Ganhe Dinheiro Programando para Si

Igor
Nascimento

Quem sou eu?

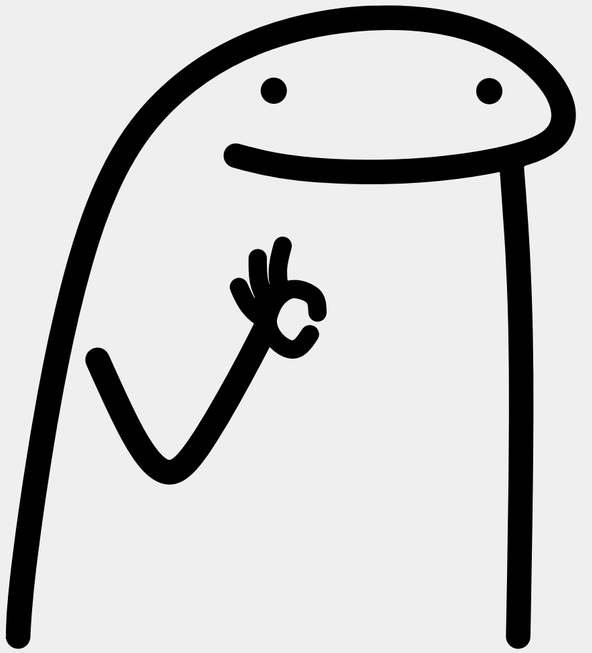
Oi! Sou Igor, engenheiro de software com vasta experiência em desenvolvimento Python e foco em Inteligência Artificial e automação

Minha paixão por inovação e desafios me impulsiona a contribuir em projetos globais de educação e a colaborar em artigos com especialistas da minha área. Sempre aberto a novas oportunidades e projetos que promovam o avanço tecnológico.

Quais formas de ganhar dinheiro?

- Serviços
- Vendas
- Negócios
- Mercados Futuros

Mercado de Milhas



Milhas são pontos acumulados por consumidores ao utilizarem cartões de crédito, comprarem passagens aéreas ou aderirem a programas de fidelidade. Esses pontos podem ser usados para adquirir produtos, passagens e serviços ou serem convertidos em dinheiro ao serem vendidos em plataformas específicas. O mercado de milhas se torna lucrativo graças à possibilidade de multiplicação desses pontos durante promoções oferecidas por bancos e companhias aéreas, onde os pontos acumulados podem ser duplicados, triplicados ou até multiplicados por cinco vezes.

A automação é fundamental para maximizar ganhos, pois permite captar rapidamente oportunidades como promoções-relâmpago de multiplicação de milhas, manter cotações atualizadas e realizar transações no momento mais vantajoso. Em um mercado competitivo e veloz, a agilidade proporcionada pela automação faz uma diferença significativa, possibilitando maior precisão e rapidez em operações, que são essenciais para obter melhores margens de lucro.

Estrutura Técnica

1

CAPTURA DE DADOS

A captura de dados é essencial para embasar decisões e otimizar estratégias com informações precisas e atualizadas.

2

MINERAÇÃO DE DADOS

Ganhamos precisão e agilidade com a automatização na análise gráfica de cotações e promoções.

3

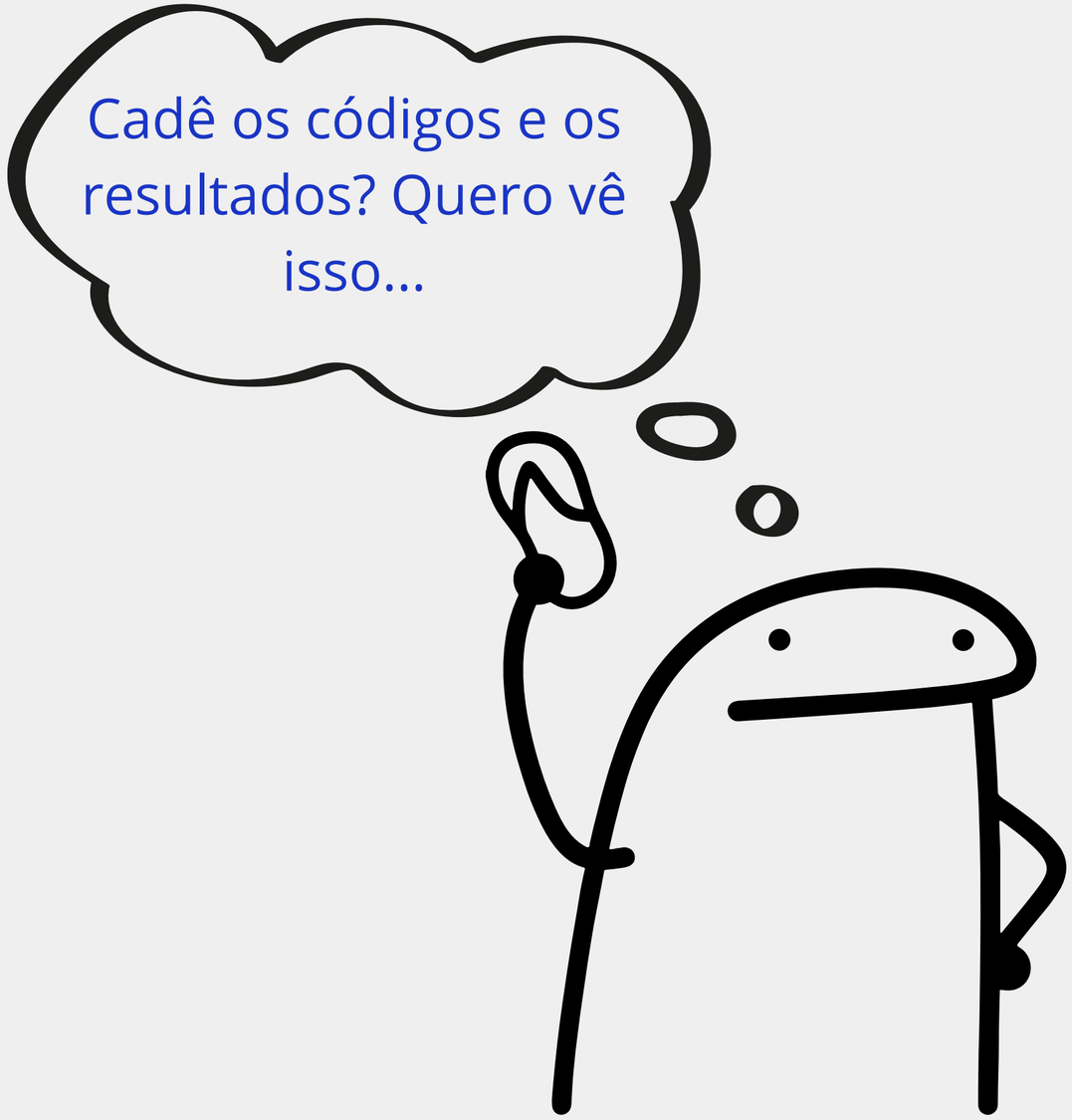
OPERAÇÃO

Análise de dados históricos com desafios e sujeira, dificultando estratégias, enquanto o mercado ainda opera de forma muito manual.

"Seja o exemplo daquilo que você deseja ver realizado, antes de pedir que os outros o façam."

"Resolva problemas e agregue valor."

"Os dados refletem o mercado e não mentem."



Ganhos Atuais

Cotação [redacted] Data: 18/12/23 11:15 **Aprovado em 18/12/23**

Programa:	Quantidade:	Recebimento:	Valor pago:
TUDO AZUL	153.000 Milhas	18/01/24	R$ 3.443,78

Conta para depósito

Banco:	Agência:	Conta:	CPF:
C6 Bank	0001	[redacted]	[redacted]

Cotação [redacted] Data: 18/11/23 16:15 **Aprovado em 18/11/23**

Programa:	Quantidade:	Recebimento:	Valor pago:
TUDO AZUL	51.000 Milhas	18/12/23	R$ 1.147,58

Conta para depósito

Banco:	Agência:	Conta:	CPF:
C6 Bank	0001	[redacted]	[redacted]

Cotação [redacted] Data: 22/03/24 18:12 **Aprovado em 22/03/24**

Programa:	Quantidade:	Recebimento:	Valor pago:
TUDO AZUL	44.000 Milhas	24/04/24	R$ 947,44

Conta para depósito

Banco:	Agência:	Conta:	CPF:
C6 Bank	0001	[redacted]	[redacted]

Cotação [redacted] Data: 18/08/23 15:15 **Aprovado em 18/08/23**

Programa:	Quantidade:	Recebimento:	Valor pago:
TUDO AZUL	51.000 Milhas	23/08/23	R$ 1.147,58

Conta para depósito

Banco:	Agência:	Conta:	CPF:
C6 Bank	0001	[redacted]	[redacted]

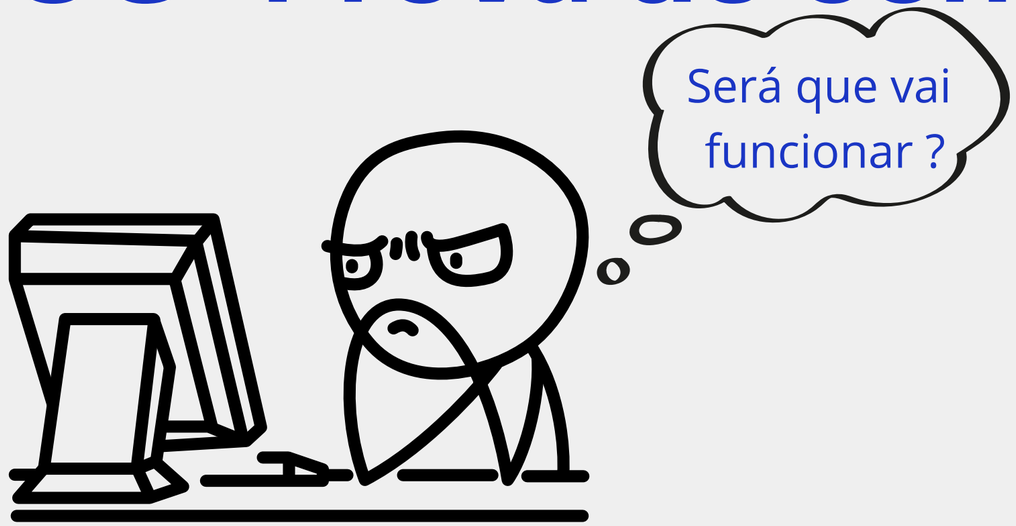
Cotação [redacted] Data: 20/03/24 18:12 **Aprovado em 20/03/24**

Programa:	Quantidade:	Recebimento:	Valor pago:
TUDO AZUL	43.000 Milhas	22/04/24	R$ 937,44

Conta para depósito

Banco:	Agência:	Conta:	CPF:
C6 Bank	0001	[redacted]	[redacted]

POC “Prova de conceito”



IMNascimento foi adicionado o arquivo excel — 1431aa9 · last year — 11 Commits

__pycache__	Foi alterado o arquivo CSV e o bankmilhas	last year
CsvHandling.py	Foi alterado o arquivo CSV e o bankmilhas	last year
LICENSE	Initial commit	last year
README.md	criação da branch da primeira versão bot	last year
bankmilhas.py	Foi alterado o arquivo CSV e o bankmilhas	last year
dicionario.txt	Foi adicionado os itens iniciais do sistema	last year
emailRead.py	Foi alterado o arquivo CSV e o bankmilhas	last year
historicos.xlsx	foi adicionado o arquivo excel	last year
hotmilhas.py	Foi adicionado o envio de email da hotmilhas	last year
maxmilhas.py	foi adicionado o script para escrever o dado no excel	last year
pass	Foi adicionado a leitura de email	last year

```python
class HotBot(FuncionarioMilhas):
    __points = 0
    __million_values = []
    __pattern = r"R\$\s?(\d+(?:\.\d+)?)"
    __sent_from = 'hotmilhas@hotmilhas.com.br'
    __url = 'https://hotmilhas.com.br/obrigado/'
    __store_data = []
    __miles_pattern = {
        1 :  r"\sSMILES\s",
        2 :  r"\sLATAM\sPASS",
        3 :  r"\sTUDO\sAZUL",
        8 :  r"\sLATAM\sPASS",
    }

    def __init__(self, company:list, total_quote:int, flag_name: list)-> None:
        self.companies = company
        self.total_quote_value = total_quote
        self.service_name = flag_name
        self.__points = int(self._total_quote_value/1000)
        self.name = "HOTMILHAS"
        super().__init__()
        self._payment_term = [150,120,90,60,30,1]



    def send_email(self,company:int)->None:
        requests.post(self.__url, data={'email':self._username, 'cia': company, 'points': self.__points})


    def read_email(self, company:int)->None:
        mil_values = []
        mail = Imbox(self._host, username=self._username, password=self._password, ssl=True, ssl_context=None, starttls=False)
        messages = mail.messages(sent_from=self.__sent_from,date__on=self._date_now)
        try:
            for (uid, message) in messages:
                msg = f"{message.body}"
                soup =  BeautifulSoup(msg, "html.parser")

            for check in soup.find_all("strong"):
                a = str(check.get_text)
                regex = re.search(self.__miles_pattern[company], a)
```

```python
from abc import ABC, abstractmethod
import requests
import json
from models.historic import Historic
from models.promotions import Promotions
from datetime import datetime
import re



class FuncionarioMilhas(ABC):
    """
    Interface que define métodos que todas as classes de bot de milhas devem implementar.
    """

    def __init__(self) -> None:
        self._json = json
        self._requests = requests
        self._historic = Historic
        self._promotion = Promotions
        self._mile_quotes = []
        self._datetime = datetime
        self._date_now = datetime.now()
        self._re = re

    @abstractmethod
    def _request_miles_api(self, company: str) -> None:
        """Deve implementar a lógica de requisição à API de milhas."""
        pass

    @abstractmethod
    def _store_miles_data(self, name: str, data_benefit: dict) -> None:
        """Deve implementar a lógica para armazenar dados de milhas no banco de dados."""
        pass

    @abstractmethod
    def work(self) -> int:
        """Deve implementar a lógica principal do bot."""
        pass
```

Classe Pai

"Para um código sustentável e flexível, aplicamos princípios como responsabilidade única, injeção de dependência, testes unitários e uso de abstrações (interfaces e classes abstratas), aliados a herança, polimorfismo e encapsulamento. A aplicação desses conceitos e práticas, como ORM, composição e agregação, surge de acordo com a necessidade do projeto, promovendo modularidade e escalabilidade conforme as demandas evoluem."

Refatoração

Old

```python
def store_miles_data(self, name:str, data_benefit:list)->None:
    i  = 0
    for price in data_benefit:
        self._historic.insert(program= name, price = price, payment_term = self._payment_term[i],\
            buyer_company= self.name, amount= self._total_quote_value, quotation_date=self._date_now).execute()
        i+=1



def work(self)->int:
    i = 0
    try:
        for company in self._companies:
            self.send_email(company)
            self.wait_time(5)
            self.read_email(company)
        for benefit in self._service_name:
            self.store_miles_data(benefit, self.__million_values[i])
            i += 1
        return 200
    except Exception as e:
        self.store_errors(e)
```

New

```python
def _store_miles_data(self) -> None:
    """Armazena as cotações de milhas no banco de dados."""
    try:
        for milha in self._mile_quotes:
            self._historic.insert(**milha.to_dict()).execute()
        self.__logger.info("Dados de milhas armazenados com sucesso.")
    except Exception as e:
        self._handle_error("Erro ao armazenar dados de milhas", e)


def work(self) -> int:
    """Método principal para realizar o trabalho do bot."""
    try:
        self.__logger.info(f"Iniciando trabalho do bot {self.__name}.")

        self.login()

        for program_code, program_name in self.service_names.items():
            self._request_miles_api(program_code, program_name)

        self._store_miles_data()

        self.__logger.info(f"Bot {self.__name} finalizou o trabalho com sucesso.")
        return 200
    except Exception as e:
        self._handle_error(f"Erro ao trabalhar com bot {self.__name}", e)
        return 500
```

```python
class HotBot(FuncionarioMilhas):
    __points = 0
    __million_values = []
    __pattern = r"R\$\s?(\d+(?:\.\d+)?)"
    __sent_from = 'hotmilhas@hotmilhas.com.br'
    __url = 'https://hotmilhas.com.br/obrigado/'
    __store_data = []
    __miles_pattern = {
        1 :  r"\sSMILES\s",
        2 :  r"\sLATAM\sPASS",
        3 :  r"\sTUDO\sAZUL",
        8 :  r"\sLATAM\sPASS",
    }

    def __init__(self, company:list, total_quote:int, flag_name: list)-> None:
        self.companies = company
        self.total_quote_value = total_quote
        self.service_name = flag_name
        self.__points = int(self._total_quote_value/1000)
        self.name = "HOTMILHAS"
        super().__init__()
        self._payment_term = [150,120,90,60,30,1]



    def send_email(self,company:int)->None:
        requests.post(self.__url, data={'email':self._username, 'cia': company, 'points': self.__points})


    def read_email(self, company:int)->None:
        mil_values = []
        mail = Imbox(self._host, username=self._username, password=self._password, ssl=True, ssl_context=None, starttls
        messages = mail.messages(sent_from=self.__sent_from,date__on=self._date_now)
        try:
            for (uid, message) in messages:
                msg = f"{message.body}"
                soup =  BeautifulSoup(msg, "html.parser")

            for check in soup.find_all("strong"):
                a = str(check.get_text)
                regex = re.search(self.__miles_pattern[company], a)
```

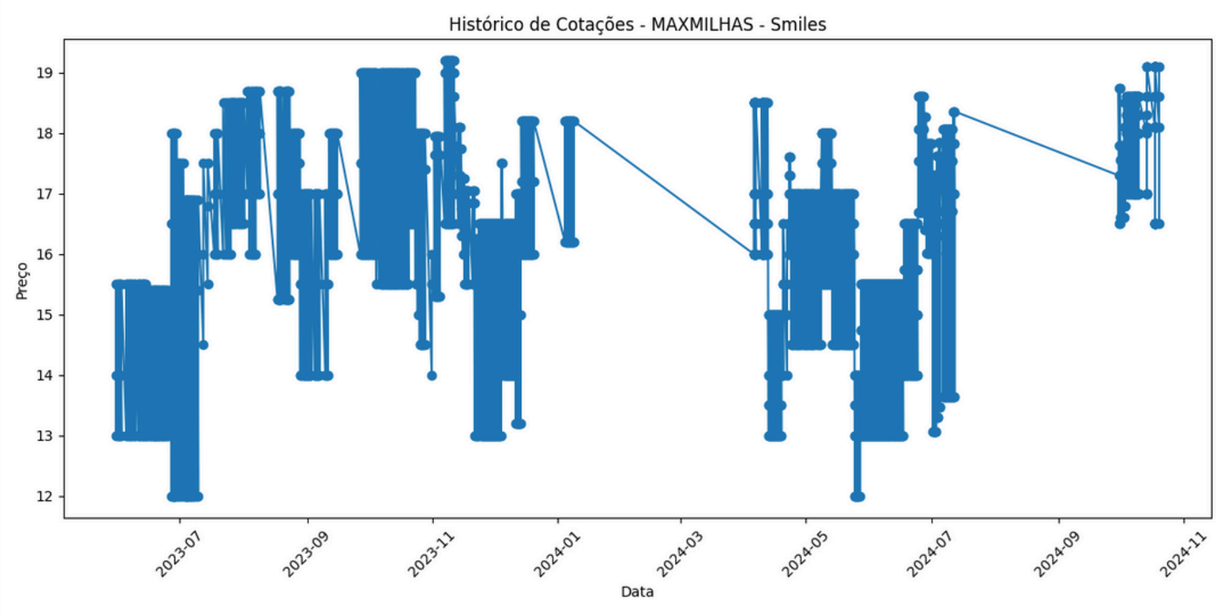
```python
class HotBot(FuncionarioMilhas):
    def __init__(self, services_names: dict, total_quote: int, logger: Logger = None) -> None:
        self.__login_url = Settings.HOT_URL_API_LOGIN
        self.service_names = services_names
        self.total_quote_value = total_quote
        self.__token_autenticate = None
        self.__name = "HOTMILHAS"
        self.__points = self.__calculate_points(self.total_quote_value)  # Pontos baseados no tot
        super().__init__()
        self.__logger = logger or Logger(log_file=f"logs/{self.__name}.log", level=logging.INFO)
        self.__logger.info(f"Bot {self.__name} inicializado.")

        self._auth_handler = APILoginHandler(
            login_url=self.__login_url,
            payload={"email": Settings.HOT_MILES_USER, "password": Settings.HOT_MILES_PASSWORD},
            token_key="auth_token",
            logger=self.__logger
        )

    def __calculate_points(self, total_quote: int) -> int:
        """Calcula os pontos baseados no total da cotação"""
        return total_quote // 1000

    def login(self) -> None:
        """Chama o método de login da classe de autenticação."""
        self._auth_handler.login()
        self.__token_autenticate = self._auth_handler.get_token()

    def __process_miles_data(self, program_name: str, receipts: dict) -> None:
        """Processa e armazena os dados de milhas como objetos `Milhas`."""
        try:
            for payment_term, price in receipts.items():
                milhas = Milhas(
                    program=program_name,
                    price=price // self.__points,
                    payment_term=int(payment_term),
                    buyer_company=self.__name,
                    amount=self.__total_quote_value,
                    quotation_date=datetime.now().strftime("%Y-%m-%d")
                )
                self._mile_quotes.append(milhas)
        except Exception as e:
            self._handle_error(f"Erro ao processar dados de milhas para {program_name}", e)

    def _store_miles_data(self) -> None:
        """Armazena as cotações de milhas no banco de dados."""
        try:
            for milha in self._mile_quotes:
                self._historic_insert(**milha.to_dict()).execute()
```

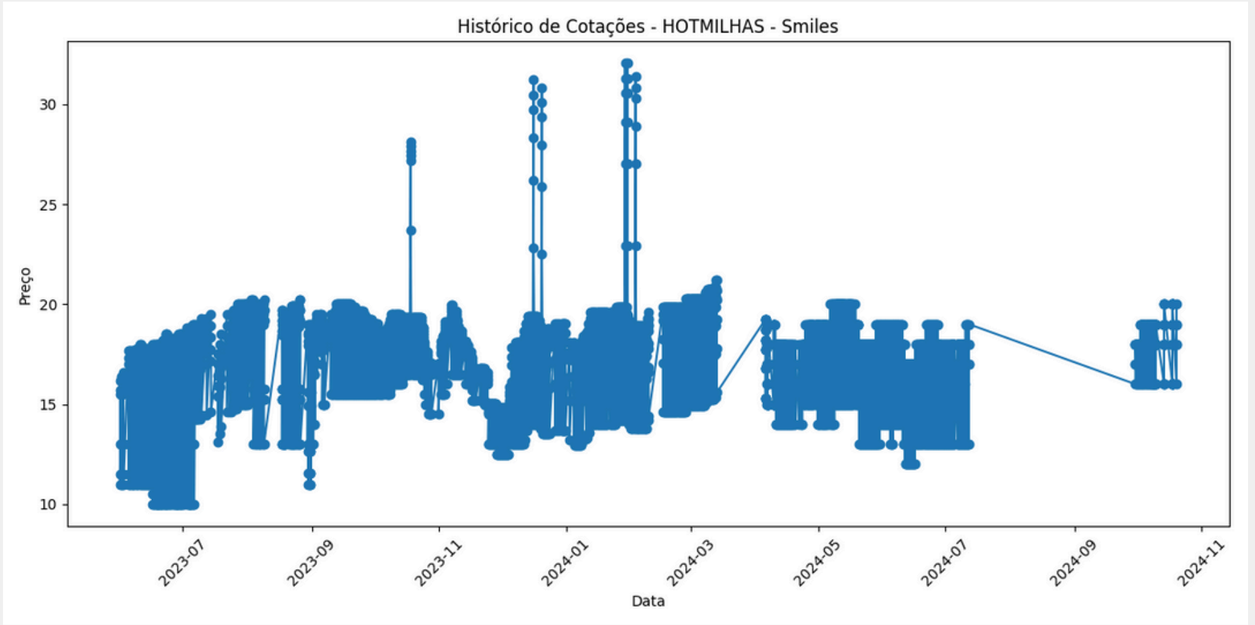
"Segurança e confiabilidade começam com testes. Garanta que sua aplicação funcione como esperado e evite surpresas com testes automatizados."

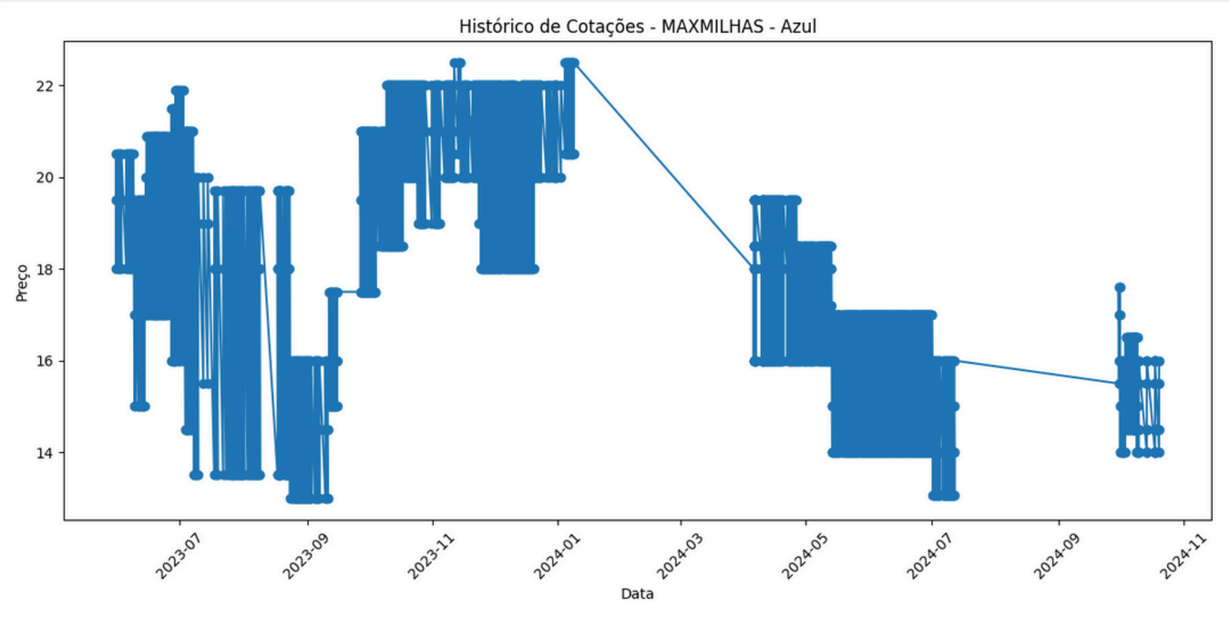
PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS COMENTÁRIOS

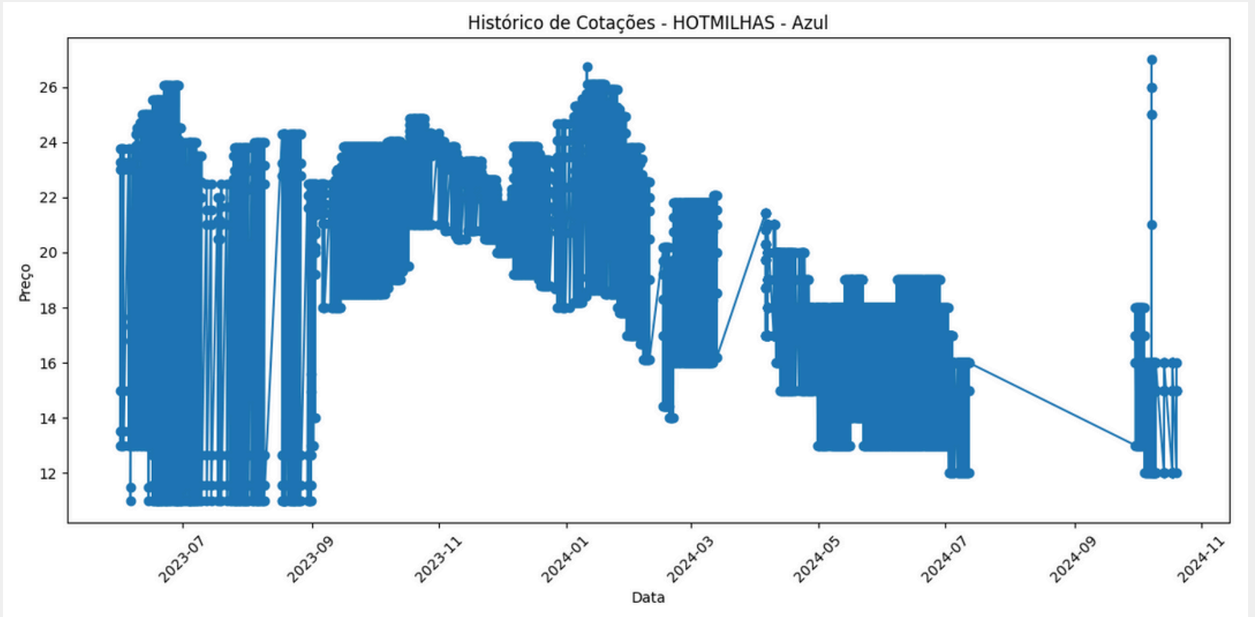
```
2024-10-20 02:44:49,292 - logs/HOTMILHAS.log - INFO - Dados de milhas solicitados com sucesso para Smiles.
.2024-10-20 02:44:49,293 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:49,293 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:49,293 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:49,293 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:49,295 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:49,295 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:49,295 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:49,295 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
<Response [200]>
2024-10-20 02:44:50,533 - logs/HOTMILHAS.log - INFO - Token gerado com sucesso.
2024-10-20 02:44:50,533 - logs/HOTMILHAS.log - INFO - Token gerado com sucesso.
2024-10-20 02:44:50,533 - logs/HOTMILHAS.log - INFO - Token gerado com sucesso.
2024-10-20 02:44:50,533 - logs/HOTMILHAS.log - INFO - Token gerado com sucesso.
.2024-10-20 02:44:50,534 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:50,534 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:50,534 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:50,534 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:50,534 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS inicializado.
2024-10-20 02:44:50,537 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:50,537 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:50,537 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:50,537 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:50,537 - logs/HOTMILHAS.log - INFO - Iniciando trabalho do bot HOTMILHAS.
2024-10-20 02:44:50,538 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,538 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,538 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,538 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,538 - logs/HOTMILHAS.log - INFO - Bot HOTMILHAS finalizou o trabalho com sucesso.
.....<MagicMock name='post()' id='2504716429072'>
..<MagicMock name='post()' id='2504727719632'>
.<MagicMock name='post()' id='2504727775888'>
.2024-10-20 02:44:50,558 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,560 - logs/test_maxbot.log - INFO - Dados de milhas solicitados com sucesso para gol.
.2024-10-20 02:44:50,560 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,560 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,561 - logs/test_maxbot.log - INFO - Iniciando trabalho do bot MAXMILHAS.
2024-10-20 02:44:50,561 - logs/test_maxbot.log - INFO - Iniciando trabalho do bot MAXMILHAS.
2024-10-20 02:44:50,562 - logs/test_maxbot.log - ERROR - Erro ao trabalhar bot MAXMILHAS: Erro na API
2024-10-20 02:44:50,562 - logs/test_maxbot.log - ERROR - Erro ao trabalhar bot MAXMILHAS: Erro na API
.2024-10-20 02:44:50,563 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,563 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,563 - logs/test_maxbot.log - INFO - Bot MAXMILHAS inicializado.
2024-10-20 02:44:50,564 - logs/test_maxbot.log - INFO - Iniciando trabalho do bot MAXMILHAS.
2024-10-20 02:44:50,564 - logs/test_maxbot.log - INFO - Iniciando trabalho do bot MAXMILHAS.
2024-10-20 02:44:50,564 - logs/test_maxbot.log - INFO - Iniciando trabalho do bot MAXMILHAS.
2024-10-20 02:44:50,566 - logs/test_maxbot.log - INFO - Bot MAXMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,566 - logs/test_maxbot.log - INFO - Bot MAXMILHAS finalizou o trabalho com sucesso.
2024-10-20 02:44:50,566 - logs/test_maxbot.log - INFO - Bot MAXMILHAS finalizou o trabalho com sucesso.
.........................
----------------------------------------------------------------------
Ran 47 tests in 1.767s

OK
PS C:\xampp\htdocs\python\MileageSystems\src>
```

Dados para decisão

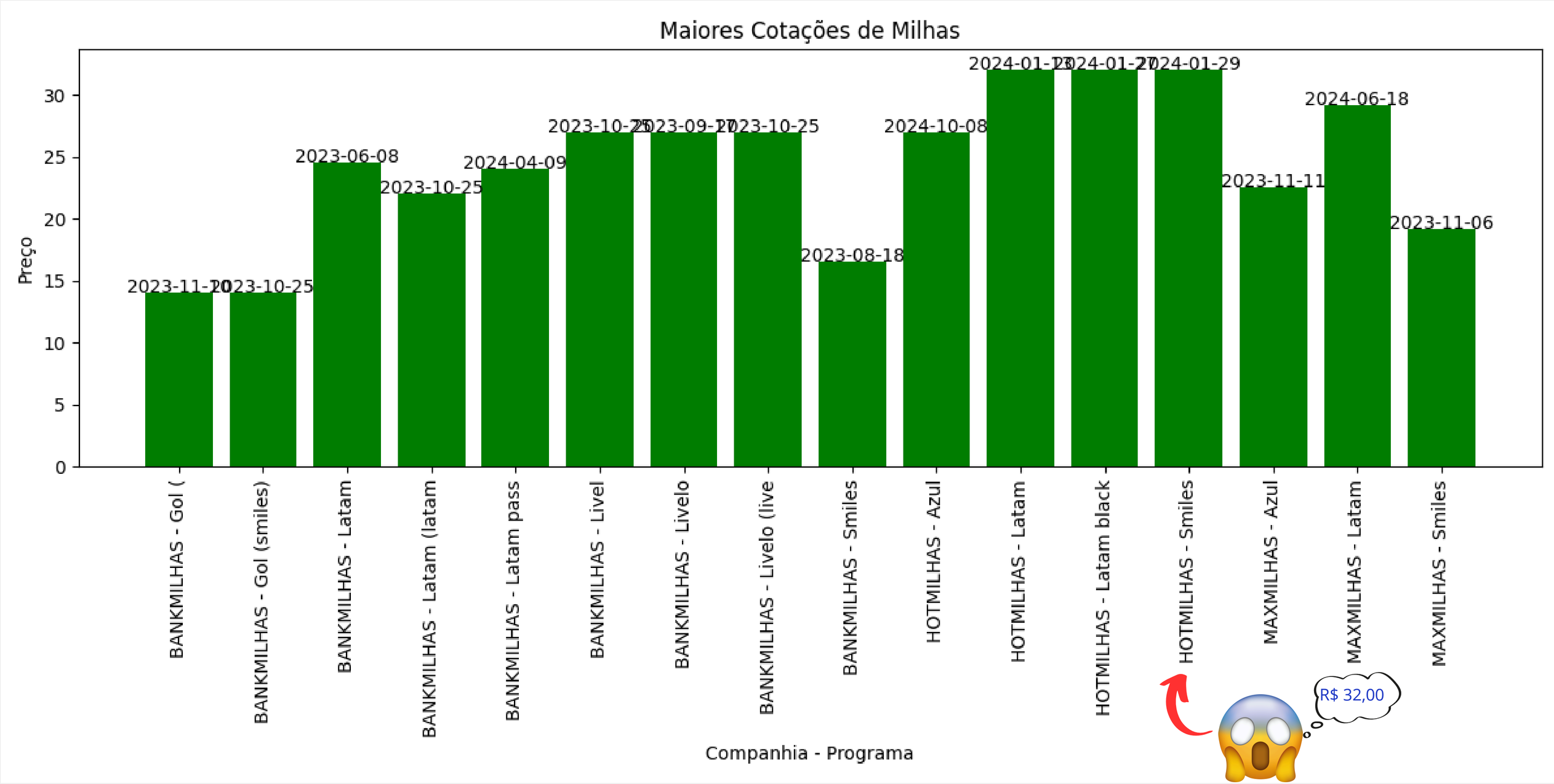








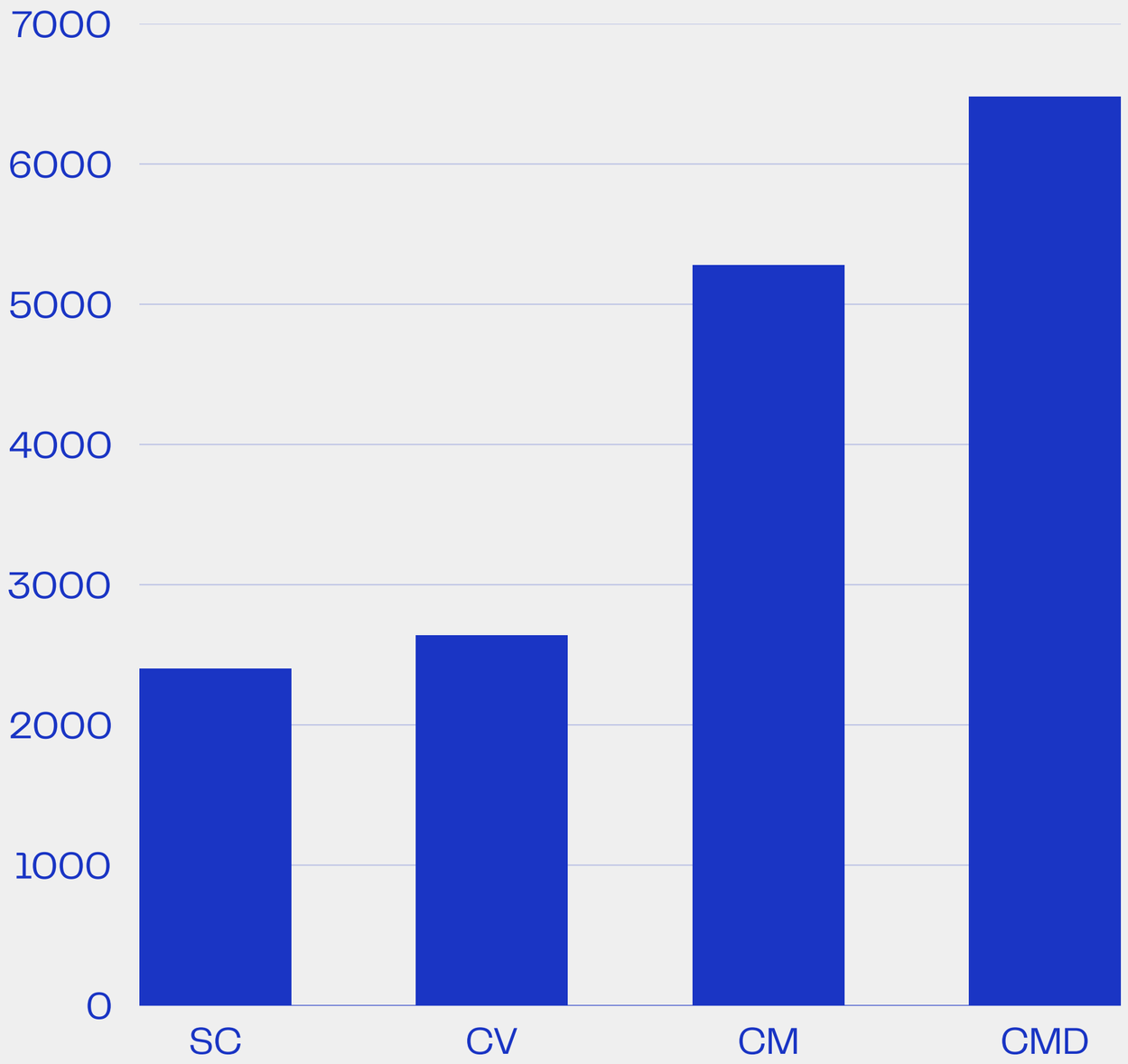
Dados Históricos



Diferenças entre vendas

Simularemos as vendas com 120 mil milhas acumuladas ao ano, cerca de 10 mil milhas por mês. A multiplicação será feita com o programa de milhas da Azul, considerando uma cotação média de R$ 22,00. Com base em análise de dados, verificamos que a cotação chegou a atingir valores superiores a R$ 27,00.

- SC = Sem conversão.
- CV = Conversão para venda
- CM = Conversão e mutiplicação
- CMD = Con. e mult. base. em dados



Análise de Retorno de investimento “ROI”

Usando um cartão com 3 pontos por dólar a R$ 5,30, seria necessário consumir R$ 212.042,00 ao ano. Isso rende um cashback de apenas 1%, mas com análise de dados, o retorno sobe para mais de 3%.

Investindo esse valor em compra de pontos na Azul, o ROI médio sobe de 27,9% para 56,97% com estratégia de dados. Em promoções especiais de assinatura, a multiplicação chega a 150%, proporcionando um ROI de até 96,21%.

O impacto para você ou empresa

A automação no mercado de milhas gera ganhos de capital ao capturar rapidamente oportunidades de multiplicação e venda, maximizando lucros e otimizando o fluxo de caixa.

Liberdade financeira e renda extra

Aproveite o mercado de milhas para gerar novas fontes de receita.

Oportunidades de negócio

Use a automação para explorar promoções e estratégias lucrativas.

Desenvolvimento como programador

Aprimore habilidades técnicas aplicadas ao mercado real.

Em resumo:

Investir na automatização no mercado de milhas permite aproveitar oportunidades de lucro que surgem rapidamente, como promoções de multiplicação de milhas e cotações favoráveis. A automação traz agilidade e precisão para capturar essas oportunidades, maximizando os ganhos com eficiência e escalabilidade. Além disso, oferece uma fonte de renda extra, desenvolvimento de habilidades técnicas e possibilidade de explorar um mercado em constante crescimento e inovação.

Obrigado!

Igor Muniz Nascimento

www.sophialabs.com.br

@igor-nascimento-a76a29155

@igosjn99

IMNascimento