UNIVERSIDADE DO ESTADO DA BAHIA - UNEB
DEPARTAMENTO DE CIÊNCIAS EXATAS E DA TERRA I

Desenvolvimento de programa em linguagem Python para aplicação em Segurança de Obras na Engenharia Civil

Trabalho apresentado a UNEB para obtenção de nota na disciplina Computação Aplicada à Engenharia.

Discentes: Beatriz Ribeiro, Gustavo Barcelos, Tais Maria
Orientador: Prof° Dr° Robson Marinho

Salvador
2023

Sumário

Resumo

Este trabalho apresenta o desenvolvimento de uma ferramenta computacional voltada ao planejamento de medidas de segurança em projetos residenciais, considerando a moradia e a capacidade de investimento do usuário. Com base em conceitos de Engenharia Civil e utilizando linguagem de programação Python, o sistema realiza uma análise simplificada das necessidades do cliente e recomenda soluções de segurança escaláveis, como sensores de movimento, câmeras de vigilância e sistemas de alarme. O objetivo principal é promover uma abordagem prática e acessível, integrando tecnologia e planejamento preventivo aos projetos habitacionais. A ferramenta também serve como recurso didático, incentivando estudantes e profissionais a incorporarem a segurança patrimonial desde as etapas iniciais da concepção da obra. Os resultados esperados incluem maior consciência sobre a importância da segurança nos projetos de Engenharia Civil e a valorização de imóveis por meio da adoção de soluções integradas, econômicas e eficazes.

1 Introdução

A segurança patrimonial é um tema que vem ganhando destaque nas últimas décadas, especialmente nos centros urbanos, onde o crescimento populacional, a verticalização das cidades e o aumento nos índices de criminalidade têm gerado uma demanda crescente por soluções preventivas e tecnológicas. Dentro desse contexto, o papel do engenheiro civil transcende a simples execução de obras: envolve também a capacidade de prever, planejar e integrar sistemas que garantam proteção aos usuários do espaço construído.

Na Engenharia Civil, a concepção de projetos seguros e eficientes não deve considerar apenas a estabilidade estrutural ou o desempenho das edificações frente às ações naturais e humanas. A proteção contra invasões, furtos e situações de risco tornou-se uma necessidade básica, exigindo que os profissionais da área se atentem também à segurança como um item indispensável na fase de planejamento. Isso vale tanto para edificações horizontais, como casas, quanto para verticais, como apartamentos, cada qual com suas especificidades e vulnerabilidades.

Neste trabalho, propõe-se o desenvolvimento de uma ferramenta simples e interativa, construída com base em lógica de programação, que auxilia engenheiros e estudantes da área a identificarem e recomendarem medidas de segurança de acordo com o tipo de moradia e o orçamento disponível do cliente. A proposta visa não apenas indicar soluções compatíveis com diferentes faixas de investimento, mas também promover uma cultura de prevenção no setor da construção civil, incentivando a adoção de tecnologias como sensores de movimento, sistemas de alarme e câmeras de vigilância, de forma racional e acessível.

A partir de um breve questionário, o programa determina o perfil da residência e sugere um conjunto de medidas adequadas, priorizando custo-benefício e viabilidade técnica. Essa abordagem tem como objetivo principal tornar o processo de decisão mais claro para o profissional e, ao mesmo tempo, contribuir para o bem-estar e a segurança dos futuros moradores. Além disso, o uso de ferramentas computacionais simples demonstra como a interdisciplinaridade entre a engenharia e a tecnologia pode favorecer soluções práticas e inovadoras dentro do ambiente acadêmico e profissional.

Portanto, este trabalho não apenas ilustra a importância da segurança no planejamento residencial, mas também reforça a necessidade de preparar engenheiros civis cada vez mais atentos às demandas sociais, econômicas e tecnológicas do mundo contemporâneo. A proposta destaca o compromisso da Engenharia Civil com a criação de ambientes seguros, funcionais e adaptados à realidade de cada cliente, sem abrir mão da responsabilidade técnica e do olhar humano.

1.1 Motivações e justificativas

A segurança nas edificações residenciais representa um dos principais anseios da população brasileira, sobretudo em áreas urbanas marcadas por vulnerabilidades sociais e elevado índice de criminalidade. Nesse cenário, a adoção de soluções tecnológicas se mostra fundamental para mitigar riscos e promover a tranquilidade dos moradores. Apesar disso, ainda é comum que projetos de Engenharia Civil deixem de integrar a segurança patrimonial no planejamento inicial, tratando-a como um item secundário ou opcional.

Ao compreender a segurança como parte integrante da qualidade do projeto, este trabalho se propõe a abordar o tema de maneira objetiva e acessível, utilizando ferramentas simples da programação para criar uma interface capaz de recomendar soluções conforme as características da moradia e o orçamento disponível. A escolha por esse caminho visa não apenas a valorização do imóvel, mas também a formação de profissionais mais completos, que associam conhecimentos técnicos a uma visão ampla e socialmente responsável da construção

civil.

1.2 Objetivo

Desenvolver uma ferramenta interativa para auxiliar no planejamento de segurança em moradias, com base no orçamento disponível, aplicando conceitos de Engenharia Civil aliados à lógica de programação e à análise de custo-benefício.

Nossos objetivos específicos são:

- Identificar as principais necessidades de segurança patrimonial em moradia;

- Mapear os principais dispositivos e sistemas de segurança disponíveis no mercado, considerando preços médios e eficiência;

- Criar um sistema de entrada de dados que permita ao usuário informar a moradia, e a verba disponível;

- Gerar recomendações automáticas de soluções de segurança adequadas ao perfil do imóvel e à capacidade de investimento;

- Incentivar a adoção de práticas de planejamento preventivo no desenvolvimento de projetos residenciais em Engenharia Civil.

1.3 Metodologia de Pesquisa

Este trabalho foi desenvolvido por meio de uma abordagem qualitativa e aplicada, com base na análise de dados de mercado sobre equipamentos de segurança residencial e no desenvolvimento de um algoritmo programado em linguagem Python. A metodologia adotada envolveu as seguintes etapas:

1- Levantamento de informações técnicas e de mercado, incluindo tipos de dispositivos de segurança (alarmes, câmeras, sensores), preços médios e funcionalidades;

2- Classificação das soluções de segurança conforme a complexidade, eficiência e investimento exigido, dividindo as recomendações para a moradia.

3- Desenvolvimento do algoritmo lógico, capaz de receber entradas do usuário como moradia e orçamento, e gerar com base em condicionais e faixas de investimento, uma recomendação personalizada;

4- Testes e validações, realizados para verificar a coerência das respostas geradas pelo programa frente às diferentes combinações possíveis de entrada;

5- Análise dos resultados e considerações sobre a viabilidade da ferramenta como apoio prático ao planejamento de segurança em projetos reais de Engenharia Civil.

2 Referencial teórico

O desenvolvimento do programa está baseado em conceitos fundamentais de programação estruturada, validação de entrada de dados, lógica condicional, interação humano-computador (IHC) e automação de recomendações. Esses conceitos são aplicados para construir um sistema funcional, capaz de interagir com o usuário, interpretar seu orçamento e sugerir soluções de segurança residencial adequadas.

A programação estruturada é uma metodologia de desenvolvimento de software que organiza o código por meio de estruturas lógicas bem definidas, como blocos de decisão e repetição. Essa abordagem facilita a leitura, manutenção e depuração do programa (Pressman, 2016). No código proposto, estruturas como if, elif, else e while são utilizadas para controlar o fluxo do sistema, tomando decisões com base nas informações fornecidas pelo usuário.

A validação de entrada de dados é um aspecto essencial em programas interativos. Ela garante que os valores informados pelo usuário estejam dentro de um formato aceitável, prevenindo falhas no sistema e melhorando a confiabilidade do programa (Pereira, 2020). O programa implementa uma função específica para essa validação, inclusive convertendo vírgulas e tratando caracteres indevidos.

Outro conceito presente é a lógica condicional aplicada à personalização de recomendações. Com base no valor informado pelo usuário, o sistema avalia diferentes faixas de orçamento e apresenta sugestões personalizadas de produtos de segurança, simulando um processo de tomada de decisão automatizada.

Por fim, a interação humano-computador (IHC) é abordada por meio da interface textual, que foi desenvolvida para ser clara, objetiva e responsiva. Segundo Preece et al. (2013), sistemas bem projetados devem se adaptar ao perfil do usuário, facilitando a navegação e a compreensão das etapas do processo.

3 Proposta do programa

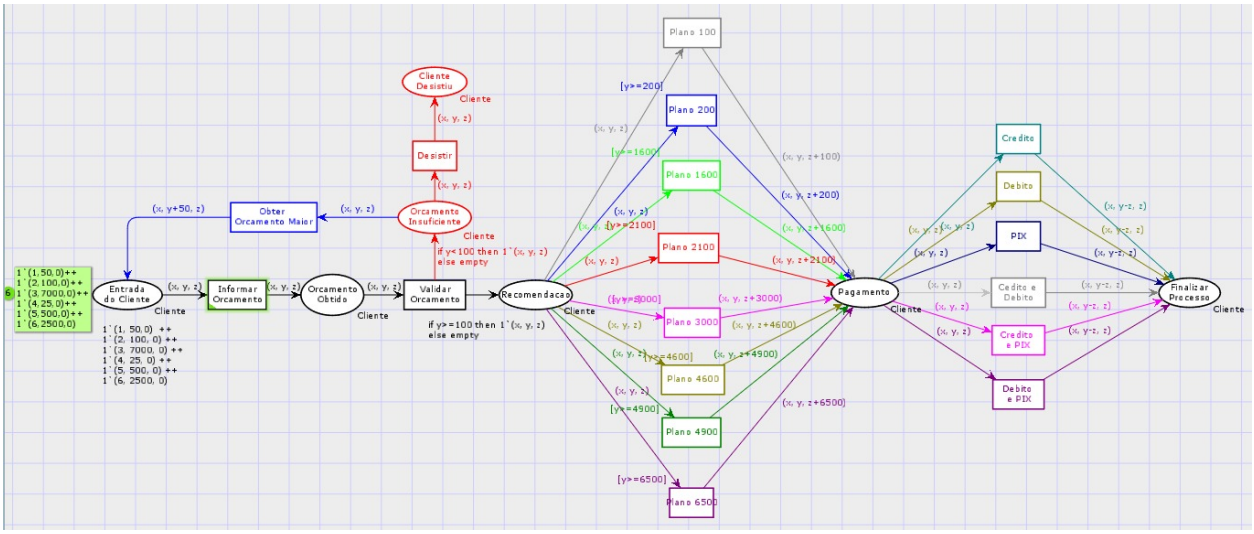
O programa foi criado com o objetivo de auxiliar usuários na escolha de medidas de segurança residencial, considerando a moradia e o valor disponível para investimento. Através de um sistema interativo, o programa realiza perguntas diretas e, com base nas respostas, oferece sugestões personalizadas de equipamentos de segurança. A proposta central é tornar o planejamento da segurança mais acessível, prático e econômico, promovendo a conscientização sobre o investimento em proteção patrimonial.

O principal objetivo do programa é orientar os usuários na definição de um plano de segurança residencial, baseado em critérios simples: a moradia e o orçamento disponível. A ferramenta busca fornecer recomendações específicas e realistas, permitindo que o usuário tome decisões mais seguras e eficientes, sem desperdiçar recursos financeiros.

A Descrição Narrativa inclui que o funcionamento do programa inicia com uma saudação e uma breve explicação sobre sua finalidade. Em seguida, o usuário é questionado sobre a moradia almejada e o valor que pretende investir em segurança. Após processar essas informações, o sistema compara o valor informado com faixas de preços predefinidas e apresenta sugestões compatíveis, aos itens sugeridos. Cada recomendação é pensada para equilibrar custo e benefício, promovendo uma solução adequada à realidade do usuário. O programa simula uma pequena consultoria automatizada, auxiliando na tomada de decisões que unem confronto e acessibilidade.

Como mostrado na ilustração abaixo do programa em Rede de Petri, o sistema segue uma lógica clara desde a apresentação até o encerramento da compra. Porém, a Rede de Petri abaixo ainda não consta todas as melhorias feitas desde a última apresentação do projeto:

A Rede de Petri é apresentada na página seguinte:



4 Script preliminar em Python

Nosso programa consiste nos seguintes passos: apresentar a empresa que fornece o serviço de segurança; apresentar os itens e preços dos materiais mais importantes em estoque; pedimos ao usuário que nos informe o valor que ele quer investir e a área que será coberta pelas câmeras para usarmos nos cálculos; informamos as opções de pagamento; entregamos o resultado e perguntamos se o usuário deseja rodar o sistema novamente (com opcão de colocar um novo valor ou nova forma de pagamento).

O código é:

```
import json
from datetime import datetime
from collections import defaultdict
import os


class ItemSeguranca:
    def __init__(self, codigo, nome, categoria, preco, especificacoes, pr
        self.codigo = codigo
        self.nome = nome
        self.categoria = categoria
        self.preco = preco
        self.especificacoes = especificacoes
        self.prioridade = prioridade  # 1 a 5 (5    mais priorit rio)

    def __str__(self):
        return f"{self.codigo}: {self.nome} - R${self.preco:.2f} (Priorid

    def info_completa(self):
        return (f"{self.codigo}: {self.nome} ({self.categoria})\n"
                f"Pre o: R${self.preco:.2f}\n"
                f"Especifica   es: {self.especificacoes}\n"
                f"Prioridade: {self.prioridade}/5")
```

```
class Orcamento:
    def __init__(self, itens):
        self.data = datetime.now()
        self.itens = [(item, qtd) for item, qtd in itens.items()]
        self.total = sum(item.preco * qtd for item, qtd in self.itens)


class Historico:
    def __init__(self):
        self.orcamentos = []
        self.arquivo = "historico_orcamentos.json"
        self._carregar_historico()

    def _carregar_historico(self):
        if os.path.exists(self.arquivo):
            with open(self.arquivo, 'r') as f:
                dados = json.load(f)
                for orc in dados:
                    itens = [(ItemSeguranca(**item['item']), item['qtd'])
                    novo_orc = Orcamento({})
                    novo_orc.itens = itens
                    novo_orc.total = orc['total']
                    novo_orc.data = datetime.strptime(orc['data'], '%Y-%m
                    self.orcamentos.append(novo_orc)

    def _salvar_historico(self):
        dados = []
        for orc in self.orcamentos:
            itens = [{'item': {'codigo': item.codigo, 'nome': item.nome,
                               'preco': item.preco, 'especificacoes': ite
                               'prioridade': item.prioridade}, 'qtd': qtd
            dados.append({'data': str(orc.data), 'itens': itens, 'total':

        with open(self.arquivo, 'w') as f:
            json.dump(dados, f)

    def adicionar_orcamento(self, orcamento):
        self.orcamentos.append(orcamento)
        self._salvar_historico()

    def listar_orcamentos(self):
        return sorted(self.orcamentos, key=lambda x: x.data, reverse=True

    def comparar_orcamentos(self, idx1, idx2):
        try:
            orc1 = self.orcamentos[idx1]
            orc2 = self.orcamentos[idx2]

            print("\nCOMPARA O DE ORAMENTOS:")
            print(f"\nOr amento 1 ({orc1.data}): R${orc1.total:.2f}")
```

```
            for item, qtd in orc1.itens:
                print(f"- {qtd}x {item.nome}")

            print(f"\nOr amento 2 ({orc2.data}): R${orc2.total:.2f}")
            for item, qtd in orc2.itens:
                print(f"- {qtd}x {item.nome}")

            diferenca = orc1.total - orc2.total
            if diferenca > 0:
                print(f"\nO Or amento 1   R${diferenca:.2f} mais caro q
            elif diferenca < 0:
                print(f"\nO Or amento 1   R${-diferenca:.2f} mais barat
            else:
                print("\nOs or amentos t m o mesmo valor")

        except IndexError:
            print(" ndice inv lido no hist rico")


class SistemaSeguranca:
    def __init__(self):
        self.itens = self._carregar_itens()
        self.categorias = list(set(item.categoria for item in self.itens)
        self.historico = Historico()
        self.categorias_selecionadas = []
        self.orcamento_cliente = 0

    def _carregar_itens(self):
        """Carrega todos os itens de seguran a dispon veis"""
        return [
            # C meras -> Tipo A
            ItemSeguranca("CAM0", "C meras B sicas de monitoramento", "
            ItemSeguranca("CAM1", "C mera Simples 720p", "C mera", 129.
            ItemSeguranca("CAM2", "C mera Full HD 1080p", "C mera", 249
            ItemSeguranca("CAM3", "C mera 360  4K", "C mera", 499.90,
            ItemSeguranca("CAM4", "C mera com IA", "C mera", 699.90, "I

            # Sensores -> Tipo B
            ItemSeguranca("SEN1", "Sensor de Porta/Janela", "Sensor", 59.
            ItemSeguranca("SEN2", "Sensor de Movimento PIR", "Sensor", 89
            ItemSeguranca("SEN3", "Sensor de Vidro Quebrado", "Sensor", 1
            ItemSeguranca("SEN4", "Sensor de Inunda  o", "Sensor", 149.

            # Alarmes -> Tipo C
            ItemSeguranca("ALA1", "Alarme de inc ndio", "Alarme", 49.90,
            ItemSeguranca("ALA2", "Alarme Sonoro B sico", "Alarme", 99.9
            ItemSeguranca("ALA3", "Alarme com Monitoramento", "Alarme", 1
            ItemSeguranca("ALA4", "Alarme Sem Fio", "Alarme", 599.90, "Si

            # Fechaduras -> Tipo D
```

```
            ItemSeguranca("FEC1", "Tranca de janela", "Fechadura", 15.90,
            ItemSeguranca("FEC2", "Trava de porta", "Fechadura", 34.90, "
            ItemSeguranca("FEC3", "Fechadura Digital", "Fechadura", 349.9
            ItemSeguranca("FEC4", "Fechadura Biom trica", "Fechadura", 8

            # Acess rios -> Tipo E
            ItemSeguranca("ACES1", "Lumin ria com Sensor", "Acess rio",
            ItemSeguranca("ACES2", "Interfone Inteligente", "Acess rio",
        ]

    def apresentacao_da_empresa(self):
        """Apresenta  o em destaque da empresa CALCULANDO COM SEGURAN
        print("=" * 60)
        print("---------------          BEM      -VINDO-  -CALCULANDO-COM-
        print("=" * 60)
        print("      -Somos-uma-empresa-focada-em-proteger-o-que-  -seu."
        print("      -Atuamos-no-planejamento-inteligente-da-seguran a-r
        print("      -Nosso-foco-  -a-combina  o-entre-economia,-pratic
        print("         --Vamos-juntos-encontrar-a-melhor-solu  o-para-
        print("=" * 60)

    def mostrar_menu_principal(self):
        """Exibe o menu principal do sistema"""
        print("\n" + "="*50)
        print("SISTEMA-DE-ORAMENTO-DE-SEGURANA-RESIDENCIAL")
        print("="*50)
        print("\n[1].-Ver-todos-os-itens-dispon veis")
        print("[2].-Selecionar-categoria-dos-itens")
        print("[3].-Definir-or amento-e-ver-itens-dentro-do-valor")
        print("[4].-Gerar-or amento-com-itens-selecionados")
        print("[5].-Consultar-hist rico")
        print("[6].-Comparar-or amentos")
        print("[7].-Sair")

    def mostrar_todos_itens(self):
        """Mostra todos os itens dispon veis organizados por categoria"
        print("\nITENS-DISPON VEIS:")

        itens_por_categoria = defaultdict(list)
        for item in self.itens:
            itens_por_categoria[item.categoria].append(item)

        for categoria, itens in itens_por_categoria.items():
            print(f"\n[{categoria.upper()}]")
            for item in sorted(itens, key=lambda x: x.codigo, reverse=Tru
                print(f"--{item}")

    def mostrar_itens_dentro_orcamento(self):
        """Mostra apenas os itens que cabem no or amento do cliente"""
        if self.orcamento_cliente <= 0:
```

```
            print("\nPor favor, defina primeiro um or amento v lido (o
            return

        print(f"\nITENS DENTRO DO ORAMENTO (R$ {self.orcamento_cliente:

        itens_por_categoria = defaultdict(list)
        for item in self.itens:
            # Mostra apenas itens com pre o menor ou igual ao or amento
            if item.preco <= self.orcamento_cliente:
                itens_por_categoria[item.categoria].append(item)

        if not itens_por_categoria:
            print("\nNenhum item dispon vel dentro deste or amento.")
            print("Sugerimos aumentar seu or amento ou ver itens mais b
            return

        for categoria, itens in itens_por_categoria.items():
            print(f"\n[{categoria.upper()}]")
            for item in sorted(itens, key=lambda x: (-x.prioridade, x.pre
                print(f"- {item}")

    def definir_orcamento_cliente(self):
        """Obt m e define o valor m ximo que o cliente deseja investir"
        while True:
            preco_input = input("\nInforme o valor que pretende investir
            preco_input = preco_input.replace(',', '.').replace('-', '')

            try:
                self.orcamento_cliente = float(preco_input)
                if self.orcamento_cliente <= 0:
                    print("O valor deve ser maior que zero.")
                    continue

                print(f"\nOr amento definido: R$ {self.orcamento_cliente
                self.mostrar_itens_dentro_orcamento()
                break

            except ValueError:
                print("Por favor, insira um valor num rico v lido.")

    def selecionar_categorias(self):
        """Permite selecionar m ltiplas categorias"""
        print("\nCATEGORIAS DISPON VEIS:")
        for i, categoria in enumerate(self.categorias, 1):
            print(f"{i}. {categoria}")

        selecao = input("\nDigite os n meros das categorias (ex: 1,3): "
        self.categorias_selecionadas = []

        try:
```

```
            indices = [int(i.strip()) - 1 for i in selecao.split(",") if
            for idx in indices:
                if 0 <= idx < len(self.categorias):
                    self.categorias_selecionadas.append(self.categorias[i

            if self.categorias_selecionadas:
                print("\nCategorias selecionadas:")
                for cat in self.categorias_selecionadas:
                    print(f"- {cat}")
            else:
                print("...")
        except ValueError:
            print("Entrada inv lida. Use n meros separados por v rgula

    def mostrar_itens_categorias_selecionadas(self):
        """Mostra itens apenas das categorias selecionadas"""
        if not self.categorias_selecionadas:
            print("Nenhuma categoria selecionada. Mostrando todos os iten
            self.mostrar_todos_itens()
            return

        print("\nITENS DAS CATEGORIAS SELECIONADAS:")
        for categoria in self.categorias_selecionadas:
            print(f"\n[{categoria.upper()}]")
            itens_cat = [item for item in self.itens if item.categoria ==
            for item in sorted(itens_cat, key=lambda x: (-x.prioridade, x
                print(f"  - {item}")

    def selecionar_quantidades(self):
        """Permite ao usu rio selecionar itens e quantidades dentro do o
        if self.orcamento_cliente <= 0:
            print("\nPor favor, defina primeiro um or amento (op   o 3
            return {}

        selecao = {}
        saldo_disponivel = self.orcamento_cliente
        self.mostrar_itens_dentro_orcamento()

        while True:
            print(f"\nSaldo dispon vel: R$ {saldo_disponivel:.2f}")
            codigo = input("\nDigite o c digo do item (ou 'fim' para ter

            if codigo == 'FIM':
                break

            item = next((i for i in self.itens if i.codigo == codigo), No
            if not item:
                print("C digo inv lido. Tente novamente.")
                continue
```

```
            if item.preco > saldo_disponivel:
                print(f"Este item custa R${item.preco:.2f} e excede seu
                continue

            try:
                max_qtd = min(10, int(saldo_disponivel // item.preco))
                if max_qtd < 1:
                    print("N o h  saldo suficiente para este item.")
                    continue

                quantidade = int(input(f"Quantidade para {item.nome} (max
                if quantidade <= 0 or quantidade > max_qtd:
                    print(f"Quantidade deve ser entre 1 e {max_qtd}.")
                    continue

                custo_total = item.preco * quantidade
                if custo_total > saldo_disponivel:
                    print(f"Isso excede seu saldo dispon vel em R${cust
                    continue

                selecao[item] = quantidade
                saldo_disponivel -= custo_total
                print(f"Adicionado: {quantidade}x {item.nome} (R${custo_
                print(f"Novo saldo: R${saldo_disponivel:.2f}")

            except ValueError:
                print("Por favor, digite um n mero v lido.")

        return selecao

    def gerar_orcamento(self, itens):
        """Gera e salva um novo or amento"""
        if not itens:
            print("\nNenhum item selecionado para o or amento.")
            return

        orcamento = Orcamento(itens)
        self.historico.adicionar_orcamento(orcamento)

        print("\n" + "="*50)
        print("ORAMENTO GERADO COM SUCESSO!")
        print("="*50)
        print(f"Data: {orcamento.data.strftime('%d/%m/%Y %H:%M')}")
        print("\nITENS SELECIONADOS:")
        for item, qtd in orcamento.itens:
            print(f"  {qtd}x {item.nome} - R${item.preco:.2f} cada")
        print("\n" + "="*50)
        print(f"TOTAL: R${orcamento.total:.2f}")
        print(f"SALDO RESTANTE: R${self.orcamento_cliente - orcamento.to
        print("="*50)
```

```
        self.forma_pagamento(orcamento.total)

    def consultar_historico(self):
        """Mostra todos os or amentos no hist rico"""
        historico = self.historico.listar_orcamentos()

        if not historico:
            print("\nNenhum or amento no hist rico")
            return

        print("\nHIST RICO DE ORAMENTOS:")
        for i, orcamento in enumerate(historico):
            print(f"\n[{i}] {orcamento.data.strftime('%d/%m/%Y %H:%M')}")
            for item, qtd in orcamento.itens:
                print(f"  {qtd}x {item.nome}")
            print(f"  TOTAL: R${orcamento.total:.2f}")

    def comparar_orcamentos(self):
        """Compara dois or amentos do hist rico"""
        historico = self.historico.listar_orcamentos()

        if len(historico) < 2:
            print("\n  necess rio ter pelo menos 2 or amentos para com
            return

        self.consultar_historico()

        try:
            idx1 = int(input("\n ndice do primeiro or amento: "))
            idx2 = int(input(" ndice  do segundo or amento: "))
            self.historico.comparar_orcamentos(idx1, idx2)
        except ValueError:
            print("Digite n meros v lidos")

    def forma_pagamento(self, total):
        """Pergunta a forma de pagamento desejada"""
        print("\nBem-vindo ao sistema de pagamento!")
        print("============================")
        print("------[1] -- CR DITO -------")
        print("------[2] -- D BITO --------")
        print("------[3] -- PIX -----------")
        print("------[4] -- COMBINAR ------")
        print("============================\n")
        while True:
                forma = input("\nQual a forma de pagamento? (1-4): ").str
                if forma == "1":
                 self.calcular_parcelas(total)
                 break
                elif forma == "2":
```

```
                print(f"Pagamento no d bito: R${total:.2f}")
                break
            elif forma == "3":
                print(f"Pagamento via PIX: R${total:.2f}")
                break
            elif forma == "4":
                self.dividir_pagamento(total)
                break
            else:
                print("Forma de pagamento n o reconhecida.")

    def calcular_parcelas(self, total):
        """Calcula parcelamento do valor total"""
        while True:
            parcelas = input("Deseja parcelar em quantas vezes? (at  3x)
            if parcelas in ["1", "2", "3"]:
                parcelas = int(parcelas)
                valor_parcela = total / parcelas
                print(f"\nPagamento no cr dito: {parcelas}x de R${valor
                break
            else:
                print("N mero de parcelas inv lido. M ximo permitido

    def dividir_pagamento(self, total):
        """Permite dividir o pagamento entre m ltiplas formas"""
        print("\nVoc  escolheu combinar formas de pagamento (Ex: cr dit
        while True:
            try:
                valor_credito = float(input("Quanto deseja pagar no cr d
                valor_debito = float(input("Quanto deseja pagar no d bit
                valor_pix = float(input("Quanto deseja pagar via PIX? R$

                soma = valor_credito + valor_debito + valor_pix

                if abs(soma - total) <= 0.05:
                    if valor_credito > 0:
                        self.calcular_parcelas(valor_credito)
                    if valor_debito > 0:
                        print(f"Pagamento no d bito: R${valor_debito:.2
                    if valor_pix > 0:
                        print(f"Pagamento via PIX: R${valor_pix:.2f}")
                    break
                else:
                    print(f"A soma dos valores ({soma:.2f}) n o bate com
            except ValueError:
                print("Entrada inv lida. Use n meros.")

    def executar(self):
        """Executa o fluxo principal do sistema"""
        self.apresentacao_da_empresa()
```

```
        while True:
            self.mostrar_menu_principal()
            opcao = input("\nEscolha uma op  o num rica: ")

            if opcao == '1':
                self.mostrar_todos_itens()
                input("\nPressione Enter para continuar...")

            elif opcao == '2':
                self.selecionar_categorias()
                self.mostrar_itens_categorias_selecionadas()
                input("\nPressione Enter para continuar...")

            elif opcao == '3':
                self.definir_orcamento_cliente()
                input("\nPressione Enter para continuar...")

            elif opcao == '4':
                selecao = self.selecionar_quantidades()
                if selecao:
                    self.gerar_orcamento(selecao)
                input("\nPressione Enter para continuar...")

            elif opcao == "5":
                self.consultar_historico()
                input("\nPressione Enter para continuar...")

            elif opcao == "6":
                self.comparar_orcamentos()
                input("\nPressione Enter para continuar...")

            elif opcao == "7":
                print("\nSaindo do sistema...")
                break

            else:
                print("Op  o inv lida")
                input("\nPressione Enter para continuar...")


if __name__ == "__main__":
    sistema = SistemaSeguranca()
    sistema.executar()
```

5 Considerações finais

Diante do programa desenvolvido durante do semestre, pode-se inferir que o Python é uma linguagem de programação fundamental para automação de procedimentos na Engenharia Civil, sobretudo em relação à área de gerenciamento de obras. A interface do programa contribui para organização de tarefas, alinhamento de prazos e serviços. Isto por sua vez, agrega benefícios a todos os envolvidos durante as etapas de execução de uma obra, seja ao engenheiro responsável, por ter as atividades devidamente enumeradas e com recebimento de alerta, seja para o cliente que consegue obter como resultado uma obra realizada com qualidade, sem gastos adicionais e tempo extrapolado. Além disso, a atribuição das atividades da construção possibilita melhor divisão das tarefas a serem efetuadas pelas equipes no canteiro de obras.

Portanto, é notório que para o engenheiro(a), especializado em Gestão de Obras, possuir habilidade com o Python pode ser fundamental para que este se torne um profissional em destaque no mercado de trabalho, visto que a inserção da tecnologia dentro de um segmento que ainda não se faz amplamente presente, neste setor da Construção Civil, o qual não se configura com muita mecanização.