

Arquivo: merge_anexos.py

```
from PyQt6.QtWidgets import (
    QDialog, QVBoxLayout, QLabel, QTreeView, QTableWidgetItem, QCheckBox,
    QPushButton, QHBoxLayout, QMessageBox, QFileIconProvider
)
from PyQt6.QtCore import Qt, QDateTime, QFileInfo
from PyQt6.QtGui import QIcon, QStandardItemModel, QStandardItem
from PyPDF2 import PdfMerger
from pathlib import Path
import os

class MergePDFDialog(QDialog):
    def __init__(self, pdf_files, parent=None):
        super().__init__(parent)
        self.pdf_files = pdf_files
        self.selected_files = [] # Lista para armazenar os PDFs selecionados na ordem
        self.setWindowTitle("Merge PDFs")
        self.setup_ui()

    def setup_ui(self):
        layout = QVBoxLayout(self)

        label = QLabel("Arraste e solte os arquivos para definir a ordem de mesclagem:")
        layout.addWidget(label)

        # Criar o QTreeView para exibir os arquivos e permitir reordenação
        self.tree_view = QTreeView()
        self.model = QStandardItemModel(0, 1)
        self.model.setHeaderData(0, Qt.Orientation.Horizontal, "Arquivos PDF")
        self.tree_view.setModel(self.model)
        self.tree_view.setDragDropMode(QTreeView.DragDropMode.InternalMove) # Ativar arrastar e
        soltar interno
        self.tree_view.setSelectionMode(QTreeView.SelectionMode.ExtendedSelection)
        layout.addWidget(self.tree_view)

        self.populate_tree()

        # Botões para realizar merge e cancelar
        button_layout = QHBoxLayout()
        self.merge_button = QPushButton("Realizar Merge")
        self.merge_button.clicked.connect(self.merge_pdfs)
        button_layout.addWidget(self.merge_button)

        self.cancel_button = QPushButton("Cancelar")
        self.cancel_button.clicked.connect(self.reject)
        button_layout.addWidget(self.cancel_button)

        layout.addLayout(button_layout)

    def populate_tree(self):
        """Preenche o QTreeView com os arquivos PDF."""

```

```

for pdf_file in self.pdf_files:
    item = QStandardItem(os.path.basename(pdf_file)) # Exibir apenas o nome do arquivo
    item.setData(pdf_file, Qt.ItemDataRole.UserRole) # Armazena o caminho completo do
arquivo
    item.setDragEnabled(True) # Permitir que o item seja arrastado
    self.model.appendRow(item)

def merge_pdfs(self):
    """Função chamada para mesclar os arquivos PDF na ordem especificada."""
    selected_items = [self.model.item(i) for i in range(self.model.rowCount())]
    self.selected_files = [item.data(Qt.ItemDataRole.UserRole) for item in selected_items]

    if not self.selected_files:
        QMessageBox.warning(self, "Erro", "Selecione pelo menos um arquivo PDF para mesclar.")
        return

    try:
        # Mesclar os PDFs na ordem especificada pelo usuário
        merged_pdf_path = self.merge_pdf_files(self.selected_files)
        QMessageBox.information(self, "Sucesso", f"PDFs mesclados com sucesso em:
{merged_pdf_path}")
        self.accept()
    except Exception as e:
        QMessageBox.critical(self, "Erro", f"Erro ao mesclar PDFs: {str(e)}")

def merge_pdf_files(self, pdf_files):
    """Função que realiza a mesclagem dos PDFs."""
    from PyPDF2 import PdfMerger
    merger = PdfMerger()
    for pdf in pdf_files:
        merger.append(str(pdf))
    merged_pdf_path = str(Path(pdf_files[0]).parent / "merged_output.pdf")
    merger.write(merged_pdf_path)
    merger.close()
    return merged_pdf_path

```