

# Plano de Testes: Beer Brewery Stock Manager API

Versão: 1.0

Data: 23 de Novembro de 2025

Responsável: Ana Santana (QA Lead)

## 1. Visão Geral e Escopo

O objetivo deste Plano de Testes é validar a API **Beer Brewery Stock Manager** contra os requisitos funcionais e as exigências não-funcionais críticas (Segurança, Performance, Usabilidade/DX), garantindo a precisão do controle de estoque por Lote e a aplicação da regra **FEFO (First-Expired, First-Out)**.

## 2. Estratégia de Teste

A estratégia principal é uma abordagem em camadas que garante a qualidade desde o código unitário até o comportamento da API em produção.

Camada de Teste	Foco Principal	Metodologia	Ferramentas
<b>Testes de Unidade</b>	Lógica de Negócio (Services) e Regras Críticas (FEFO, Validação, Alertas).	Isolamento total; Uso de Mocks para simular dependências.	JUnit 5, Mockito
<b>Testes de Integração</b>	Persistência de Dados, Transações e Comunicação entre Service e Repositório (JPA).	Uso de um banco de dados real (PostgreSQL) ou Test Containers para simular o ambiente de execução.	Spring Boot Test

<b>Testes Funcionais/Aceitação</b>	Verificação dos Endpoints REST contra os Requisitos (RFs) e Casos de Uso (UCs).	Execução de Casos de Teste (via REST Assured/Postman) em ambiente de <i>staging</i> .	REST Assured, Postman
------------------------------------	---	---	-----------------------

### 3. Testes Não-Funcionais Críticos

#### 3.1. Segurança (Autenticação e Autorização)

Item	Objetivo	Critério de Sucesso
<b>Autorização (ACL)</b>	Garantir que o acesso aos endpoints seja restrito de acordo com o <b>Role</b> do usuário.	Tentativas de acesso a endpoints restritos devem retornar <b>HTTP 403 Forbidden</b> .
<b>Autenticação (OAuth2/JWT)</b>	Garantir que a API só processe requisições com <i>tokens</i> de autenticação válidos.	Tentativas sem token ou com token inválido devem retornar <b>HTTP 401 Unauthorized</b> .
<b>Análise Estática (SAST)</b>	Identificar vulnerabilidades de código (Ex: Injeção de SQL ou Log Forging) em <i>build time</i> .	Zero vulnerabilidades Críticas ou Altas identificadas antes da <i>feature merge</i> .

#### 3.2. Performance e Estabilidade

Item	Objetivo	Critério de Sucesso (Exigência)
<b>Teste de Carga (Load Test)</b>	Determinar o volume máximo de requisições que a API suporta.	Suportar <b>200 usuários virtuais simultâneos</b> por 5 minutos nos endpoints de Movimentação/Vendas ( <b>POST /pedidos</b> ) com tempo de resposta estável.

<b>Latência (Tempo de Resposta)</b>	Medir a velocidade de resposta em requisições críticas de consulta.	O tempo de resposta (p95) para o endpoint de Consulta de Estoque ( <a href="#">GET /estoque</a> ) deve ser <b>inferior a 250ms</b> .
<b>Teste de Estresse</b>	Identificar o ponto de falha do sistema.	A API deve falhar de forma controlada (sem corromper dados) e se recuperar em menos de 60 segundos após a sobrecarga.

### 3.3. Usabilidade e Consistência (Developer Experience - DX)

Item	Objetivo	Critério de Sucesso
<b>Consistência REST</b>	Garantir que o design dos endpoints siga o padrão RESTful (Ex: plural e verbos HTTP corretos).	<b>POST /cervejas</b> para criar; <b>GET /cervejas/{id}</b> para consultar.
<b>Tratamento de Exceções</b>	Fornecer feedback claro e consistente para o cliente da API em caso de falha.	Erros de validação devem retornar <b>400 Bad Request</b> e erros de recurso não encontrado devem retornar <b>404 Not Found</b> , ambos com corpo JSON padronizado.
<b>Documentação (Swagger)</b>	Garantir que todos os endpoints e modelos de dados estejam detalhados no OpenAPI.	A documentação online deve ser gerada automaticamente e estar 100% sincronizada com o código.

---

## 4. Cobertura e Critérios de Saída

Categoria	Critério de Saída (Go/No-Go)

<b>Cobertura de Código (JaCoCo)</b>	Mínimo de <b>85%</b> nas Classes de Serviço (Service Layer) e Entidades Críticas.
<b>Bugs</b>	Zero (0) bugs Críticos ou Altos pendentes na branch <b>develop</b> .
<b>Performance</b>	Todos os requisitos de Latência e Carga devem ser atendidos nos ambientes de Teste e Staging.
<b>Testes Funcionais</b>	100% dos Casos de Teste de Prioridade CRÍTICA e ALTA Aprovados.

## 5. Casos de Teste Essenciais (Exemplos Críticos)

ID	Módulo	Objetivo do Teste	Passos de Execução (API)	Resultado Esperado
CT_FE01	FEFO	<b>Teste FEFO:</b> Verificar se a baixa prioriza o Lote com a validade mais próxima.	1. Cadastrar 3 Lotes de uma cerveja (Validades T1, T2, T3). 2. Simular venda que só zere T1.	A API deve registrar a saída <b>integralmente no Lote T1</b> e retornar <b>200 OK</b> .
CT_EN01	Entrada	Validação: Bloquear <b>POST</b> com data de validade vencida.	Enviar <b>POST /movimentacoes/entrada</b> com <b>dataValidade = "Ontem"</b> .	A API deve retornar <b>400 Bad Request</b> devido à falha na RN de validação.

<b>CT_SE02</b>	Segurança	Bloquear acesso por perfil incorreto (ACL).	Obter token do perfil "Vendedor". 2. Tentar usar este token para acessar o endpoint <b>POST /usuarios</b> .	A API deve retornar <b>403 Forbidden</b> .
<b>CT_EX01</b>	Exceção	Verificar tratamento de ID inexistente.	Tentar <b>GET /cervejas/{id}</b> onde <b>{id}</b> é um valor que não existe no DB.	A API deve retornar <b>404 Not Found</b> com mensagem clara de "Recurso não encontrado".

## Estratégia de Teste por Camada

Camada de Teste	Foco Principal	Metodologia	Ferramentas
<b>Unidade (Unit)</b>	Lógica de Negócio (Services) e Regras Críticas (FEFO, Alertas).	Simulação de dependências (Mocks) para isolamento total do componente.	JUnit 5, Mockito
<b>Integração (Integration)</b>	Fluxo completo do Controller ao Repositório e DB.	Uso de <i>Test Containers</i> ou banco de dados em memória (H2) para <i>fast feedback</i> .	Spring Boot Test
<b>Aceitação/Funcional</b>	Verificação dos Endpoints REST contra os Casos de Uso (UCs).	Uso de scripts de teste para simular o comportamento do cliente final da API.	REST Assured, Postman

<b>Regressão (Contínua)</b>	Garantir que alterações não quebrem funcionalidades Core.	Automação do Conjunto Crítico de Testes (CTs de Movimentação/Segurança) a cada PR.	GitHub Actions, JaCoCo
-----------------------------	---	--	------------------------

### 3. Testes Funcionais e Regras de Negócio (Amostra Crítica)

ID	Requisito (RF)	Objetivo do Teste	Passos de Execução (API)	Resultado Esperado
CT_FE01	RF008 (Baixa em Vendas)	<b>Validação FEFO:</b> Priorizar o Lote com validade mais próxima.	1. Cadastrar 3 Lotes (T1, T2, T3) com validades T1 < T2 < T3. 2. Simular venda (baixa) zerando apenas T1.	A API deve registrar a saída integralmente no <b>Lote T1</b> (Validade mais próxima).
CT_EN01	RF006 (Entrada)	Bloquear <b>POST</b> com data de validade vencida.	Enviar <b>POST</b> <code>/movimentacoes/entrada</code> com <b>dataValidade</b> = Data Passada.	A API deve retornar <b>400 Bad Request</b> devido à falha na Regra de Negócio.
CT_SE02	RF026 (ACL)	Bloquear acesso por perfil incorreto.	Obter token de um usuário "Vendedor". 2. Tentar acessar <b>POST /usuarios</b> (restrito ao Admin).	A API deve retornar <b>403 Forbidden</b> .

### 4. Testes Não-Funcionais Detalhados

#### 4.1. Segurança e Acessibilidade

Item	Foco	Estratégia de Teste	Critério de Sucesso
Autorização (ACL)	Controle de Acesso por Perfil (Role).	Testar todos os endpoints restritos com tokens de perfis de baixo privilégio.	Tentativas de violação devem retornar <b>HTTP 403 Forbidden</b> .
Análise Estática (SAST)	Vulnerabilidades de Código (Injeção).	Executar análise de segurança de código no pipeline de CI/CD.	Zero vulnerabilidades Críticas ou Altas (OWASP Top 10) no código.
Inclusão/i18n	Tratamento de Mensagens.	Verificar que todas as mensagens de erro/validação são extraídas para <i>Resource Bundles</i> .	Facilidade de adaptação para outros idiomas no futuro.

## 4.2. Performance e Estabilidade

Item	Foco	Estratégia de Teste	Critério de Sucesso
Teste de Carga (Load)	Volume de Usuários/Requisições.	Simular 200 usuários virtuais por 5 minutos nos endpoints de Movimentação/Vendas.	Supor tar a carga com taxa de erro < 1%.
Latência	Tempo de Resposta da API.	Medir o tempo de resposta do endpoint <code>GET /estoque</code> sob carga.	O tempo de resposta (p95) deve ser < 250ms.

<b>Teste de Estresse</b>	Ponto de Falha e Recuperação.	Sobrecarga progressiva até a falha do sistema.	A API deve se recuperar em até 60 segundos após a redução da carga.
--------------------------	-------------------------------	--	---

#### 4.3. Usabilidade e Tratamento de Exceções (Developer Experience - DX)

Item	Foco	Estratégia de Teste	Critério de Sucesso
<b>Consistência REST</b>	Design de Endpoints e Métodos HTTP.	Revisão da documentação OpenAPI e execução de testes de consistência.	Todos os recursos seguem o padrão RESTful (Ex: <b>DELETE /recurso/{id}</b> ).
<b>Tratamento de Exceções</b>	Resposta a Falhas Internas e Erros de Cliente.	Testar <i>payloads</i> inválidos, IDs inexistentes e dependências simuladamente indisponíveis.	Erros de validação retornam <b>400 Bad Request</b> ; Recurso não encontrado retorna <b>404 Not Found</b> .

---

### 5. Critérios de Saída (Go/No-Go para Produção)

Categoria	Critério de Saída (Obrigatório)
<b>Cobertura de Código</b>	Mínimo de <b>85%</b> nas Classes de Serviço (Service Layer).
<b>Bugs</b>	Zero (0) bugs Críticos ou Altos pendentes.
<b>Performance</b>	Todos os requisitos de Latência e Carga devem ser atendidos.

<b>Testes Funcionais</b>	100% dos Casos de Teste de Regressão e Prioridade CRÍTICA Aprovados.
<b>Segurança</b>	100% dos Casos de Teste de Permissão Aprovados e Análise SAST limpa.