**Nome:** João Chrisóstomo Ribeiro Abegão

**RGM:** 26409577

|  |  |
| --- | --- |
| **Linguagem do *Back-end*** | Python (Django) |
| **Banco de Dados** | SQLite |
| **Hospedagem** | Local |
| **Plataforma** | Django (Framework Web) |
| **Modo de Codificação** | (X) Tradicional  ( ) *Low-code*  ( ) *No-code* |
| ***Link* do repositório no** [**GitHub**](https://github.com/login) **com os códigos abertos** | https://github.com/jchrisostomo/loja-virtual |
| ***Link* da solução em funcionamento** | Local // 127.0.0.1:8080 |
| ***Link* do vídeo narrado (no mínimo 5 min)** | https://youtu.be/T4WDV7H5PhM |

**Projeto Loja de Cupcake**

A aplicação é um sistema de e-commerce desenvolvido com Django que permite aos usuários navegar por produtos, adicioná-los ao carrinho e finalizar compras. A aplicação foi projetada com foco em modularidade e reutilização de código, utilizando boas práticas de desenvolvimento.

**Estrutura da Aplicação**

**Modelos (Models)**

1. Category: Representa uma categoria de produtos com nome e descrição.
2. Product: Representa um produto com nome, descrição, preço, imagem, estoque e referência à categoria.
3. Cart: Representa o carrinho de compras com usuário, produto, quantidade e valor total calculado automaticamente.

**Views**

1. product\_list: Exibe a lista de produtos disponíveis na loja.
2. cart: Mostra os itens adicionados ao carrinho de um usuário.
3. add\_to\_cart: Adiciona um produto ao carrinho com a quantidade desejada.
4. checkout: Gerencia a finalização da compra.

**URLs**

1. /product\_list/: Lista de produtos.
2. /cart/: Visualização do carrinho.
3. /add\_to\_cart/<id>/: Adiciona um produto ao carrinho.
4. /checkout/: Finaliza a compra.

**Diagrama UML**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Testes da Solução**

|  |  |
| --- | --- |
| **Nome:** Maria Vitória França Ribeiro | **Data do teste:** 30/11/2024 |
| **O que testou e funcionou:**  Listagem de produtos na página inicial. | |
| **O que testou e não funcionou – O que deve ser corrigido:** | |
| **Funcionalidade não testada (faltou ou não foi implementada):**  Possibilidade de filtrar os produtos. | |

|  |  |
| --- | --- |
| **Nome:** Camila Aparecida Maia Dias | **Data do teste:** 30/11/2024 |
| **O que testou e funcionou:**  Adição de produto no carrinho. | |
| **O que testou e não funcionou – O que deve ser corrigido:** | |
| **Funcionalidade não testada (faltou ou não foi implementada):**  Opção de remover produto do carrinho. | |

|  |  |
| --- | --- |
| **Nome:** João Diogo Garcia da Fonseca | **Data do teste:** 30/11/2024 |
| **O que testou e funcionou:**  Somatório do valor dos produtos está funcionando corretamente. | |
| **O que testou e não funcionou – O que deve ser corrigido:** | |
| **Funcionalidade não testada (faltou ou não foi implementada):**  Não tem o valor unitário, somente o subtotal por produto e total. | |

|  |  |
| --- | --- |
| **Nome:** Maria Eduarda | **Data do teste:** 30/11/2024 |
| **O que testou e funcionou:**  Ao clicar em finalizar o pedido é executado e aparece uma mensagem informando que o mesmo foi realizado. | |
| **O que testou e não funcionou – O que deve ser corrigido:** | |
| **Funcionalidade não testada (faltou ou não foi implementada):**  Falta campos para informar o tipo de pagamento. | |

|  |  |
| --- | --- |
| **Nome:** Lucas Palis | **Data do teste:** 30/11/2024 |
| **O que testou e funcionou:**  Responsividade do site está funcionando conforme esperado. | |
| **O que testou e não funcionou – O que deve ser corrigido:** | |
| **Funcionalidade não testada (faltou ou não foi implementada):** | |

**Resumo do Laudo de Qualidade do Sistema Loja Virtual**

1. **Descrição:**Sistema desenvolvido em Python (Django) com SQLite, oferecendo exibição de produtos, gerenciamento de carrinho e finalização de compras.
2. **Metodologia:**Testes unitários no Django, validação de formulários, visualizações e inspeções manuais de interações do usuário.
3. **Testes unitários**

* **TESTE 1**. Teste para Criar uma Categoria: Verifica se uma categoria é criada corretamente com nome e descrição atribuídos.
* **TESTE 2**. Teste para Criar um Produto: Confirma que um produto é criado com atributos corretos, incluindo associação com uma categoria.
* **TESTE 3**. Teste para Adicionar um Produto ao Carrinho: Avalia a adição de um produto ao carrinho, verificando usuário, produto, quantidade e cálculo total do item.
* **TESTE 4**. Teste para Total do Carrinho: Valida o cálculo do total do carrinho com múltiplos produtos adicionados por um usuário.
* **TESTE 5**. Teste de URL para Listar Produtos: Confirma que a URL para listar produtos resolve corretamente para a função de view correspondente.
* **TESTE 6**. Teste de URL para o Carrinho: Verifica se a URL do carrinho resolve para a função de view correta.
* **TESTE 7**. Teste de URL para Adicionar Produto ao Carrinho: Testa a resolução da URL de adicionar produto ao carrinho para a função apropriada.
* **TESTE 8**. Teste de URL para Finalizar Compra: Certifica que a URL de checkout está associada à função de view de finalização de compra.
* **TESTE 9**. Teste da View de Lista de Produtos: Verifica se a view de listagem de produtos retorna um status HTTP 200 e se exibe o produto criado corretamente.

1. **Resultados:**

* Unitários: 9 testes executados com sucesso.

Texto

Descrição gerada automaticamente

* **Conclusão:**O sistema possui boa qualidade no back-end, mas recomenda-se:
* Testes de integração e usabilidade no front-end.
* Garantir rotas configuradas para todas as funcionalidades.

**Vídeo da Solução atualizada**

|  |  |
| --- | --- |
| ***Link* para o vídeo** | **https://youtu.be/T4WDV7H5PhM** |