

DOCUMENTAÇÃO

```
① README.md > # projetonogit
1  # projetonogit
2  //Detalhes do projeto, como o objetivo e funcionalidades.
3
4  Este projeto tem como objetivo desenvolver um sistema simples
   de processamento de pagamentos em Python, utilizando os
   conceitos de Programação Orientada a Objetos e Classes
   Abstratas. O sistema implementa dois métodos principais de
   pagamento: por cartão de crédito e boleto. Ele simula a
   validação de cartões, geração de boletos e processamento de
   pagamentos.
5
6  Funcionalidade: Pagamento com Cartão de Crédito (
   Processamento de pagamentos com exibição de confirmação e
   detalhes do cartão.) e Boleto ( Processamento de pagamentos
   com exibição de confirmação e o código de barras gerado.
   Código de barras fictício).
7  |
```

No processo de desenvolvimento, primeiramente foi preciso criar um arquivo chamado “**README.md**”, onde se encontram as informações do projeto, como o seu objetivo principal seria a criação de sistema de dados na linguagem **Python**. Conforme a POO, utilizará conceitos de classes **Abstratas**. O sistema se baseia em duas formas de pagamento, o cartão de crédito e por boleto.

E sua funcionalidade seria as formas de pagamento, onde seriam exibidas as informações do cartão (caso seja pago no crédito), e os detalhes do boleto, como a geração de código de barras fictício.

formapagamento.py

```

1 from abc import ABC, abstractmethod
2 import random
3
4 # Classe abstrata Pagamento
5 class Pagamento(ABC):
6     @abstractmethod
7     def processar_pagamento(self, valor):
8         pass
9
10 # Subclasse PagamentoCartao
11 class PagamentoCartao(Pagamento):
12     def __init__(self, numero_cartao):
13         self.numero_cartao = numero_cartao
14
15     def validar_cartao(self):
16         # Validar se o número do cartão tiver 16 dígitos
17         return len(self.numero_cartao) == 16 and self.numero_cartao.isdigit()
18
19     def processar_pagamento(self, valor):
20         if self.validar_cartao():
21             print(f"Pagamento confirmado: R${valor:.2f}; Via cartão de crédito: {self.numero_cartao}")
22         else:
23             print("Número de cartão inválido. Não foi possível processar o pagamento.")
24
25 # Subclasse PagamentoBoleto
26 class PagamentoBoleto(Pagamento):
27     def __init__(self):
28         self.codigo_barras = None
29
30     def gerar_boleto(self):
31         # Gerar um código de barras irreal (com apenas números aleatórios)
32         self.codigo_barras = ''.join(str(random.randint(0, 9)) for _ in range(20))
33         return self.codigo_barras
34
35     def processar_pagamento(self, valor):
36         if not self.codigo_barras:
37             print("Nenhum código de barras gerado. Gerando um agora...")
38             self.gerar_boleto()
39         print(f"\nPagamento confirmado: R${valor:.2f}; Via boleto. \nCódigo de barras: {self.codigo_barras}")
40
41 # Exemplo de uso
42 if __name__ == "__main__":
43     # Pagamento com cartão
44     pagamento_cartao = PagamentoCartao("4502678930571705")
45     pagamento_cartao.processar_pagamento(500.00)
46
47     # Pagamento com boleto
48     pagamento_boleto = PagamentoBoleto()
49     pagamento_boleto.gerar_boleto()
50     pagamento_boleto.processar_pagamento(750.00)
```

Logo em seguida, o arquivo onde se encontra o sistema (**formapagamento.py**). A classe abstrata é a classe **Pagamento**. E as subclasses **PagamentoCartao** e **PagamentoBoleto**, que herdam os métodos da classe Abstrata. E o sistema possui funções para processar o pagamento e validar o cartão.

DESAFIOS

Criar um projeto no Git, foi sem dúvidas um desafio, pois exige concentração e pelo fato que alguns erros acabam aparecendo. Mas o Git foi uma forma de ajuda para separar por etapas todo o projeto, tornando o gerenciamento mais eficiente.