Desenvolvimento de Projeto Aplicado em Python com Tkinter e MongoDB

Professor: Pedro Capelari

December 20, 2024

Introdução

Este documento aborda o desenvolvimento de um projeto aplicado em Python que combina os seguintes elementos:

- Interface gráfica desenvolvida com **Tkinter** e **TTK**.
- Estrutura cliente-servidor com conexão ao banco de dados **MongoDB**.
- Princípios da **teoria de produção de software**.

O objetivo é fornecer aos alunos uma visão prática de como desenvolver uma aplicação completa, aplicando boas práticas de desenvolvimento de software.

1 Teoria de Produção de Software

1.1 Ciclo de Vida de Desenvolvimento de Software

O ciclo de vida de desenvolvimento de software descreve as etapas envolvidas na criação de um sistema de software de alta qualidade. Os principais modelos incluem:

- **Modelo em Cascata (Waterfall)**: Abordagem sequencial em fases (Requisitos, Projeto, Implementação, Teste, Manutenção).
- **Desenvolvimento Incremental e Iterativo**: Construção em ciclos incrementais, permitindo feedback contínuo.
- **Metodologias Ágeis**: Desenvolvimento adaptativo, focado em entregas rápidas e colaboração com o cliente (ex.: Scrum, Kanban).

1.2 Princípios de Engenharia de Software

- **Modularidade**: Dividir o sistema em módulos menores para facilitar o desenvolvimento e a manutenção.
- **Reusabilidade**: Escrever código que possa ser reutilizado em diferentes partes do projeto.
- **Manutenibilidade**: Facilitar a correção de erros e a adição de novas funcionalidades.
- **Testabilidade**: Garantir que o código possa ser facilmente testado.

2 Estrutura do Projeto

O projeto consiste em um aplicativo de gerenciamento de estoque que permite adicionar, remover e listar itens. Ele é estruturado da seguinte forma:

```
gerenciamento_estoque/
 main.py
 gui.py
 database.py
 estoque.py
```

2.1 Descrição dos Arquivos

- **main.py**: Ponto de entrada da aplicação.
- **gui.py**: Interface gráfica construída com Tkinter e TTK.
- **database.py**: Conexão com o banco de dados MongoDB.
- **estoque.py**: Lógica de negócios para gerenciar o estoque.

3 Código do Projeto

3.1 Módulo de Estoque (`estoque.py`)

Este módulo contém a classe responsável pelas operações de gerenciamento de estoque.

```
class Estoque:
    def __init__(self, db):
        self.db = db
        self.collection = db["estoque"]

    def adicionar_item(self, nome, quantidade):
        item = self.collection.find_one({"nome": nome})
        if item:
            nova_quantidade = item["quantidade"] + quantidade
            self.collection.update_one({"nome": nome}, {"$set": {"
    quantidade": nova_quantidade}})
        else:
            self.collection.insert_one({"nome": nome, "quantidade":
    quantidade})

    def remover_item(self, nome, quantidade):
        item = self.collection.find_one({"nome": nome})
        if item:
            nova_quantidade = item["quantidade"] - quantidade
            if nova_quantidade <= 0:
                self.collection.delete_one({"nome": nome})
            else:
                self.collection.update_one({"nome": nome}, {"$set": {"
    quantidade": nova_quantidade}})
        else:
            print(f"Item '{nome}' n o encontrado no estoque.")

    def listar_itens(self):
        return list(self.collection.find())
```

Listing 1: Código do módulo estoque

3.2 Conexão com MongoDB (`database.py`)

Este módulo estabelece a conexão com o MongoDB.

```
import pymongo

def conectar_mongo():
    try:
        cliente = pymongo.MongoClient("mongodb://localhost:27017/")
        db = cliente["estoque_db"]
        return db
    except Exception as e:
        print(f"Erro ao conectar ao MongoDB: {e}")
        return None
```

Listing 2: Código do módulo de conexão com MongoDB

3.3 Interface Gráfica (gui.py)

A interface gráfica permite interagir com o estoque através de botões e entradas.

```
import tkinter as tk
from tkinter import ttk
from estoque import Estoque
from database import conectar_mongo

class EstoqueApp:
    def __init__(self, root):
        self.db = conectar_mongo()
        self.estoque = Estoque(self.db)

        self.root = root
        self.root.title("Gerenciamento de Estoque")

        # Widgets
        self.nome_label = ttk.Label(root, text="Nome do Item:")
        self.nome_label.grid(row=0, column=0, padx=5, pady=5)
        self.nome_entry = ttk.Entry(root)
        self.nome_entry.grid(row=0, column=1, padx=5, pady=5)

        self.qtd_label = ttk.Label(root, text="Quantidade:")
        self.qtd_label.grid(row=1, column=0, padx=5, pady=5)
        self.qtd_entry = ttk.Entry(root)
        self.qtd_entry.grid(row=1, column=1, padx=5, pady=5)

        self.adicionar_btn = ttk.Button(root, text="Adicionar Item",
    command=self.adicionar_item)
        self.adicionar_btn.grid(row=2, column=0, padx=5, pady=5)

        self.remover_btn = ttk.Button(root, text="Remover Item",
    command=self.remover_item)
        self.remover_btn.grid(row=2, column=1, padx=5, pady=5)

        self.listar_btn = ttk.Button(root, text="Listar Itens", command
    =self.listar_itens)
        self.listar_btn.grid(row=3, column=0, columnspan=2, padx=5,
    pady=5)

        self.resultado_text = tk.Text(root, height=10, width=40)
        self.resultado_text.grid(row=4, column=0, columnspan=2, padx=5,
     pady=5)

    def adicionar_item(self):
        nome = self.nome_entry.get()
        quantidade = int(self.qtd_entry.get())
        self.estoque.adicionar_item(nome, quantidade)
        self.resultado_text.insert(tk.END, f"Item '{nome}' adicionado
    com sucesso.\n")

    def remover_item(self):
        nome = self.nome_entry.get()
        quantidade = int(self.qtd_entry.get())
        self.estoque.remover_item(nome, quantidade)
        self.resultado_text.insert(tk.END, f"Item '{nome}' removido com
     sucesso.\n")

```

```
    def listar_itens(self):
        self.resultado_text.delete(1.0, tk.END)
        itens = self.estoque.listar_itens()
        for item in itens:
            self.resultado_text.insert(tk.END, f"{item['nome']}: {item
    ['quantidade']}\n")

if __name__ == "__main__":
    root = tk.Tk()
    app = EstoqueApp(root)
    root.mainloop()
```

Listing 3: Código da interface gráfica com Tkinter

4 Conclusão

Este projeto integra conceitos de interface gráfica com Tkinter e uma estrutura cliente-servidor usando MongoDB. Ele também segue princípios fundamentais de produção de software, como modularidade, reutilização de código e boas práticas de desenvolvimento.