de produtos	Baixa
14	Consulta de carrinho por ID	Enviar GET para `/carrinhos/{id}`	Status 200 + Dados do carrinho	Baixa
15	Listar todos os carrinhos cadastrados	Enviar GET para `/carrinhos`	Status 200 + Lista de carrinhos	Baixa
16	Excluir carrinho ao cancelar compra	Enviar DELETE para `/carrinhos/cancelar-compra`	Status 200 + Mensagem "Registro excluído com sucesso"	Baixa

Matriz de risco

Nome do Risco	Prob.	Imp.	R=PxI	Estratégia	Como?	Responsável	Reavaliação
Login falha mesmo com dados válidos	3	3	9	Mitigar	Testes automatizados e de regressão	QA	Mensalmente
Cadastro de usuário indisponível	2	3	6	Mitigar	Testes com diferentes inputs, ambiente isolado	QA	Mensalmente
Produto não pode ser cadastrado	2	3	6	Mitigar	Verificar autenticação e regras de negócio	Backend	Mensalmente
Carrinho não é criado corretamente	3	3	9	Mitigar	Testar cenários com estoque, autenticação e preço	QA	Mensalmente
Dados do carrinho inconsistentes	2	2	4	Evitar	Validar com testes exploratórios	QA	Por evento
Falha na exclusão de carrinho	2	2	4	Evitar	Teste funcional e de integração	QA	Anualmente
Dados de teste insuficientes para validação	2	2	4	Mitigar	Criar massa de dados automatizada	QA	Semanalmente
API de usuário responde com atraso	2	3	6	Mitigar	Validar performance e limites de timeout	Backend	Mensalmente
Falha na atualização de produto	2	2	4	Evitar	Testes com diferentes cargas de dados	QA	Por evento
Testador ausente	2	2	4	Aceitar ativo	Ter backup ou rota de substituição	RH	Mensalmente

durante sprints							

Testes candidatos a automação

- Testes de login
- Validação de cadastro de usuários
- Adição de produtos
- Listagem de produtos

Motivo: são testes que se repetem com frequência, têm dados de entrada e saída presumíveis e tem respostas padroes