

Plano de Testes

Por: Daniela Siana Pabis | Squad 01: Unit to Unity | Data: 06/02/2026

IDENTIFICAÇÃO DO PROJETO

Nome do Projeto: Planejamento de Testes Funcionais - ServeRest

API Testada: ServeRest Compass UOL (<https://compassuol.serverest.dev/>)

Versão do Plano: 1.0

Data de Criação: 02/02/2026

Data da Última Atualização: 06/02/2026

Responsável pela Elaboração: Daniela Siana Pabis

Responsável pela Aprovação: Avaliadores do programa de bolsas

Tempo Estimado de Execução: Em torno de 16 horas (Considerando o tempo integral do estágio durante a semana)

ESCOPO

Serão realizados testes funcionais e testes exploratórios baseados na validação de entrada (Input Validation) e no fluxo de estado e regras de negócio (State Transition and Business Rules Testing) da API ServeRest.

Os testes exploratórios têm como objetivo identificar falhas relacionadas ao tratamento inadequado de dados inválidos, comportamentos inesperados entre requisições encadeadas e inconsistências na aplicação das regras de negócio, especialmente em cenários que envolvem dependência entre recursos.

O escopo contempla os principais recursos da API:

Usuários: validação de campos obrigatórios, unicidade de e-mail, criação, atualização e exclusão de registros, além do impacto da exclusão de usuários vinculados a carrinhos.

Autenticação: validação de credenciais, geração e uso de token e impacto do estado de autenticação nos fluxos dependentes.

Produtos: operações CRUD com foco na validação de payloads, permissões de acesso e impacto do estado do produto nos carrinhos.

Carrinhos: criação, finalização e cancelamento, avaliando dependências entre usuário, produtos e estoque, bem como o comportamento da API em fluxos inválidos.

Não fazem parte do escopo:

- Testes automatizados;
- Testes de performance ou carga;
- Testes de segurança avançados (pentest).

Ambiente de Testes:

- API Base URL: ServeRest;
- Ferramenta: Postman;
- Sistema Operacional: Windows 11.

OBJETIVOS

Geral

Avaliar a qualidade funcional da API ServeRest por meio de testes funcionais e testes exploratórios baseados na validação de entrada (Input Validation) e no fluxo de estado e regras de negócio (State Transition and Business Rules Testing), assegurando que os endpoints tratem corretamente dados válidos e inválidos e mantenham a integridade do sistema ao longo das operações.

Específicos

- Verificar se a API valida corretamente os dados de entrada, rejeitando campos ausentes, inválidos ou fora dos limites esperados;
- Avaliar o comportamento da API diante de fluxos sequenciais, como criação, atualização e exclusão de recursos dependentes;
- Identificar falhas na aplicação das regras de negócio, como duplicidade de registros, restrições de exclusão e controle de estoque;
- Avaliar a consistência das respostas, incluindo status HTTP, mensagens de erro e estrutura dos payloads;
- Explorar cenários fora do fluxo esperado, analisando estados inválidos do sistema.

ESTRATÉGIA DE TESTES

Testes Funcionais

Os testes funcionais serão executados com base em cenários previamente definidos, cobrindo os fluxos principais da API, como cadastro de usuários, autenticação, gerenciamento de produtos e carrinhos.

Testes Exploratórios

Técnicas utilizadas: Input Validation e State Transition and Business Rules Testing.

A estratégia de testes exploratórios adotada consiste na execução de testes exploratórios baseados na validação de entrada (Input Validation) e no fluxo de estado e regras de negócio (State Transition and Business Rules Testing).

A exploração será conduzida a partir de:

- Variações nos dados de entrada, incluindo ausência de campos obrigatórios, tipos inválidos e valores fora dos limites esperados;
- Execução de fluxos completos e incompletos, observando o impacto de cada operação no estado do sistema;
- Essa abordagem permite identificar falhas funcionais, inconsistências de negócio e comportamentos inesperados que não seriam facilmente detectados apenas por testes funcionais isolados.

MAPEAMENTO DE ENDPOINTS

Recurso	Descrição	Endpoint	Método	Regras	Respostas
Login	Autentica usuário e retorna token	/login	POST	Autenticação para acesso a rotas privadas	200, 401
Usuários	Cadastra um novo usuário	/usuarios	POST	Não é permitido email duplicado	201, 400
Usuários	Lista usuários cadastrados	/usuarios	GET	Permite filtragem de resultados	200
Usuários	Buscar usuário por ID	/usuarios/{_id}	GET	Localiza registro específico via identificador	200, 400
Usuários	Editar ou criar usuário	/usuarios/{_id}	PUT	Operação de UPSERT; valida e-mail único	200, 201, 400
Usuários	Exclui usuário	/usuarios/{_id}	DELETE	Proibida a exclusão de usuários com carrinho ativo	200, 400
Produtos	Cadastra novo produto	/produtos	POST	Restrito a administradores; nome deve ser único	201, 400, 401, 403
Produtos	Lista produtos cadastrados	/produtos	GET	Permite busca por atributos do produto	200, 400
Produtos	Retorna dados de um produto	/produtos/{_id}	GET	Localiza produto específico via identificador	200, 400
Produtos	Edita ou cria produto	/produtos/{_id}	PUT	Restrito a administradores; realiza UPSERT	200, 201, 400, 401, 403
Produtos	Exclui produto	/produtos/{_id}	DELETE	Restrito a administradores; proibido se houver carrinho vinculado	200, 400, 401, 403
Carrinhos	Cria carrinho	/carrinhos	POST	Apenas 1 carrinho por usuário; reduz estoque automaticamente	201, 400, 401

Carrinhos	Lista carrinhos cadastrados	/carrinhos	GET	Exibe todos os carrinhos ativos na base	200
Carrinhos	Buscar carrinho por ID	/carrinhos/{_id}	GET	Localiza carrinho específico via identificador	200, 400
Carrinhos	Finalizar Compra	/carrinho s/ concluir-compra	DELETE	Exclui o carrinho vinculado ao token do usuário	200, 401
Carrinhos	Cancelar Carrinho	/carrinhos / cancelar-compra	DELETE	Exclui o carrinho e devolve os itens ao estoque	200, 401

RISCOS E PRIORIDADE

Risco	Descrição	Impacto	Mitigação
Inconsistência no mecanismo de autenticação	O processo de emissão e validação de tokens pode aceitar credenciais inválidas ou rejeitar acessos legítimos	Alto	Executar testes de autenticação com tokens válidos, inválidos e expirados
Violação de controle de acesso	Regras de autorização podem não ser aplicadas corretamente aos endpoints restritos	Alto	Validar acesso a rotas protegidas com diferentes perfis de usuário
Falhas no saneamento de dados de entrada	Parâmetros podem aceitar tipos, formatos ou valores fora dos limites definidos	Alto	Aplicar Input Validation, incluindo testes de valores limite e tipos inválidos
Divergência de contrato da API	Estrutura de requests ou responses pode não seguir o contrato especificado	Alto	Validar schemas JSON conforme documentação da API
Estados inválidos entre operações	Sequências de operações podem gerar estados inconsistentes entre recursos relacionados	Alto	Executar State Transition and Business Rules Testing
Padronização inadequada de respostas de erro	Códigos HTTP ou payloads de erro podem ser inconsistentes entre endpoints	Médio	Validar status HTTP, mensagens e estrutura das respostas de erro

CENÁRIOS DE TESTES FUNCIONAIS

ID	Cenário	Endpoint	Tipo	Prior.	Status
CT01	Cadastrar usuário com dados válidos	POST /usuarios	Positivo	Alta	Planejado
CT02	Cadastrar usuário com e-mail já existente	POST /usuarios	Negativo	Alta	Planejado
CT03	Cadastrar usuário sem campos obrigatórios	POST /usuarios	Negativo	Alta	Planejado
CT04	Validar schema de request e response do cadastro de usuário	POST /usuarios	Contrato	Média	Planejado
CT05	Realizar login com credenciais válidas	POST /login	Positivo	Alta	Planejado
CT06	Realizar login com credenciais inválidas	POST /login	Negativo	Alta	Planejado
CT07	Validar schema da resposta de login (token)	POST /login	Contrato	Alta	Planejado
CT08	Criar produto com usuário administrador	POST /produtos	Positivo	Alta	Planejado
CT09	Criar produto com usuário não administrador	POST /produtos	Negativo	Alta	Planejado
CT10	Criar produto com dados inválidos	POST /produtos	Negativo	Média	Planejado
CT11	Validar schema de request e response do produto	POST /produtos	Contrato	Média	Planejado
CT12	Listar produtos cadastrados	GET /produtos	Positivo	Média	Planejado
CT13	Buscar produto por ID válido	GET /produtos/{id}	Positivo	Média	Planejado
CT14	Atualizar produto com usuário administrador	PUT /produtos/{id}	Positivo	Média	Planejado
CT15	Excluir produto sem vínculo com carrinho	DELETE /produtos/{id}	Positivo	Média	Planejado
CT16	Criar carrinho com produtos válidos	POST /carrinhos	Positivo	Alta	Planejado
CT17	Criar carrinho sem autenticação	POST /carrinhos	Negativo	Alta	Planejado
CT18	Criar carrinho com produto inexistente	POST /carrinhos	Negativo	Alta	Planejado
CT19	Validar schema de request e response da criação de carrinho	POST /carrinhos	Contrato	Média	Planejado
CT20	Listar carrinhos cadastrados	GET /carrinhos	Positivo	Média	Planejado
CT21	Buscar carrinho por ID válido	GET /carrinhos/{id}	Positivo	Média	Planejado
CT22	Finalizar carrinho com sucesso	DELETE /carrinhos/concluir-compra	Positivo	Alta	Planejado
CT23	Finalizar carrinho inexistente	DELETE /carrinhos/concluir-compra	Negativo	Média	Planejado
CT24	Validar schema da resposta de finalização de carrinho	DELETE /carrinhos/concluir-compra	Contrato	Média	Planejado
CT25	Cancelar carrinho com sucesso	DELETE /carrinhos/cancelar-compra	Positivo	Média	Planejado

CT26	Listar usuários cadastrados	GET /usuarios	Positivo	Média	Planejado
CT27	Buscar usuário por ID válido	GET /usuarios/{id}	Positivo	Média	Planejado

CENÁRIOS DE TESTES EXPLORATÓRIOS

Exploratory Testing based on Input Validation

ID	Cenário Exploratório	Endpoints Alvo	Foco da Exploração
ET01	Submissão de payloads com tipos inválidos, campos ausentes e campos extras	POST /usuarios, POST /produtos	Validação de tipos, obrigatoriedade e saneamento de dados
ET02	Envio de valores extremos e limites de tamanho nos campos de entrada	POST /usuarios, POST /produtos	Boundary values e restrições de tamanho
ET03	Uso de token ausente, malformado ou expirado em requisições autenticadas	POST /produtos, POST /carrinhos	Validação do token e consistência de respostas de erro

Exploratory Testing based on State Transition and Business Rules

ID	Cenário Exploratório	Endpoints Alvo	Foco da Exploração
ET04	Execução de operações fora da ordem esperada do fluxo	POST /carrinhos, DELETE /carrinhos/concluir-compra	Transições de estado inválidas
ET05	Repetição de operações que deveriam ser bloqueadas ou idempotentes	POST /carrinhos, DELETE /carrinhos/concluir-compra	Regras de negócio e proteção contra duplicidade
ET06	Uso do mesmo token em operações sequenciais com perfis diferentes	POST /produtos, PUT /produtos/{id}	Autorização baseada em perfil e regras de acesso

CRITÉRIOS DE ENTRADA, SAÍDA E DEFINIÇÃO DE PRONTO

Condições para Início dos Testes

Os testes somente serão iniciados após o atendimento dos seguintes pré-requisitos:

- Disponibilidade da API em ambiente acessível para testes;
- Definição clara dos endpoints que farão parte do ciclo de testes;
- Existência de dados mínimos para execução, incluindo exemplos válidos e inválidos;

- Compreensão das regras de negócio associadas a cada recurso;
- Definição prévia do que caracteriza sucesso ou falha para cada cenário avaliado.

Condições para Encerramento dos Testes

O ciclo de testes será encerrado quando:

- Os cenários funcionais planejados tiverem sido executados;
- As sessões de testes exploratórios tiverem sido concluídas;
- Os fluxos considerados críticos estiverem avaliados;
- As anomalias encontradas estiverem registradas;
- Não existirem cenários pendentes relacionados aos fluxos prioritários.

Definição de Pronto

A API será considerada pronta do ponto de vista de testes quando:

- Os principais fluxos de uso estiverem validados;
- O comportamento da API estiver consistente com as regras de negócio conhecidas;
- As evidências de teste estiverem documentadas;
- As falhas relevantes identificadas estiverem registradas para acompanhamento.

EVIDÊNCIAS (Exemplos de Requisições)

Exemplo 1 – Login com credenciais válidas (Cenário Positivo – Fluxo principal)

Endpoint: POST /login

Request Body:

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

Response Esperado – 200 OK:

```
{  
  "message": "Login realizado com sucesso",  
  "authorization": "Bearer <token>"  
}
```

Validação: Retorno do status code 200 e presença do campo authorization, conforme contrato da API.

**Exemplo 2 – Cadastro de usuário com dados válidos
(Cenário Positivo – Fluxo principal)**

Endpoint: POST /usuarios

Request Body:

```
{  
  "nome": "Fulano da Silva",  
  "email": "beltrano@qa.com.br",  
  "password": "teste",  
  "administrador": "true"  
}
```

Response Esperado – 201 Created:

```
{  
  "message": "Cadastro realizado com sucesso",  
  "_id": "<id>"  
}
```

Validação: Status code 201 e retorno do identificador do usuário criado.

**Exemplo 3 – Cadastro de produto com usuário administrador
(Cenário Positivo – Regra de negócio)**

Endpoint: POST /produtos

Header: Authorization: Bearer <tokenAdmin>

Request Body:

```
{  
  "nome": "Produto Exemplo",  
  "preco": 3500,  
  "descricao": "Produto para testes",  
  "quantidade": 10  
}
```

Response Esperado – 201 Created:

```
{  
  "message": "Cadastro realizado com sucesso",  
  "_id": "<idProduto>"
```

```
}
```

Validação: Status code 201, token de administrador aceito e estrutura da resposta conforme contrato

**Exemplo 4 – Tentativa de criação de carrinho quando já existe um carrinho ativo
(Cenário Negativo – Regra crítica)**

Endpoint: POST /carrinhos

Header: Authorization: Bearer <tokenUsuario>

Request Body:

```
{
  "produtos": [
    {
      "idProduto": "<idProduto>",
      "quantidade": 2
    }
  ]
}
```

Response Esperado – 400 Bad Request:

```
{
  "message": "Não é permitido ter mais de 1 carrinho"
}
```

Validação: Status code 400 e validação da regra que impede múltiplos carrinhos ativos por usuário.

**Exemplo 5 – Validação de contrato na listagem de usuários
(Teste de Contrato)**

Endpoint: GET /usuarios

Response Esperado – 200 OK (modelo):

```
{
  "quantidade": 1,
  "usuarios": [
    {
      "_id": "string",
      "nome": "string",
      "email": "string"
    }
  ]
}
```

```
        "nome": "string",
        "email": "string",
        "password": "string",
        "administrador": "string"
    }
]
}
```

Validação: Verificação da presença dos campos obrigatórios, tipos de dados corretos e estrutura do JSON conforme especificação da API.

Exemplo 6 – Tentativa de cadastro de produto por usuário não administrador (Cenário Negativo – Autorização)

Endpoint: POST /produtos

Header: Authorization: Bearer <tokenUsuarioComum>

Request Body:

```
{
    "nome": "Produto Teste",
    "preco": 500,
    "descricao": "Teste de permissão",
    "quantidade": 3
}
```

Response Esperado – 403 Forbidden:

```
{
    "message": "Rota exclusiva para administradores"
}
```

Validação: Status code 403 e bloqueio correto da operação conforme regra de autorização.