Arquivo: comprasnet_api.py

```
from PyQt6.QtWidgets import *
from PyQt6.QtGui import *
from PyQt6.QtCore import *
from modules.contratos.delegate import Dialogs
from datetime import datetime
import pandas as pd
from contextlib import contextmanager
from modules.contratos.database_manager.db_manager import DatabaseManager
import os, sqlite3, json, requests
from PyQt6.QtCore import QThread, pyqtSignal
from PyQt6.QtWidgets import (
    QDialog, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton,
    QProgressBar, QFrame, QMessageBox
)
from paths import JSON_COMPRASNET_CONTRATOS

class ConsultaAPIWindow(QDialog):
    def __init__(self, icons, model, view, parent=None):
        super().__init__(parent)
        self.icons = icons
        self.model = model
        self.view = view
        self.setWindowTitle("Consulta a API do Comprasnet")
        self.setWindowIcon(self.icons.get("api", None))
        self.setFixedSize(350, 150)
        self.setup_ui()

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

        # Frame com estilo personalizado
        frame_container = QFrame(self)
        frame_container.setObjectName("FrameContainer")
        frame_container.setStyleSheet("""
            #FrameContainer {
                border: 1px solid black;
                border-radius: 10px;
                background: #f1eeee;
            }
        """)
        main_layout.addWidget(frame_container)

        # Layout interno do frame
        frame_layout = QVBoxLayout(frame_container)
        frame_layout.setContentsMargins(10, 10, 10, 10)

        # Linha com label e campo para inserir o sequencial
        hlayout = QHBoxLayout()
        hlayout.addWidget(QLabel("Sequencial:", frame_container))
        self.sequencial_input = QLineEdit(frame_container)
```

```
    hlayout.addWidget(self.sequencial_input)
    frame_layout.addLayout(hlayout)

    # Botão para consulta
    self.btn_consulta = QPushButton("Consultar", frame_container)
    frame_layout.addWidget(self.btn_consulta)

    # Barra de progresso
    self.progress_bar = QProgressBar(frame_container)
    self.progress_bar.setRange(0, 100)
    self.progress_bar.setValue(0)
    frame_layout.addWidget(self.progress_bar)

    # Conecta o botão com a função de consulta
    self.btn_consulta.clicked.connect(self.realizar_consulta)

def realizar_consulta(self):
    unidade_codigo = self.sequencial_input.text().strip()
    if not unidade_codigo:
        QMessageBox.warning(self, "Aviso", "Insira um código de unidade válido.")
        return

    self.progress_bar.setValue(10)
    self.thread = RequestThread(unidade_codigo)
    self.thread.data_received.connect(self.on_data_received)
    self.thread.error_occurred.connect(self.on_error_occurred)
    self.thread.save_json.connect(self.on_save_json)
    self.thread.start()

def on_data_received(self, data):
    self.progress_bar.setValue(70)
    self.processar_dados_para_tabela(data)
    QMessageBox.information(self, "Sucesso", "Dados recebidos com sucesso.")

def on_error_occurred(self, error_message):
    self.progress_bar.setValue(0)
    QMessageBox.critical(self, "Erro", error_message)

def on_save_json(self, data, unidade_codigo):
    # Cria o diretório se não existir
    JSON_COMPRASNET_CONTRATOS.mkdir(parents=True, exist_ok=True)
    file_path = JSON_COMPRASNET_CONTRATOS / f"contratos_{unidade_codigo}.json"
    try:
        with open(file_path, 'w', encoding='utf-8') as json_file:
            json.dump(data, json_file, ensure_ascii=False, indent=4)
        self.progress_bar.setValue(100)
        QMessageBox.information(self, "Arquivo Salvo", f"Arquivo JSON salvo em: {file_path}")
    except Exception as e:
        QMessageBox.critical(self, "Erro", f"Erro ao salvar o arquivo JSON: {e}")

def processar_dados_para_tabela(self, data):
    contratos_list = []
    for contrato in data["data"]:
```

```
            prorrogavel = "Sim" if contrato.get("prorrogavel") == "Sim" else "Não"
            custeio = "Sim" if contrato.get("custeio") == "Sim" else "Não"
                situacao = contrato.get("situacao") if contrato.get("situacao") is not None else
"Seção de Contratos"
            vigencia_final = contrato.get("vigencia_fim")
            if not vigencia_final or pd.isna(vigencia_final):
                vigencia_final = "2016-01-01"
            contrato_info = {
                "situacao": situacao,
                "id": contrato.get("id"),
                "numero_contrato": contrato.get("numero"),
                "codigo_uasg": contrato["contratante"]["orgao"]["unidade_gestora"].get("codigo"),
                                                                        "orgao_contratante_resumido":
contrato["contratante"]["orgao"]["unidade_gestora"].get("nome_resumido"),
                                                                    "orgao_contratante":
contrato["contratante"]["orgao"]["unidade_gestora"].get("nome"),
                "cnpj_cpf_idgener": contrato["fornecedor"].get("cnpj_cpf_idgener"),
                "nome_fornecedor": contrato["fornecedor"].get("nome"),
                "tipo": contrato.get("tipo"),
                "subtipo": contrato.get("subtipo"),
                "prorrogavel": prorrogavel,
                "custeio": custeio,
                "situacao": contrato.get("situacao"),
                "categoria": contrato.get("categoria"),
                "processo_nup": contrato.get("processo"),
                "objeto": contrato.get("objeto"),
                "amparo_legal": contrato.get("amparo_legal"),
                "modalidade": contrato.get("modalidade"),
                "licitacao_numero": contrato.get("licitacao_numero"),
                "data_assinatura": contrato.get("data_assinatura"),
                "data_publicacao": contrato.get("data_publicacao"),
                "vigencia_inicial": contrato.get("vigencia_inicio"),
                "vigencia_final": vigencia_final,
                "valor_global": contrato.get("valor_global")
            }
            contratos_list.append(contrato_info)

        # Cria DataFrame e assegura a presença das colunas necessárias
        df = pd.DataFrame(contratos_list)
        required_columns = [
            "status", "id", "numero_contrato", "codigo_uasg", "orgao_contratante_resumido",
            "orgao_contratante", "cnpj_cpf_idgener", "nome_fornecedor", "tipo", "subtipo",
            "prorrogavel", "custeio", "situacao", "categoria", "processo_nup",
            "objeto", "amparo_legal", "modalidade", "licitacao_numero", "data_assinatura",
            "data_publicacao", "vigencia_inicial", "vigencia_final", "valor_global", "situacao"
        ]
        for col in required_columns:
            if col not in df.columns:
                df[col] = None
        df = df[required_columns]

        # Converte 'vigencia_final' para datetime para ordenação e reverte para string
                 df['vigencia_final']  =  pd.to_datetime(df['vigencia_final'],  format='%Y-%m-%d',
```

```
errors='coerce')
        df = df.sort_values(by='vigencia_final', ascending=False)
        df['vigencia_final'] = df['vigencia_final'].dt.strftime('%Y-%m-%d')

        # Insere ou atualiza os dados no modelo
        for _, row in df.iterrows():
            self.model.insert_or_update_data(row.to_dict())
        self.view.refresh_model()


class RequestThread(QThread):
    data_received = pyqtSignal(object)
    error_occurred = pyqtSignal(str)
    save_json = pyqtSignal(object, str)

    def __init__(self, unidade_codigo):
        super().__init__()
        self.unidade_codigo = unidade_codigo

    def run(self):
        base_url = "https://contratos.comprasnet.gov.br"
        endpoint = f"/api/contrato/ug/{self.unidade_codigo}"
        url = base_url + endpoint
        headers = {
            "User-Agent": ("Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
                           "AppleWebKit/537.36 (KHTML, like Gecko) "
                           "Chrome/115.0 Safari/537.36"),
            "Accept": "application/json, text/javascript, */*; q=0.01",
            "Referer": "https://contratos.comprasnet.gov.br",
            "Connection": "close"
        }
        try:
            session = requests.Session()
            adapter = requests.adapters.HTTPAdapter(max_retries=3)
            session.mount("https://", adapter)
            session.mount("http://", adapter)
            response = session.get(url, headers=headers, timeout=30)
            print("Raw response content:", response.text)
            response.raise_for_status()
            data = response.json()
            if isinstance(data, list):
                data = {"data": data}
            self.data_received.emit(data)
            self.save_json.emit(data, self.unidade_codigo)
        except requests.exceptions.HTTPError as http_err:
            error_message = f"HTTP error occurred: {http_err}"
            print(error_message)
            self.error_occurred.emit(error_message)
        except Exception as err:
            error_message = f"Other error occurred: {err}"
            print(error_message)
            self.error_occurred.emit(error_message)
```