Arquivo: view.py

```python
from PyQt6.QtWidgets import *
from PyQt6.QtGui import *
from PyQt6.QtCore import *
from modules.utils.add_button import add_button, add_button_result
from modules.atas_api.widgets.importar_tr import TermoReferenciaWidget
from modules.atas_api.widgets.instrucoes import InstructionWidget
from modules.atas_api.widgets.sicaf import RegistroSICAFDialog
from modules.atas_api.widgets.consultar_api import ConsultarAPI
from modules.atas_api.widgets.gerar_atas import GerarAtaWidget
from pathlib import Path
from paths import PDF_DIR
from .database import DatabaseATASAPIManager

class GerarAtasApiView(QMainWindow):
    instructionSignal = pyqtSignal()
    trSignal = pyqtSignal()
    homologSignal = pyqtSignal()
    sicafSignal = pyqtSignal()
    apiSignal = pyqtSignal()
    atasSignal = pyqtSignal()
    pdf_dir_changed = pyqtSignal(Path)

    def __init__(self, icons, model, database_path, parent=None):
        super().__init__(parent)
        self.icons = icons
        self.model = model
        self.database_ata_manager = DatabaseATASAPIManager(database_path)
        self.pdf_dir = PDF_DIR
        self.setup_ui()

        self.pdf_dir_changed.connect(self.on_pdf_dir_changed)

    def setup_ui(self):
        # Configuração inicial do layout principal
        self.main_widget = QWidget(self)
        self.setCentralWidget(self.main_widget)
        self.main_layout = QVBoxLayout(self.main_widget)

        # Título da interface
        label_ata_api = QLabel("Atas de Registro de Preços (API)", self)
        label_ata_api.setStyleSheet("font-size: 26px; font-weight: bold;")
        self.main_layout.addWidget(label_ata_api)

        # Adiciona menu e conteúdo
        menu_widget = self.create_menu_layout()

        # Área de conteúdo com QStackedWidget
        self.content_area = QStackedWidget()
        self.main_layout.addWidget(menu_widget)
        self.main_layout.addWidget(self.content_area)
        # Inicializa os widgets de conteúdo para cada seção
        self.init_content_widgets()

        # Define o conteúdo inicial
        self.show_initial_content()

    def init_content_widgets(self):
        # Adiciona o widget de instruções
        self.instrucoes_widget = InstructionWidget(self)
        self.content_area.addWidget(self.instrucoes_widget)

        self.tr_widget = TermoReferenciaWidget(self, self.icons)
        self.content_area.addWidget(self.tr_widget)

        self.sicaf_widget = RegistroSICAFDialog(
            pdf_dir=self.pdf_dir,
            model=self.model,
            icons=self.icons,
                      database_ata_manager=self.database_ata_manager,    #  Passa  a  instância  do
DatabaseATASManager
            main_window=self
        )
        self.content_area.addWidget(self.sicaf_widget)

        self.api_widget = ConsultarAPI(
            icons=self.icons,
            database_ata_manager=self.database_ata_manager,
            main_window=self
            )
        self.content_area.addWidget(self.api_widget)

        self.atas_widget = GerarAtaWidget(
            icons=self.icons,
            database_ata_manager=self.database_ata_manager,
            main_window=self)
        self.content_area.addWidget(self.atas_widget)

    def show_initial_content(self):
        # Define o widget inicial como ativo
        self.content_area.setCurrentWidget(self.instrucoes_widget)

    def create_menu_layout(self):
        menu_widget = QWidget()
        menu_layout = QHBoxLayout(menu_widget)

        # Garante que um layout válido seja adicionado
        button_layout = self.create_button_layout()
        if button_layout is not None:
            menu_layout.addLayout(button_layout)

        return menu_widget
    def create_button_layout(self):
        button_layout = QHBoxLayout()

        # Adiciona o botão para instruções
             add_button("Instruções", "info", self.instructionSignal, button_layout, self.icons,
"Exibir Instruções")

        # Outros botões já existentes
           add_button("Termo de Referência", "layers", self.trSignal, button_layout, self.icons,
"Acessar Termo de Referência")
            add_button("Consultar", "api", self.apiSignal, button_layout, self.icons, "Consultar
informações da API")
              add_button_result("SICAF", "layers", self.sicafSignal, button_layout, self.icons,
"Atualizar SICAF", lambda: self.sicaf_widget.carregar_tabelas_result() if hasattr(self,
'sicaf_widget') else None)
         add_button_result("Gerar Ata", "star", self.atasSignal, button_layout, self.icons, "Gerar
nova ata", lambda: self.atas_widget.carregar_tabelas_result() if hasattr(self, 'sicaf_widget')
else None)

        return button_layout

    def configurar_visualizacao_tabela_tr(self, table_view):
        # Verifica se o modelo foi configurado antes de prosseguir
        if table_view.model() is None:
            print("O modelo de dados não foi configurado para table_view.")
            return  # Sai da função se o modelo não estiver configurado

        # Define colunas visíveis
        visible_columns = [1, 2, 3, 4]
        for col in range(table_view.model().columnCount()):
            if col not in visible_columns:
                table_view.hideColumn(col)
            else:
                header = table_view.model().headerData(col, Qt.Orientation.Horizontal)
                table_view.model().setHeaderData(col, Qt.Orientation.Horizontal, header)

        # Configuração de redimensionamento das colunas
        table_view.setColumnWidth(1, 40)
        table_view.setColumnWidth(2, 70)
        table_view.setColumnWidth(3, 200)
        table_view.horizontalHeader().setStretchLastSection(True)

    def on_pdf_dir_changed(self, new_pdf_dir):
        # Lida com a mudança do diretório PDF, se necessário
        print(f"Novo diretório PDF definido: {new_pdf_dir}")
```