Front-end

Descrição e códigos refentes ao front-end.

- Classes Python

Classes Python

Introdução

O software de controle patrimonial tem como objetivo gerenciar, organizar e monitorar o patrimônio de uma empresa ou instituição. Ele permite a gestão eficiente de bens móveis e imóveis, facilitando o acompanhamento de seu esçtado, localização, movimentação e manutenção. A documentação a seguir apresenta os principais componentes do sistema, com exemplos de código para ilustrar a implementação de suas funcionalidades.

Estrutura do Sistema

O sistema de controle patrimonial pode ser estruturado em várias camadas, como a camada de dados, de lógica de negócio e a camada de apresentação. A seguir, descrevemos os principais módulos do sistema.

---

1. Cadastro de Bens

> *A primeira funcionalidade essencial do software é o cadastro dos bens patrimoniais. Cada bem é associado a informações como nome, código de identificação, categoria, data de aquisição e valor. Abaixo está um exemplo de código que mostra como o cadastro de bens pode ser estruturado:*

```
class BemPatrimonial:
  #asdasdasd
    def __init__(self, codigo, nome, categoria, data_aquisicao, valor):
        self.codigo = codigo
        self.nome = nome
        self.categoria = categoria
        self.data_aquisicao = data_aquisicao
        self.valor = valor


    def exibir_dados(self):
```

```
        return f'Código: {self.codigo}, Nome: {self.nome}, Categoria: {self.categoria}, Data de Aquisição: {self.data_aquisicao}, Valor: {self.valor}'

# Exemplo de cadastro de um bem
bem1 = BemPatrimonial("12345", "Computador", "Equipamento de Informática", "2022-01-10", 5000.00)
print(bem1.exibir_dados())

  # Testeeee
```

Saída Esperada:

> **Código: 12345, Nome: Computador, Categoria: Equipamento de Informática, Data de Aquisição: 2022-01-10, Valor: 5000.0**

---

2. Movimentação de Bens

> *Outro aspecto fundamental do controle patrimonial é o registro das movimentações dos bens, como transferências entre departamentos ou unidades da empresa. O código a seguir exemplifica como isso pode ser feito:*

```
class Movimentacao:
    def __init__(self, bem, data_movimentacao, origem, destino):
        self.bem = bem
        self.data_movimentacao = data_movimentacao
        self.origem = origem
        self.destino = destino

    def registrar_movimentacao(self):
        return f'Movimento do bem {self.bem.nome} de {self.origem} para {self.destino} em {self.data_movimentacao}.'

# Exemplo de movimentação
mov1 = Movimentacao(bem1, "2025-03-10", "Departamento de TI", "Departamento de Marketing")
print(mov1.registrar_movimentacao())
```

Saída Esperada:

>

> **Movimento do bem Computador de Departamento de TI para Departamento de Marketing em 2025-03-10.**

---

3. Relatório de Bens

A geração de relatórios é uma funcionalidade crucial para acompanhar o estado do patrimônio. Este módulo pode permitir a criação de relatórios sobre bens ativos, bens em manutenção, bens fora de uso, entre outros. Um exemplo simples de relatório seria:

```
class RelatorioPatrimonial:
    def __init__(self, bens):
        self.bens = bens

    def gerar_relatorio(self):
        relatorio = "Relatório de Bens Patrimoniais:\n"
        for bem in self.bens:
            relatorio += bem.exibir_dados() + "\n"
        return relatorio

# Exemplo de geração de relatório
bens = [bem1]
relatorio = RelatorioPatrimonial(bens)
print(relatorio.gerar_relatorio())
```

Saída Esperada:

> **Relatório de Bens Patrimoniais:**
> **Código: 12345, Nome: Computador, Categoria: Equipamento de Informática, Data de Aquisição: 2022-01-10, Valor: 5000.0**

---

4. Manutenção de Bens

Além de controlar a movimentação dos bens, é importante gerenciar as manutenções realizadas, registrando as datas e tipos de manutenção. O código abaixo exemplifica como implementar esse registro:

```
class Manutencao:
    def __init__(self, bem, tipo_manutencao, data_manutencao):
        self.bem = bem
        self.tipo_manutencao = tipo_manutencao
        self.data_manutencao = data_manutencao

    def registrar_manutencao(self):
        return f'Manutenção do bem {self.bem.nome} do tipo {self.tipo_manutencao} realizada em
{self.data_manutencao}.'

# Exemplo de manutenção
manutencao1 = Manutencao(bem1, "Preventiva", "2025-02-28")
print(manutencao1.registrar_manutencao())
```

Saída Esperada:

> **Manutenção do bem Computador do tipo Preventiva realizada em 2025-02-28.**

Conclusão

Este é um exemplo básico de como o software de controle patrimonial pode ser estruturado. A implementação completa envolve mais funcionalidades, como o controle de acesso dos usuários, a integração com outros sistemas e a geração de relatórios detalhados. As funcionalidades apresentadas garantem um controle eficiente dos bens patrimoniais, proporcionando à empresa um gerenciamento preciso e organizado de seus ativos.

chatbox.png
Image not found or type unknown

e