**Primeiro Cenário: Integração de marketplaces terceiros**

**1. Documentação e Materiais de Apoio**

**Identificação da Documentação:**

Os documentos e materiais de apoio utilizados para embasar os testes incluem:

* **Documentação oficial das APIs dos marketplaces** (Amazon, Mercado Livre)
* **Especificações técnicas de integração** fornecidas pela equipe de desenvolvimento
* **Requisitos de negócio**  fornecidas pelo P.O tais como: regras de estoque, precificação, políticas de frete
* **Histórico de requisitos no Jira/Trello**
* **Diagramas de fluxo e arquitetura da integração** (Se disponível)

**Análise da Documentação:**

* Realização de **leitura detalhada** dos documentos para identificar endpoints, campos obrigatórios, regras de negócio e fluxos de integração.
* Criar um **checklist dos pontos críticos** por módulo ( estoque, anúncios, pedidos, faturamento, preço).
* Validar consistência entre requisitos funcionais e técnicos.

**Mapeamento dos Requisitos:**

* Mapear **requisitos funcionais** (sincronizar estoque em tempo real) com **pontos de teste** específicos (produto com estoque zerado deve ser desativado).
* Criar uma **matriz de rastreabilidade** entre requisito < - > cenário de teste.
* Exemplos:
  + REQ01: Atualizar estoque > CT01: Reduzir estoque após pedido confirmado
  + REQ02: Criar anúncio > CT02: Validar criação com título, imagem e preço

**Ferramentas:**

* **Jira / Trello**: controle e rastreabilidade dos requisitos/bugs/melhorias
* **Confluence / Google Docs**: documentação colaborativa (se aplicável)
* **Postman / Swagger**: análise e testes das APIs
* **Miro**: mapeamento visual de fluxos
* **ChatGPT / ferramentas de IA** (como auxílio na geração de testes baseados em requisitos)

**2. Abrangência dos Testes**

**Funcionalidades a serem testadas:**

* **Estoque**: sincronização automática, bloqueio de venda sem estoque, contabilização na compra de mais de um produto ou mais unidades de um único produto.
* **Anúncios**: publicação, alteração e exclusão de produtos
* **Faturamento**: geração de faturas após pedidos confirmados
* **Pedidos**: importação e centralização de pedidos dos marketplaces
* **Preços**: atualização automática de preços conforme regras do sistema

**Cases de Uso:**

* **Cenários de sucesso**:
  + Produto com estoque é sincronizado corretamente
  + Pedido do marketplace aparece no sistema em determinado período de tempo
* **Cenários de falha**:
  + Tentativas falhas de sincronização de produto
  + Erro de autenticação na API do marketplace
* **Possíveis cenários de carga / performance**:
  + Sincronização de alto volume de produtos simultaneamente
  + Importação de pedidos em lote

**Priorização dos Testes:**

Critérios para priorizar os testes:

* **Impacto no negócio**: pedidos e faturamento = alta prioridade
* **Probabilidade de falha**: APIs instáveis, novas integrações = prioridade média
* **Regressões conhecidas**: bugs anteriores devem ter retestes (incluindo testes regressivos em subidas de novas features de correção)

Exemplo de priorização:

| **Cenário** | **Impacto** | **Frequência** | **Prioridade** |
| --- | --- | --- | --- |
| Falha na importação de pedidos | Alta | Alta | Alta |
| Erro de imagem em anúncio | Baixo | Média | Média |
| Atualização de preço | Média | Alta | Alta |



**3. Execução dos Testes**

**Ambiente de Teste:**

* Ambiente de **homologação/stage**, espelhando produção
* Se aplicável: sandbox oficial do marketplace (ex: Amazon MWS Sandbox, Mercado Livre Test Users)
* Banco de dados com cópias anonimizadas de dados reais (BD teste)

**Dados de Teste:**

* Produtos fictícios (com variações, sem estoque, com imagens inválidas)
* Pedidos simulados (itens múltiplos, cupons, frete grátis)
* Usuários de teste com permissões diferentes (admin, vendedor, operador)

**Se aplicável, possíveis ferramentas de Automação:**

* **Postman** com collections para testes de API automatizados
* **Selenium / Cypress** para testes de interface (se aplicável)
* **JMeter** para testes de carga das APIs
* Integração com **GitLab CI** para execução contínua

**Registro de Resultados:**

* Armazenamento de logs em ferramentas como **Grafana**, ou simples planilhas para histórico
* Notificações de falhas via **Slack / e-mail**

**Considerações Finais:**

* Realizar **testes exploratórios** para identificar casos não mapeados
* Avaliar **resiliência da integração** em casos de indisponibilidade dos marketplaces
* Validar a **conformidade de dados sensíveis**  ao trafegar informações entre sistemas

Segundo cenário: Integração de gerenciamento de estoque “BLING”  
  
**1. Documentação e Materiais de Apoio**

**Identificação de Documentação:**

* **Documentação oficial da API do Bling**
* **Especificações técnicas** da integração definidas pela equipe de desenvolvimento
* **Fluxos de integração definidos em reuniões de projeto**
* **Requisitos funcionais e de negócio** relacionados a estoque, pedidos e produtos
* **Histórico de demandas no Jira / Trello / ClickUp**
* **Registros de reuniões e decisões (ata, gravações, Confluence)**

**Mapeamento dos Requisitos:**

* Identificar os **requisitos de integração** (ex: enviar pedidos para o Bling, importar estoque atualizado)
* Criar **cenários de teste para cada requisito**
* Montar uma **tabela de rastreabilidade**, exemplo:

| **Requisito** | **Caso de Teste** |
| --- | --- |
| REQ01: Atualizar estoque no Bling | CT01: Reduzir estoque após venda |
| REQ02: Importar pedidos do Bling | CT02: Validar dados importados |
|  |  |

**Ferramentas:**

Ferramentas para análise de documentação e requisitos:

* **Jira / Trello / Notion** para controle e rastreio de requisitos
* **Confluence / Google Docs** para documentação colaborativa (se aplicavel)
* **Postman / Swagger** para testar as APIs
* **Miro** para desenhar fluxos da integração

**2. Abrangência dos Testes**

**Funcionalidades da Integração:**

As funcionalidades que serão testadas incluem:

* **Sincronização de produtos**: criação, alteração e exclusão
* **Atualização de estoque**: envio e recebimento de dados como a quantidade
* **Importação de pedidos**: pedidos criados no Bling devem aparecer no sistema e vice-versa
* **Processamento de pedidos**: status, faturas, etc.
* **Geração de relatórios**: exportação de dados entre os sistemas

**Cases de Uso:**

* **Cenários de sucesso**:
  + Produto criado com sucesso no Bling após cadastro no sistema
  + Pedido sincronizado corretamente com todos os campos válidos
* **Cenários de falha**:
  + Tentativa de criar produto sem campos obrigatórios
  + Pedido com status inválido não é sincronizado
* **Testes carga/desempenho**:
  + Enviar alto volume de produtos em lote
  + Sincronizar estoque de todos os itens de uma categoria em curto intervalo

**Priorização dos Testes:**

Critérios:

* **Impacto no negócio** como: estoque desatualizado pode gerar vendas erradas
* **Complexidade**: integrações mais complexas são mais propensas a erro
* **Frequência de uso**: funcionalidades mais utilizadas recebem maior prioridade

Exemplo:

| **Cenário** | **Impacto** | **Frequência** | **Prioridade** |
| --- | --- | --- | --- |
| Erro na importação de pedidos | Alta | Alta | Alta |
| Falha na atualização de estoque | Alta | Alta | Alta |
| Problema ao exportar relatório | Baixo | Média | Média |

**3. Execução dos Testes**

**Dados de Teste:**

* Produtos fictícios (com e sem imagens, com variações ou com preços zerados)
* Pedidos de teste com diferentes formas de pagamento, status e descontos
* Estoques simulados (estoque zerado, estoque alto, negativo)
* Empresas e usuários de teste com permissões diferentes no Bling

**Possiveis Ferramentas para Automação:**

* **Postman**: para testes automatizados das APIs do Bling
* **Selenium / Cypress**: se a interface da integração for web e interativa
* **JMeter**: para testes de performance com carga elevada
* **CI/CD (GitHub Actions, GitLab CI)**: para executar testes automaticamente em cada subida

**Documentação de Resultados:**

* **Testrail**: para controle de execuções, sucesso/falha por cenário
* **Relatórios automatizados em PDF / HTML**
* **Logs da aplicação e integração** armazenados e analisados
* Planilhas de controle em Excel/Google (caso não haja ferramenta dedicada)

**Considerações Finais:**

* Realizar testes em **ambiente isolado de homologação**, conectado a ambiente de teste do Bling
* Validar **tratamento de erros** ( logs e mensagens de falha)
* Garantir **resiliência da integração** (fila de reenvio em caso de falha temporária)
* Certificar-se de que as integrações respeitam **prazos contratuais e operacionais pré-definidos**
* Avaliar **segurança e LGPD** no tráfego de informações entre sistemas

Terceiro Cenário: Erro de anuncio: Pausado sem estoque

**Resumo do Problema:**

* Anúncios com **estoque zerado** estão sendo marcados como **“Pausado”** em vez de **“Pausado (sem estoque)”**.
* Permitindo que os anúncios continuem disponíveis no Mercado Livre mesmo **sem estoque real**.
* O cliente não pausou os anúncios manualmente.
* Quando uma **atualização manual de estoque** é feita, a situação muda corretamente para “Pausado sem estoque”.
* O **log da integração** não registra envio automático de estoque = 0 no momento em que o produto esgota.
* Exemplo afetado: [**MLB3097510082**](https://produto.mercadolivre.com.br/MLB-3097510082-escultura-em-pedra-pirita-com-cupula-de-acrilico-_JM)

**Passos de Análise (sem acesso ao painel do cliente)**

**1. Verificar comportamento esperado segundo a documentação oficial do Mercado Livre**

* A API de produtos e publicações do Mercado Livre especifica:
  + Quando o **campo available\_quantity é atualizado para 0**, o anúncio entra automaticamente no status de **“paused\_by\_system”**, ou seja, **"Pausado (sem estoque)"**.
  + Se nenhum envio de available\_quantity = 0 for feito, o anúncio pode permanecer **ativo ou apenas "Pausado"**, mas ainda visível em buscas.

**2. Analisar Logs de Integração (print disponibilizado no github)**

* Nenhum log de envio com available\_quantity = 0 foi encontrado.
* Isso indica que o **Magazord não está disparando automaticamente a atualização de estoque** quando ele chega a zero.

**3. Testes e Validações Técnicas Possíveis (sem acesso ao painel Seller ou ERP)**

**Simulação de comportamento esperado:**

* Criar um produto de teste.
* Reduzir o estoque manualmente no sistema Magazord até **chegar a zero**.
* Verificar se a atualização via API está sendo feita automaticamente.
* Confirmar se o campo available\_quantity está sendo enviado.

**Verificar regras internas Magazord:**

* Existem regras configuráveis no ERP que **definem se o envio de 0 é permitido**?
* Há algum **agendamento** responsável por sincronização de estoque e ele falhou?
* A lógica de “estoque indisponível” pode estar apenas **pausando o anúncio manualmente**, sem atualizar o campo available\_quantity.

**Possíveis hipóteses para o Erro**

| **Hipótese** | **Descrição** |
| --- | --- |
| **1. ERP não envia atualização de estoque 0 automaticamente** | O sistema só dispara atualizações se houver uma redução real ou um evento configurado. Zeragem natural (última unidade vendida) pode não disparar envio. |
| **2. Lógica interna do ERP está apenas pausando anúncios** | A ação de "pausar" pode estar sendo tratada sem considerar o status correto do Mercado Livre (paused\_by\_system). |
| **3. Delay ou falha no service de sincronização** | O sistema responsável por monitorar e sincronizar o estoque pode estar com atraso, falha de execução ou falta de logs. |
| **4. Configuração no cliente desativa envio de estoque 0** | O cliente pode ter alguma configuração que bloqueia esse tipo de envio automático. |

**Soluções e Ações Recomendadas**

1. **Solicitar à equipe técnica** verificação da lógica de sincronização ao atingir estoque 0:
   * Estão disparando a tarefa?
   * Está montando corretamente o payload com available\_quantity = 0?
2. **Incluir testes automatizados** que possam validar se:
   * Quando o estoque chega a zero, a API do ML recebe a atualização.
   * O status muda de “Ativo” para “Pausado (sem estoque)” automaticamente.
3. **Registrar os eventos de estoque zerado no log da integração**:
   * Mesmo que nenhuma ação seja tomada, o log deveria capturar isso para auditorias futuras.
4. **Validar com a equipe Magazord** se há necessidade de configuração especial para esse comportamento.

O problema provavelmente está **na ausência de envio automático de available\_quantity = 0** via API ao esgotar o estoque. Isso impede que o ML mude o status corretamente para **“Pausado (sem estoque)”**, resultando em anúncios ativos indevidamente.

Apesar da falta de acesso ao painel do cliente e seller central, os logs, prints e comportamento descrito apontam para uma **falha na lógica de atualização de estoque** dentro da integração.

Quarto Cenário: Validação de dados cadastrais

**Pontos Principais:**

1. Informações sendo salvas corretamente.
2. Respeitando regras de validação de formato e obrigatoriedade.
3. Mantendo consistência e integridade dos dados.

**Plano de Testes – Validação de Campos Cadastrais**

**1. Campos a Testar (conforme o cenário apresentado no github):**

* Nome completo
* E-mail
* Número de telefone
* Data de nascimento
* Endereço: rua, cidade, estado, CEP

**Testes por Campo**

**1. Nome Completo**

| **Tipo de Teste** | **Descrição** |
| --- | --- |
| Validação básica | Inserir nome com letras e espaços (Pedro Henrique Neri) |
| Nome vazio | Tentar gravar sem preencher o campo |
| Caracteres inválidos | Inserir números ou símbolos (ex: “Pedro123” ou “#%$Ana\*”) |
| Tamanho máximo | Testar o limite de caracteres (nome com tamanho máximo de caracteres pré-definido) |
| Tamanho acima do limite | Inserir nome com número superior de caracteres permitido e tentar gravar |

**2. E-mail**

| **Tipo de Teste** | **Descrição** |
| --- | --- |
| E-mail válido | Testar com formatos válidos ( pedro@dominio.com) |
| E-mail inválido | Testar com formatos errados (ex: pedro@, @dominio.com, pedro@dominio) |
| Campo vazio | Tentar gravar sem e-mail |
| Tamanho limite | Validar campo com caracteres acima do limite pré-definido (nome.muito.longo@dominio.com.br) |

**3. Número de Telefone**

| **Tipo de Teste** | **Descrição** |
| --- | --- |
| Número válido | Inserir número com DDD (ex: (11) 91234-5678) |
| Sem DDD | Tentar inserir número sem DDD |
| Letras ou símbolos | Tentar inserir "ABCD1234@#$" |
| Campo vazio | Tentar salvar sem telefone |
| Formatos alternativos | Validar com e sem máscara |

**4. Data de Nascimento**

| **Tipo de Teste** | **Descrição** |
| --- | --- |
| Data válida | Inserir uma data válida (ex: 10/05/1990) |
| Futuro | Inserir uma data futura |
| Data inválida | "31/02/2020", "00/00/0000" |
| Campo vazio | Tentar salvar sem data |
| Limite de idade (maioridade) | Inserir usuário com menos de 13 anos (se houver regra de idade mínima) |

**5. Endereço (Rua, Cidade, Estado, CEP)**

**Rua**

| **Teste** | **Descrição** |
| --- | --- |
| Nome de rua padrão | "Av. Brasil, 1500" |
| Campo vazio | Tentar gravar sem rua |
| Caracteres inválidos | "@@Rua#123" |
| Tamanho limite | Nome longo com 100+ caracteres |

**Cidade**

| **Teste** | **Descrição** |
| --- | --- |
| Nome comum | "São Paulo" |
| Números ou símbolos | "12345" ou "#Cidade!" |

**Estado (UF)**

| **Teste** | **Descrição** |
| --- | --- |
| Sigla válida | "SP", "SC", "RJ" |
| Sigla inválida | "XX", "SÃO", "Rj" (minúsculo) |
| Campo vazio | Verificar obrigatoriedade |

**CEP**

| **Teste** | **Descrição** |
| --- | --- |
| CEP válido | "89000-000" |
| Sem máscara | "89000000" |
| CEP inválido | "ABCDE-000", "1234" |
| API CEP | Possui uma API que puxa os dados do logradouro após inserção de CEP? |
| Campo vazio | Tentar salvar sem preencher |

**6. Testes Gerais**

| **Tipo** | **Descrição** |
| --- | --- |
| **Persistência de dados** | Após salvar, abrir novamente e verificar se os dados estão corretos |
| **Edição parcial** | Alterar apenas um campo (exemplo: só e-mail) e salvar |
| **Cancelar edição** | Editar, clicar em "Fechar" sem salvar: verificar se dados antigos permanecem sem alteração |
| **Exportação** | Se houver exportação de cadastros, validar se dados exportados estão corretos e atualizados |
| **Edições simultâneas** | Dois usuários editando o mesmo registro ao mesmo tempo (se aplicável) |

**Testes de Interface**

| **Tipo** | **Descrição** |
| --- | --- |
| Máscaras de entrada | Telefone, CPF e CEP devem ter máscara automática |
| Feedbacks visuais | Mensagens de erro apresentadas ao inserir dados inválidos |
| Enunciados das informações | Devem estar claros e intuitivos |
| Botão "Gravar" | Desabilitado enquanto há erro de validação ou falta de preenchimento de informações obrigatórias |

**Conclusão**

Deve ser validado não só o **funcionamento técnico** (dados válidos e inválidos), mas também a **usabilidade** do sistema (máscaras, mensagens, comportamento dos botões). Assim, garantindo que os campos alterados estão respeitando regras de negócio, interface de usuário e integridade dos dados.