**Test Plan Cadastrar Editar e Pesquisar Categorias**

**1**  **Introdução**

Trata-se de um sistema de e-commerce onde  o propósito dele é cadastrar, editar e pesquisar novas categorias a fim de organizar o nossos produtos.

**2**    **Arquitetura**

O framework utilizado para a implementação do back-end do Cadastrar, editar e pesquisar Categorias  é o C#, que possui uma arquitetura orientada a Objetos.

Para o armazenamento e consulta de dados das categorias, será usada uma API que disponibiliza um conjunto de serviços REST. Funciona como na imagem, em que o cliente (aplicação) envia uma requisição à uma URL (endpoint) para acessar os serviços da API, e tem como retorno um objeto na forma de um JSON.

**3**    **Funcionalidades do Cadastro de Categorias**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidades** | **Comportamento Esperado** | **Verificações** | **Critérios de Aceite** |
| **Cadastro** | Deve ser exibido uma caixa de texto para o usuário escrever a categoria que deseja acessar.  Ao digitar uma categoria o usuário deve ser direcionado para a tela onde deve aparecer data de criação | Nome da categoria (obrigatório; até 128 caracteres; somente alfabeto);  Status booleano;  Data de criação.  Usuário não preencher campos obrigatórios  Ultrapassar o limite de caracteres | Uma categoria não pode ter o mesmo nome de outra categoria;  A categoria cadastrada deve ser registrada com o status de ativo;  A categoria cadastrada deve ser registrada com a data e hora de criação. |

**4**    **Funcionalidades da Edição de Categorias**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidades** | **Comportamento Esperado** | **Verificações** | **Critérios de Aceite** |
| **Edição** | Deve ser exibido uma caixa de texto para o usuário escrever a categoria que deseja editar.  Ao digitar uma categoria o usuário deve ser direcionado para a tela onde deve aparecer as opções de edição.  O usuário deve escolher o que vai editar e após feita a edição ele deve visualizar a categoria editada e as datas de criação e alteração | Nome da categoria (obrigatório; até 128 caracteres; somente alfabeto);  Status booleano;  Data de criação.  Data da alteração.  Usuário não preencher campos obrigatórios  Ultrapassar o limite de caracteres | Toda categoria alterada deve ser registrada com a data e hora de modificação. |

**5**     **Funcionalidades da Pesquisa de Categorias.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Funcionalidades** | **Comportamento Esperado** | **Verificações** | **Critérios de Aceite** |
| **Pesquisa** | Deve ser exibido uma caixa de texto para o usuário escrever a categoria que deseja pesquisar.  Ao digitar uma categoria o usuário deve ser direcionado para a tela onde deve aparecer:  Id de registro;  Nome da categoria/subcategoria;  Indicar se é categoria ou subcategoria;  Status;  Data de criação;  Data de modificação. | Nome da categoria (até 128 caracteres; somente alfabeto; informar no mínimo 3 caracteres para a busca);  Status (Todos, Ativos, Inativos);  Apenas Categorias.  Usuário não preencher campos obrigatórios  Ultrapassar o limite de caracteres | É possível combinar os filtros existentes de qualquer forma;  A ordenação deverá ser crescente por nome da categoria (possibilidade de realizar uma ordenação decrescente). |

**6**  **Estratégia de Teste**

●                **Escopo de Testes**

 O plano de testes abrange todas as funcionalidades descritas na tabela acima. Serão executados testes em todos os níveis conforme a descrição abaixo.

Testes Unitários: o código terá uma cobertura de 60% de testes unitários, que são de responsabilidade dos desenvolvedores.

Testes de Integração: Serão executados testes de integração em todos os endpoints, e esses testes serão de responsabilidade do time de qualidade.

Testes Manuais: Todas as funcionalidades serão testadas manualmente pelo time de qualidade seguindo a documentação de Cenários de teste e destes TestPlan.

**7** **Ambiente e Ferramentas**

Os testes serão feitos do ambiente de homologação, e contém as mesmas configurações do ambiente de produção com uma massa de dados gerada previamente pelo time de qualidade.

As seguintes ferramentas serão utilizadas no teste:

|  |  |  |
| --- | --- | --- |
| Ferramenta | Time | Descrição |
| POSTMAN | Qualidade | Ferramenta para realização de testes de API |

**8**   **Classificação de Bugs**

Os Bugs serão classificados com as seguintes severidades:

|  |  |  |
| --- | --- | --- |
| ID | Nivel de Severidade | Descrição |
| 1 | Blocker | ●   Bug que bloqueia o teste de uma função ou feature causa crash na aplicação.  ●  Botão não funciona impedindo o uso completo da funcionalidade.  ●  Bloqueia a entrega. |
| 2 | Grave | ●  Funcionalidade não funciona como o esperado  ●  Input incomum causa efeitos irreversíveis |
| 3 | Moderada | ●  Funcionalidade não atinge certos critérios de aceitação, mas sua funcionalidade em geral não é afetada  ●  Mensagem de erro ou sucesso não é exibida |
| 4 | Pequena | ●  Quase nenhum impacto na funcionalidade porém atrapalha a experiência  ●  Erro ortográfico  ●  Pequenos erros de UI |

**9**   **Definição de Pronto**

Será considerada pronta as funcionalidades que passarem pelas verificações e testes descritas nestes TestPlan, não apresentarem bugs com a severidade acima de Minor, e passarem por uma validação de negócio de responsabilidade do time de produto.