

2022

PLANO DE TESTES SERVEREST.DEV



»» **Vinícius Alexandre Pinto de Lima**
Compass.UOL

SUMÁRIO

- 01 SUMÁRIO**
- 02 INTRODUÇÃO**
- 03 OBJETIVOS**
- 04 ESCOPO**
- 05 MAPA MENTAL**
- 06 SUÍTES DE CASOS DE TESTE**
- 11 ESTRATÉGIA DE TESTE**
- 12 TESTES CANDIDATOS À AUTOMAÇÃO**
- 13 FERRAMENTAS**
- 14 EQUIPE DE TESTES**
- 15 RELATÓRIO DE BUGS**
- 19 CONCLUSÃO**

INTRODUÇÃO

Através deste documento serão descritas as metodologias e ferramentas utilizadas para a validação da API do e-commerce Serverest.

Este documento deve ser disponibilizado para todos os stakeholders envolvidos no projeto para o devido acompanhamento de todo o planejamento e passos executados durante o período de validação.

Os métodos de testes adotados visam cobrir o back-end da aplicação, com o foco maior na integração e também usando a automação para as rotas principais, com a expectativa de validar o comportamento da aplicação em diferentes cenários.



OBJETIVOS

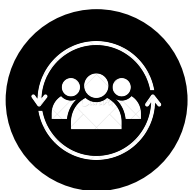
Por meio deste documento, é descrito o plano para a implementação de testes na API do E-commerce Serverest, com o foco principal na validação de seu funcionamento correto de acordo com a documentação estabelecida para a aplicação. Com isso, os principais objetivos são:

- Recomendar a metodologia para o desenvolvimento e implementação dos testes;
- Listar as ferramentas e equipe necessárias para a implementação dos testes;
- Validar o comportamento da API em diferentes cenários;
- Identificar informações essenciais sobre os testes a serem feitos e também sobre a aplicação qual está sendo testada

ESCOPO

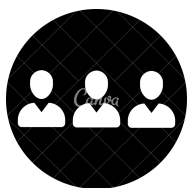
Este plano de testes descreve os testes de integração que serão realizados sobre a aplicação Serverest.

Levando em consideração que os testes unitários realizados já cobriram grande parte da aplicação, os seguintes itens dentro de cada rota estão no escopo dos testes:



Nº 01 – **Área Administrativa**

Edição das informações de um usuário, cadastro de produto, edição das informações de um produto.



Nº 02 – **Cadastro**

Rota utilizada para o cadastro de um novo usuário.
www.serverest.dev/usuarios (usando o método POST)



Nº 03 – **Login**

Realização de login, edição de e-mail e senha.



Nº 04 – **Itens fora do fluxo de Compra**

Cancelar um carrinho, excluir usuário, excluir um produto,

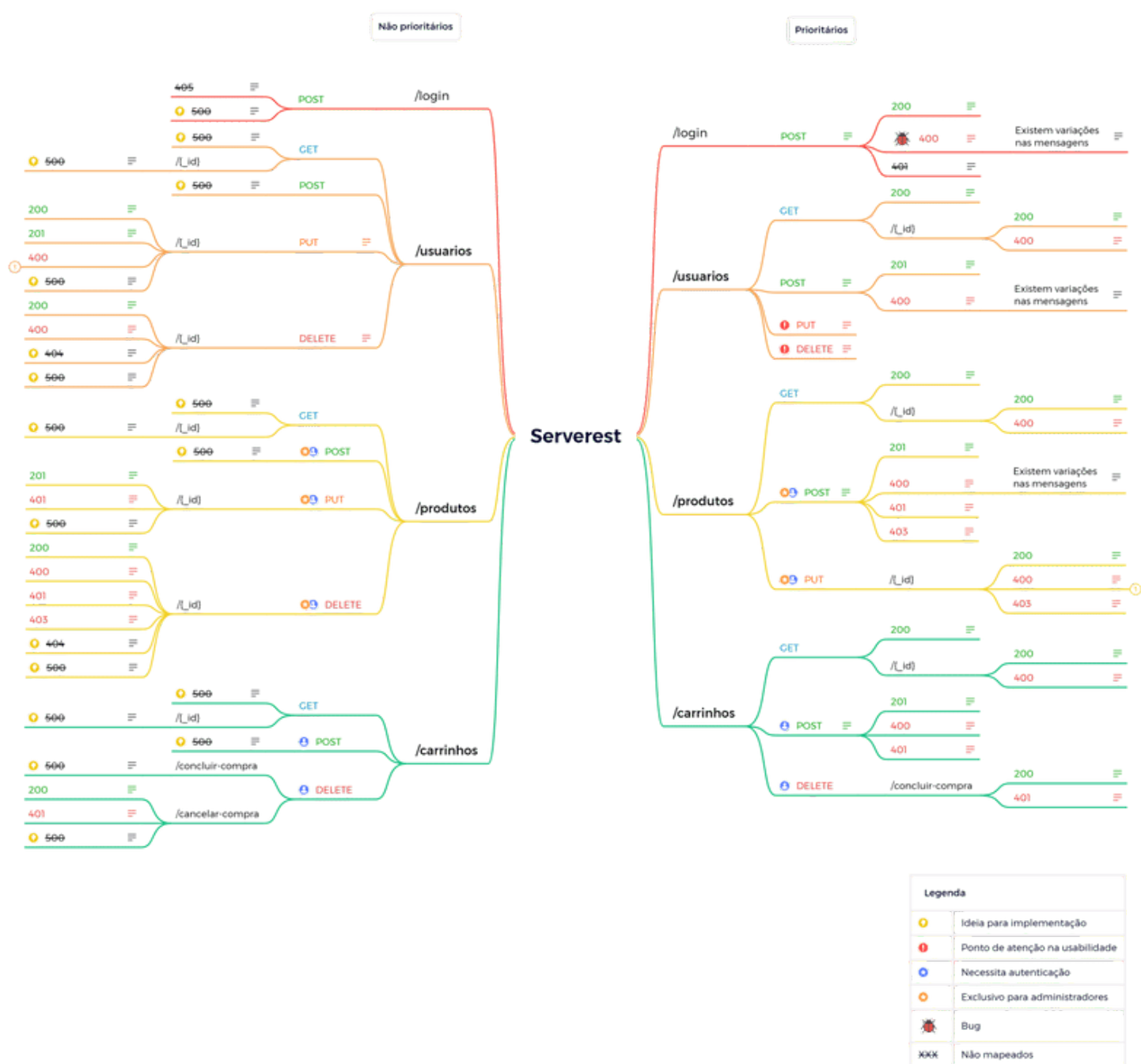


Nº 05 – **Fluxo de Compra**

Adicionar produtos ao carrinho, finalizar uma compra, lista de produtos, lista de carrinhos.

MAPA MENTAL

Através da documentação disponibilizada sobre o funcionamento da aplicação, foi feito o mapeamento de suas funções. Abaixo segue o mapa mental com toda a cobertura da API:



SUITE DE CASOS DE TESTE

Tendo em vista o tamanho e complexidade da aplicação, temos de dar cobertura as suas funcionalidades, validando se tudo ocorre conforme o previsto pela sua documentação. Abaixo seguem os casos de teste descritos para as rotas presentes na API Serverest:

/LOGIN	
LG01	Realizar o login com sucesso
LG02	Tentativa de login com senha incorreta
LG03	Tentativa de login com e-mail incorreto
LG04	Tentativa de login com e-mail em branco
LG05	Tentativa de login com senha em branco
LG06	Tentativa de login sem enviar um body na request a API

/USUARIOS	
US01	Buscar a lista de todos os usuários
US02	Buscar um usuário pelo ID com sucesso
US03	Buscar um usuário com um ID inexistente
US04	Adicionar um usuário com sucesso
US05	Adicionar um usuário com um e-mail já cadastrado
US06	Tentativa de adicionar um usuário com o e-mail em branco

SUITE DE CASOS DE TESTE

/USUARIOS	
US07	Tentativa de adicionar um usuário com a senha em branco
US08	Tentativa de adicionar um usuário com o nome em branco
US09	Tentativa de adicionar um usuário com a informação de administrador em branco
US10	Tentativa de adicionar um usuário com um e-mail inválido
US11	Tentativa de adicionar um usuário sem enviar um body na request a API
US12	Editar os dados de um usuário pelo ID com sucesso
US13	Tentativa de editar os dados de um usuário inexistente, resultando na criação de um novo usuário
US14	Tentativa de editar o usuário com um e-mail inválido
US15	Tentativa de editar um usuário com o nome em branco
US16	Tentativa de editar um usuário com o e-mail em branco
US17	Tentativa de editar um usuário com a senha em branco
US18	Tentativa de editar um usuário com a informação de administrador em branco
US19	Tentativa de editar um usuário sem encaminhar um body na request a API
US20	Tentativa de deletar um usuário pelo seu ID com sucesso
US21	Tentativa de deletar um usuário com um ID inexistente
US22	Tentativa de deletar um usuário com um carrinho de compras cadastrado

SUITE DE CASOS DE TESTE

/PRODUTOS	
PD01	Realizar a busca de todos os produtos cadastrados
PD02	Realizar a busca de um produto pelo ID com sucesso
PD03	Realizar a busca de um produto por um ID inexistente
PD04	Adicionar um novo produto com sucesso
PD05	Tentativa de adicionar um novo produto sem o token de autenticação
PD06	Tentativa de adicionar um produto com o nome em branco
PD07	Tentativa de adicionar um produto com a descrição em branco
PD08	Tentativa de usar números negativos na quantidade ao adicionar um produto
PD09	Tentativa de adicionar um produto com o valor negativo ou igual a zero
PD10	Tentativa de adicionar um produto com o nome de um produto já cadastrado
PD11	Tentativa de adicionar um produto com um usuário que não é administrador
PD12	Editar as informações de um produto com sucesso
PD13	Tentativa de editar os dados de um produto que não existe, realizando assim o cadastro de um novo produto com sucesso
PD14	Tentativa de editar os dados de um produto sem o Token de autenticação
PD15	Tentativa de editar os dados de um produto, enviando o nome de um produto que já existe
PD16	Tentativa de editar os dados de um produto colocando o nome em branco

SUITE DE CASOS DE TESTE

/PRODUTOS	
PD17	Tentativa de editar os dados de um produto colocando a descrição em branco
PD18	Tentativa de editar os dados de um produto com o preço do produto negativo ou igual a zero
PD19	Tentativa de editar os dados do produto com a quantidade menor que zero
PD20	Tentativa de editar os dados do produto repassando informações que não sejam números no campo de quantidade
PD21	Tentativa de editar os dados do produto repassando informações que não sejam números no campo de preço
PD22	Tentativa de editar os dados do produto sem enviar um body na request a API
PD23	Tentativa de realizar a edição nas informações de um produto com um usuário que não é administrador
PD24	Deletar um produto com sucesso
PD25	Tentativa de deletar um produto com um ID que não existe
PD26	Tentativa de deletar um produto que está dentro de um carrinho
PD27	Tentativa de deletar um produto sem o token de autenticação
PD28	Tentativa de deletar um produto com um usuário que não seja administrador

SUITE DE CASOS DE TESTE

/CARRINHOS	
CA01	Listar todos os carrinhos cadastrados com sucesso
CA02	Buscar um carrinho pelo seu ID com sucesso
CA03	Buscar um carrinho por um ID inexistente
CA04	Realizar o cadastro de um carrinho com sucesso
CA05	Realizar o cadastro de um produto duplicado no carrinho
CA06	Tentativa de cadastro de um carrinho sem realizar o login
CA07	Tentativa de cadastro de um carrinho com o campo "id do produto" em branco
CA08	Tentativa de cadastro de carrinho sem especificar a quantidade do produto que está sendo adicionado ao carrinho
CA09	Tentativa de cadastro de carrinho com uma quantidade do produto maior que a quantidade disponível no momento
CA10	Finalizar uma compra com sucesso(usando um usuário que possui o carrinho criado)
CA11	Tentativa de finalizar a compra com um usuário sem carrinho cadastrado
CA12	Tentativa de finalizar a compra com usuário sem token de autenticação
CA13	Cancelar um carrinho com sucesso (através de um usuário com um carrinho cadastrado)
CA14	Tentativa de cancelar um carrinho em um usuário sem carrinho cadastrado
CA15	Tentativa de cancelar um carrinho em um usuário sem o token de autenticação
CA16	Tentativa de adicionar carrinho com produto duplicado
CA17	Tentativa de adicionar carrinho para usuário com carrinho já cadastrado
CA18	Tentativa de adicionar carrinho com produto de ID inexistente

ESTRATÉGIA DE TESTE

TESTE DE INTEGRAÇÃO

A estratégia de testes utilizada nesse plano é para os testes de integração, conforme a funcionalidade da aplicação, para isso será utilizada a ferramenta Postman, onde serão realizadas validações sobre a API para confirmar as respostas das rotas estabelecidas em sua documentação pelo Swagger. Posteriormente algumas rotas já pré-definidas terão sua automação desenvolvida usando a ferramenta Cypress.



TESTES CANDIDATOS À AUTOMAÇÃO

Os seguintes testes foram selecionados para a automatização, tendo em vista que são essenciais para o objetivo principal do e-commerce, sendo assim, através dessa automatização sempre será possível validar se o funcionamento das funcionalidades está de acordo com o esperado.

- **LG01** - Realizar o login com sucesso
- **LG02** - Tentativa de login com senha incorreta
- **US01** - Buscar a lista de todos os usuários
- **US02** - Buscar um usuário pelo ID com sucesso
- **US03** - Buscar um usuário com um ID inexistente
- **US04** - Adicionar um usuário com sucesso
- **US05** - Adicionar um usuário com um e-mail já cadastrado
- **PD01** - Realizar a busca de todos os produtos cadastrados
- **PD02** - Realizar a busca de um produto pelo ID com sucesso
- **PD03** - Realizar a busca de um produto por um ID inexistente
- **PD04** - Adicionar um novo produto com sucesso
- **PD05** - Tentativa de adicionar um novo produto sem o token de autenticação
- **PD10** - Tentativa de adicionar um produto com o nome de um produto já cadastrado
- **PD11** - Tentativa de adicionar um produto com um usuário que não é administrador
- **PD12** - Editar as informações de um produto com sucesso
- **PD14** - Tentativa de editar os dados de um produto sem o Token de autenticação
- **PD15** - Tentativa de editar os dados de um produto, enviando o nome de um produto que já existe
- **CA01** - Listar todos os carrinhos cadastrados com sucesso
- **CA02** - Buscar um carrinho pelo seu ID com sucesso
- **CA03** - Buscar um carrinho por um ID inexistente
- **CA04** - Realizar o cadastro de um carrinho com sucesso
- **CA05** - Realizar o cadastro de um produto duplicado no carrinho
- **CA06** - Tentativa de cadastro de um carrinho sem realizar o login
- **CA10** - Finalizar uma compra com sucesso(usando um usuário que possui o carrinho criado)
- **CA12** - Tentativa de finalizar a compra com usuário sem token de autenticação
- **CA16** - Tentativa de adicionar carrinho com produto duplicado

FERRAMENTAS

Para o desenvolvimento deste plano de testes e também para a implementação dos testes descritos neste documento foram utilizadas as seguintes ferramentas:

- **Visual Studio Code:** IDE usada para o desenvolvimento da automação;
- **Pacote Office:** Usado na elaboração do plano de testes;
- **Canva:** Utilizado na formatação do plano de testes;
- **Microsoft Teams:** Usado para comunicações e reuniões;
- **Cypress:** Framework para a automação;
- **JavaScript:** Linguagem de programação usado na automação dos testes;
- **Postman:** Usado na interação com as rotas da API;
- **Git e GitHub:** Usados no versionamento do código;
- **Faker-js:** Biblioteca utilizada para gerar dados aleatórios;
- **Ajv:** Usado na validação do contrato da API;
- **Mochawesome:** Ferramenta utilizada para gerar reports sobre os testes automatizados

EQUIPE DE TESTES



Para a implementação dos testes listados neste plano teremos presente a seguinte equipe:

- **Lider de QA:** Maria José
- **Analista de Testes:** José Maria
- **Testador:** José Ramalho
- **Dev. de automação:** Fátima Bernardes
- **Arquiteto de automação:** Pedro Ballas

RELATÓRIO DE BUGS

A seguir estão listados alguns bugs que foram encontrados através dos testes implementados na aplicação e suas evidências

1. Realizar login com dados inválidos

Classificação: Baixa

Resultado obtido

Ao tentar fazer o login com e-mail ou senha inválidos, a aplicação está retornando o status code 401 que não foi mapeado pela documentação.

Resultado esperado

Realizar tentativa de login com dados inválidos e receber uma response com status code 400 com a seguinte mensagem: "Email e/ou senha inválidos"

Passos para a reprodução

1. Acessar a ferramenta por onde a requisição à API será realizada
2. Preencher as informações do body da requisição que será feita
 - a. O body da requisição precisa ser preenchido usando um e-mail ou senha incorretos
3. Realizar a requisição com o método POST para a rota serverest.dev/login

Evidências:

Body enviado na requisição:

```
{  
  "email": "f@qa.com",  
  "password": "teste"  
}
```

Resposta obtida:

```
401  
Unauthenticated Error;  
  
Response body  
{  
  "message": "Email e/ou senha inválidos"  
}
```


RELATÓRIO DE BUGS

2. Erro no contrato ao realizar requisição à rota de produtos

Classificação: Baixa

Resultado obtido

Ao realizar requisição para rota de produtos da API estamos recebendo uma response com as propriedades quantidade e produtos.

Resultado esperado

Ao realizar a request pelo método get devemos receber uma resposta com as propriedades quantidade e usuários.

Passos para a reprodução

1. Acessar a ferramenta por onde a requisição à API será realizada
2. Realizar uma requisição à rota serverest.dev/produtos pelo método GET
3. Confirmar a resposta obtida

Evidências:

Conforme documentação:

Resultado obtido:

Code	Description
200	Lista de produtos
	Example Value Model
	<pre>{ "quantidade": 3, "usuarios": [{ "nome": "Logitech MX Vertical", "preco": 470, "descricao": "Mouse", "quantidade": 381, "_id": "8ee3h51z3k8k51zA" }, { "nome": "Samsung 60 polegadas", "preco": 5340, "descricao": "TV", "quantidade": 49377, "_id": "K61eHdftCe03j883" }] }</pre>

Code	Details
200	Response body
	<pre>{ "quantidade": 164, "produtos": [{ "nome": "Rustic Plastic Pants", "preco": 58122, "descricao": "Ergonomic executive", "quantidade": 82709, "_id": "15HuHqFy0DQt4yq" }, { "nome": "Modern Plastic Tuna", "preco": 221, "descricao": "Ergonomic executive", "quantidade": 81859, "_id": "15Nnb4SesH40t3Y2" }, { "nome": "Refined Soft Shoes", "preco": 962, "descricao": "Andy shoes are design", "quantidade": 2654, "_id": "1X81w6CI2d04F2G" }] }</pre>

RELATÓRIO DE BUGS

3. Variações nas mensagens na rota de login (MELHORIA)

Classificação: Baixa

Resultado obtido

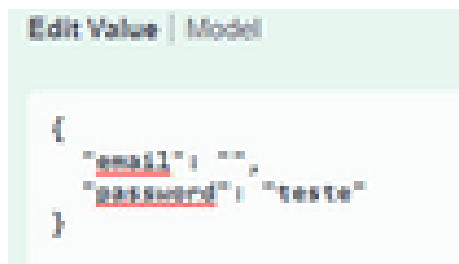
Ao validar diferentes cenários em tentativas de login, foram obtidas diferentes respostas que não estão previstas na documentação.

Passos para a reprodução

1. Acessar a ferramenta por onde a requisição à API será realizada;
2. Realizar uma requisição à rota `serverest.dev/login` pelo método POST;
3. Preencher o body para a requisição ser realizada
 - a. Requisição com Email em branco
 - b. Requisição com Senha em branco
 - c. Requisição com email inválido
 - d. Requisição sem email ou senha enviados no body da request

Evidências:

Body enviado:



The screenshot shows a REST client interface with a tab labeled 'Edit Value | Model'. The body of the request is a JSON object:

```
{  "email": "",  "password": "teste"}
```

Resposta obtida:

Code	Details
400	<p>Error:</p> <p>Response body</p> <pre>{ "email": "email não pode ficar em branco"}</pre>

RELATÓRIO DE BUGS

4. Edição e exclusão de cadastros pode ser realizada por qualquer usuário (MELHORIA)

Classificação: Alta

Descrição:

Atualmente, na rota utilizada para realizar alterações no cadastro do usuário e também deletar o cadastro, as requisições podem ser feitas por qualquer usuário que conseguir o ID da conta que deseja ser editada.

Uma sugestão para a melhoria da aplicação e aumentar a segurança de nossos usuários seria alterar essa solicitação, deixando a permissão de realizá-las apenas para o próprio usuário da conta ou então para algum usuário administrador.

CONCLUSÃO

Através deste documento ficam listadas as atividades e processos mínimos para o desenvolvimento e implementação das atividades de teste.

Conforme o avanço das atividades ocorre, se necessário, as atividades aqui transcritas podem sofrer alterações de acordo com a urgência da adaptação.

Como única regra inalterável, não podemos excluir nenhuma determinação aqui presente, apenas estender os prazos de conclusão.