

Ciclo de Testes - Configuração no Jira com Zephyr Scale

1. Informações do Ciclo de Testes

Nome do Ciclo: Sprint 1 - Carrinho de Compras - Rodada 1
Projeto: E-commerce
Versão: 1.0.0
Ambiente: Staging
Data de Início: 23/11/2025
Data de Término: 29/11/2025
Responsável: Equipe QA
Status: Planejado

2. Estrutura do Ciclo de Testes

2.1 Hierarquia no Jira

```
PROJETO E-COMMERCE
├─ Epic: Gestão de Carrinho
│   ├── US-001: Adicionar Produto ao Carrinho
│   │   ├── TC-001: Adicionar Produto com Sucesso (Step-by-Step)
│   │   ├── TC-002: Quantidade Inválida (Step-by-Step)
│   │   ├── TC-BDD-001: Diferentes Quantidades (BDD)
│   │   └── TC-BDD-002: Validações e Restrições (BDD)
│   └── US-002: Remover Produto do Carrinho
│       ├── TC-003: Remover Produto com Sucesso
│       ├── TC-004: Cancelar Remoção
│       ├── TC-005: Remover Último Produto
│       └── TC-006: Desfazer Remoção
└─ Test Cycle: Sprint 1 - Carrinho - Rodada 1
    ├── Folder: Funcional
    │   ├── TC-001
    │   ├── TC-002
    │   ├── TC-003
    │   ├── TC-004
    │   ├── TC-005
    │   └── TC-006
    └── Folder: BDD
        ├── TC-BDD-001
        └── TC-BDD-002
```

3. Configuração no Zephyr Scale

3.1 Criar Test Cases

Passo 1: Navegar para Test Cases

Jira → Apps → Zephyr Scale → Test Cases → Create Test Case

Passo 1: Navegar para Test Cases

Jira → Apps → Zephyr Scale → Test Cases → Create Test Case

Passo 2: Preencher Informações do TC-001

Detalhes Básicos:

- **Name:** TC-001: Adicionar Produto ao Carrinho com Sucesso
- **Project:** E-commerce
- **Folder:** Carrinho/Funcional
- **Priority:** High
- **Status:** Draft → Approved
- **Labels:** smoke, regression, funcional

Pré-condições:

- Usuário deve estar logado no sistema
- Produto deve estar disponível no catálogo
- Produto deve ter estoque disponível (quantidade > 0)
- Navegador: Chrome 119+

Test Script (Formato Step-by-Step):

Step	Action	Expected Result	Test Data
1	Acessar https://ecommerce.com.br	Sistema carrega página inicial	-
2	Clicar em "Login"	Modal de login exibido	-
3	Inserir e-mail	E-mail inserido no campo	teste@email.com
4	Inserir senha	Senha inserida (mascarada)	Teste@123
5	Clicar em "Entrar"	Redirecionado para home autenticado	-
6	Navegar para Eletrônicos > Notebooks	Lista de notebooks exibida	-
7	Clicar em "Notebook Dell Inspiron 15"	Página de detalhes carregada	-
8	Alterar quantidade para 2	Campo aceita valor 2	2
9	Clicar "Adicionar ao Carrinho"	Loading exibido	-
10	Aguardar processamento	Mensagem de sucesso exibida	-
11	Verificar contador do carrinho	Contador exibe "2"	-
12	Abrir carrinho	Modal/página do carrinho aberto	-
13	Verificar produto no carrinho	Produto aparece na lista	-
14	Verificar quantidade	Quantidade = 2	-
15	Verificar preço unitário	R\$ 2.200,00	-

13	Verificar produto no carrinho	Produto aparece na lista	-
14	Verificar quantidade	Quantidade = 2	-
15	Verificar preço unitário	R\$ 3.299,00	-
16	Verificar subtotal	R\$ 6.598,00	-
17	Verificar total	R\$ 6.598,00	-

Test Data:

```
Usuário: teste@email.com / Teste@123
Produto: Notebook Dell Inspiron 15
Preço: R$ 3.299,00
Estoque: 50 unidades
Quantidade: 2
```

Attachments:

- Screenshot da página do produto
- Screenshot do carrinho após adição
- Cenário de teste documentado

Custom Fields:

- **Test Type:** Functional
- **Automation Status:** Not Automated
- **Technique:** Equivalence Partitioning
- **Estimated Time:** 10 minutes

Passo 3: Criar demais Test Cases

Repetir o processo para:

- TC-002: Quantidade Inválida (Step-by-Step)
- TC-003 a TC-006: Remoção de produtos
- TC-BDD-001 e TC-BDD-002: Casos BDD

3.2 Criar Test Cycle

Passo 1: Criar o Ciclo

```
Zephyr Scale → Test Cycles → Create Test Cycle
```

Informações do Ciclo:

- **Name:** Sprint 1 - Carrinho de Compras - Rodada 1
- **Project:** E-commerce
- **Folder:** Sprint 1
- **Version:** 1.0.0
- **Status:** Not Executed
- **Planned Start Date:** 23/11/2025
- **Planned End Date:** 29/11/2025

- **Status:** Not Executed
- **Planned Start Date:** 23/11/2025
- **Planned End Date:** 29/11/2025
- **Description:**

Primeiro ciclo de testes do módulo de Carrinho de Compras.
Objetivo: Validar funcionalidades de adicionar e remover produtos.
Ambiente: Staging (<https://staging.ecommerce.com.br>)
Build: v1.0.0-rc1

Passo 2: Adicionar Test Cases ao Ciclo

Método 1: Via Interface

```
Test Cycle → Add Test Cases → Selecionar:  
- TC-001: Adicionar Produto com Sucesso  
- TC-002: Quantidade Inválida  
- TC-003: Remover Produto  
- TC-004: Cancelar Remoção  
- TC-005: Remover Último Produto  
- TC-006: Desfazer Remoção  
- TC-BDD-001: Diferentes Quantidades  
- TC-BDD-002: Validações e Restrições
```

Método 2: Via JQL (Query)

```
project = "E-COMMERCE"  
AND labels IN (smoke, regression)  
AND component = "Carrinho"  
AND priority IN (High, Highest)  
ORDER BY priority DESC, key ASC
```

Passo 3: Configurar Execuções

Para cada Test Case no ciclo:

TC-001: Adicionar Produto com Sucesso

- **Assignee:** João Silva (QA)
- **Environment:** Staging
- **Estimated Time:** 10 minutes
- **Execution Type:** Manual

TC-002: Quantidade Inválida

- **Assignee:** Maria Santos (QA)
- **Environment:** Staging
- **Estimated Time:** 15 minutes
- **Execution Type:** Manual

TC-BDD-001 e TC-BDD-002

- **Assignee:** Pedro Oliveira (QA/Automação)
- **Environment:** Staging

- **Assignee:** Pedro Oliveira (QA/Automação)
- **Environment:** Staging
- **Estimated Time:** 20 minutes cada
- **Execution Type:** Manual (será automatizado)

4. Estrutura de Pastas no Zephyr Scale

4.1 Test Cases - Estrutura de Folders

```
E-commerce/  
├── Carrinho/  
│   ├── Funcional/  
│   │   ├── Adicionar Produto/  
│   │   │   ├── TC-001: Cenário Positivo  
│   │   │   └── TC-002: Validações  
│   │   └── Remover Produto/  
│   │       ├── TC-003: Cenário Positivo  
│   │       ├── TC-004: Cancelamento  
│   │       ├── TC-005: Último Item  
│   │       └── TC-006: Desfazer  
│   └── BDD/  
│       ├── TC-BDD-001: Adicionar Diferentes Quantidades  
│       └── TC-BDD-002: Validações e Restrições  
├── Performance/  
│   └── (futuros testes)  
├── Segurança/  
│   └── (futuros testes)  
└── Checkout/  
    └── (outros módulos)
```

4.2 Test Cycles - Estrutura

```
Sprints/  
├── Sprint 1/  
│   ├── Rodada 1 - Testes Funcionais  
│   ├── Rodada 2 - Correções  
│   └── Regressão Final  
└── Sprint 2/  
    └── (próximas sprints)
```

5. Execução dos Testes

5.1 Iniciar Execução



```
Zephyr Scale → Test Cycles → "Sprint 1 - Carrinho - Rodada 1"  
→ Select Test Case → Execute
```

5.2 Executar Test Case

Para cada passo:

5.2 Executar Test Case

Para cada passo:

1. **Execute o passo** conforme descrito
2. **Registre o resultado:**
 -  **Pass:** Passo executado com sucesso
 -  **Fail:** Passo falhou
 -  **Blocked:** Não foi possível executar
 -  **Skipped:** Pulado intencionalmente
3. **Anexar evidências:**
 - Screenshots
 - Logs
 - Vídeos
 - JSON de requisições
4. **Adicionar comentários** se necessário

5.3 Registrar Defeitos

Se um teste falhar:

1. **Capturar evidências completas:**
 - Screenshot do erro
 - Console logs (F12)
 - Network tab (requisições)
 - Vídeo da reprodução
2. **Criar Bug no Jira:**

```
Project: E-COMMERCE
Issue Type: Bug
Summary: [Carrinho] Produto não é adicionado com quantidade específica

Description:
Ao tentar adicionar um produto com quantidade 2, o sistema adiciona apenas 1 unidade.

Environment: Staging
Build: v1.0.0-rc1
Browser: Chrome 119.0

Steps to Reproduce:
1. Fazer login
2. Navegar para produto
3. Alterar quantidade para 2
4. Clicar em "Adicionar ao Carrinho"

Expected Result: Produto adicionado com quantidade 2
Actual Result: Produto adicionado com quantidade 1

Priority: High
Severity: Major
Labels: bug, carrinho, quantidade

Attachments:
- screenshot_error.png
- console_logs.txt
```

Labels: bug, carrinho, quantidade

Attachments:

- screenshot_error.png
- console_logs.txt
- network_trace.har

3. Linkar Bug ao Test Case:

Zephyr Scale → Test Execution → Link Issue → Select Bug

4. Atualizar Status do Teste:

Status: Failed
Defects: BUG-123
Comments: Bug encontrado e reportado. Teste será reexecutado após correção.

6. Status dos Testes

6.1 Status Disponíveis

Status	Descrição	Ação Seguinte
Not Executed	Teste não iniciado	Executar teste
In Progress	Teste em andamento	Continuar execução
Pass	Teste passou	Nenhuma ação
Fail	Teste falhou	Criar/linkar bug
Blocked	Teste bloqueado	Resolver bloqueio
Skipped	Teste pulado	Documentar motivo

6.2 Critérios de Conclusão

O ciclo é considerado concluído quando:

- ✓ 100% dos casos de teste críticos executados
- ✓ 95% de todos os casos executados
- ✓ Taxa de aprovação $\geq 90\%$
- ✓ Bugs críticos = 0
- ✓ Bugs de alta prioridade analisados
- ✓ Evidências anexadas em todos os testes
- ✓ Relatório gerado e aprovado

7. Relatórios e Métricas

7.1 Gerar Relatórios

Zephyr Scale → Reports → Test Execution Report

7.1 Gerar Relatórios

Zephyr Scale → Reports → Test Execution Report

Relatórios Disponíveis:

- Test Execution Summary**
 - Total de testes
 - Pass/Fail/Blocked
 - Taxa de execução
 - Taxa de aprovação
- Test Results by Priority**
 - Distribuição por prioridade
 - Status de cada prioridade
 - Bugs encontrados por prioridade
- Test Execution Timeline**
 - Gráfico de execuções ao longo do tempo
 - Tendências de pass/fail
 - Velocidade de execução
- Defect Density**
 - Bugs por funcionalidade
 - Severidade dos bugs
 - Taxa de reopen
- Traceability Matrix**
 - Requisitos vs Test Cases
 - Cobertura de testes
 - Gaps de cobertura

7.2 Dashboards

Dashboard Personalizado - QA Overview:

Gadgets:

- Test Execution Status (Pie Chart)
- Test Results by Assignee (Bar Chart)
- Recent Test Executions (Table)
- Open Defects by Priority (Bar Chart)
- Test Coverage by Requirement (Matrix)
- Execution Progress (Line Chart)

Exemplo de Métricas:

SPRINT 1 - CICLO 1 - MÉTRICAS	
Total de Testes:	8
Executados:	8 (100%)
Aprovados:	6 (75%)
Falhados:	2 (25%)

Total de Testes:	8
Executados:	8 (100%)
Aprovados:	6 (75%)
Falhados:	2 (25%)
Bloqueados:	0 (0%)
Bugs Encontrados:	2
- Críticos:	0
- Altos:	1
- Médios:	1
Taxa de Aprovação:	75%
Tempo Total:	2h 30m
Tempo Médio/Teste:	18.75m

8. Boas Práticas

8.1 Durante Execução

- ✓ **Sempre anexar evidências** (screenshots no mínimo)
- ✓ **Documentar desvios** do comportamento esperado
- ✓ **Registrar tempo real** de execução
- ✓ **Atualizar status** imediatamente após execução
- ✓ **Reportar bugs** assim que encontrados
- ✓ **Comunicar bloqueios** imediatamente
- ✓ **Revisar pré-condições** antes de cada teste
- ✓ **Limpar dados de teste** após cada execução

8.2 Nomenclatura e Organização

- ✓ **IDs sequenciais:** TC-001, TC-002, TC-003...
- ✓ **Nomes descritivos:** "Adicionar produto com quantidade válida"
- ✓ **Labels consistentes:** smoke, regression, funcional, bdd
- ✓ **Prioridades alinhadas:** Seguir matriz de priorização
- ✓ **Versionamento claro:** Indicar versão do sistema testado

8.3 Comunicação

- ✓ **Daily updates** sobre progresso dos testes
- ✓ **Notificar PO** sobre bugs críticos imediatamente
- ✓ **Compartilhar relatórios** ao final de cada rodada
- ✓ **Documentar decisões** em comentários
- ✓ **Slack/Teams:** Canal dedicado para QA

9. Checklist de Execução

Antes de Iniciar o Ciclo

- ☐ Ambiente de testes configurado e acessível
- ☐ Build correto deployado
- ☐ Massa de dados preparada
- ☐ Credenciais de teste disponíveis
- ☐ Ferramentas configuradas (browsers, postman, etc)
- ☐ Test cases revisados e aprovados

- ☐ Credenciais de teste disponíveis
- ☐ Ferramentas configuradas (browsers, postman, etc)
- ☐ Test cases revisados e aprovados
- ☐ Team alocado e disponível

Durante o Ciclo

- ☐ Executar testes conforme prioridade
- ☐ Anexar evidências em cada execução
- ☐ Reportar bugs imediatamente
- ☐ Atualizar status no Jira diariamente
- ☐ Comunicar bloqueios e impedimentos
- ☐ Participar das dailies
- ☐ Reteste de bugs corrigidos

Ao Final do Ciclo

- ☐ 100% dos testes críticos executados
- ☐ Todos os bugs reportados e triados
- ☐ Evidências anexadas em todos os testes
- ☐ Relatório de execução gerado
- ☐ Métricas coletadas e documentadas
- ☐ Retrospectiva realizada
- ☐ Lições aprendidas documentadas
- ☐ Aprovação do PO obtida

10. Integração com CI/CD

10.1 Testes Automatizados (Futuro)

```
# .github/workflows/test.yml
name: E2E Tests

on:
  push:
    branches: [main, staging]
  pull_request:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2

      - name: Run E2E Tests
        run: npm run test:e2e

      - name: Upload to Zephyr Scale
        uses: SmartBear/zephyr-scale-action@v1
        with:
          api-key: ${ secrets.ZEPHYR_API_KEY }
          project-key: ECOM
```

```
- name: Upload to Zephyr Scale
uses: SmartBear/zephyr-scale-action@v1
with:
  api-key: ${ secrets.ZEPHYR_API_KEY }
  project-key: ECOM
  test-cycle: Sprint-1-Automated
  results-file: test-results.json
```

10.2 Webhooks

Configurar webhooks para notificações:

- Test execution completo
- Bug crítico encontrado
- Taxa de aprovação < 80%
- Build failure

11. Templates Úteis

Template de Test Case

```
**TC-XXX: [Título]**

**Pré-condições:**
- Lista de pré-condições

**Passos:**
1. Ação → Resultado Esperado
2. Ação → Resultado Esperado
...

**Dados de Teste:**
- Campo: valor

**Critério de Aprovação:**
- Condição de sucesso
```

Template de Bug Report

```
**[Módulo] Título do Bug**

**Ambiente:** Staging / Production
**Severidade:** Critical / Major / Minor
**Prioridade:** High / Medium / Low

**Passos para Reproduzir:**
1. Passo 1
2. Passo 2
...

**Resultado Atual:**
[O que está acontecendo]

**Resultado Esperado:**
```

****Resultado Atual:****

[O que está acontecendo]

****Resultado Esperado:****

[O que deveria acontecer]

****Evidências:****

- Screenshot
- Logs
- Vídeo

Documento preparado por: Equipe QA

Data: 11/11/2025

Versão: 1.0

Próxima Revisão: 01/12/2025