Arquivo: config_path.py

```
from PyQt6.QtWidgets import *
from PyQt6.QtCore import *
from .base_path import JSON_DIR, DATABASE_DIR, CONFIG_FILE
from pathlib import Path
import json

def load_config_path_id():
    if not Path(CONFIG_FILE).exists():
        return {}
    with open(CONFIG_FILE, 'r') as file:
        return json.load(file)

def load_config(key, default_value):
    try:
        with open(CONFIG_FILE, 'r') as f:
            config = json.load(f)
            return config.get(key, default_value)
    except (FileNotFoundError, json.JSONDecodeError):
        return default_value

def save_config(key, value):
    config = {}
    try:
        with open(CONFIG_FILE, 'r') as f:
            config = json.load(f)
    except (FileNotFoundError, json.JSONDecodeError):
        pass
    config[key] = value
    with open(CONFIG_FILE, 'w') as f:
        json.dump(config, f)

def update_dir(title, key, default_value, parent=None):
    new_dir = QFileDialog.getExistingDirectory(parent, title)
    if new_dir:
        save_config(key, new_dir)
        return Path(new_dir)
    return default_value

PRE_DEFINICOES_JSON = JSON_DIR / "pre_definicioes.json"
AGENTES_RESPONSAVEIS_FILE = JSON_DIR / "agentes_responsaveis.json"
ORGANIZACOES_FILE = JSON_DIR / "organizacoes.json"

PDF_DIR = Path(load_config("PDF_DIR", DATABASE_DIR / "pdf"))

class ConfigManager(QObject):
    config_updated = pyqtSignal(str, Path)  # sinal emitido quando uma configuração é atualizada

    def __init__(self, config_file):
        super().__init__()
        self.config_file = config_file
```

```
        self.config = self.load_config()

    def load_config(self):
        try:
            with open(self.config_file, 'r') as f:
                return json.load(f)
        except (FileNotFoundError, json.JSONDecodeError):
            return {}

    def save_config(self, key, value):
        self.config[key] = value
        with open(self.config_file, 'w') as f:
            json.dump(self.config, f)
        self.config_updated.emit(key, Path(value))

    def update_config(self, key, value):
        # Aqui garantimos que ambos os parâmetros sejam passados corretamente para save_config
        self.save_config(key, value)
        self.config_updated.emit(key, Path(value))

    def get_config(self, key, default_value):
        return self.config.get(key, default_value)

class EventManager(QObject):
    controle_dados_dir_updated = pyqtSignal(Path)
    pdf_dir_updated = pyqtSignal(Path)
    sicaf_dir_updated = pyqtSignal(Path)
    relatorio_path_updated = pyqtSignal(Path)
    controle_dir_updated =  pyqtSignal(Path)

    def __init__(self):
        super().__init__()

    def update_database_dir(self, new_file):
        global CONTROLE_DADOS
        CONTROLE_DADOS = new_file
        save_config("CONTROLE_DADOS", str(new_file))
        self.controle_dir_updated.emit(new_file)

    def update_pdf_dir(self, new_dir):
        print(f"Emitindo sinal de atualização de PDF_DIR: {new_dir}")
        self.pdf_dir_updated.emit(new_dir)

    def update_controle_dados_dir(self, new_file):
        global CONTROLE_DADOS
        if new_file != CONTROLE_DADOS:
            CONTROLE_DADOS = new_file
            save_config("CONTROLE_DADOS", str(new_file))
            self.controle_dados_dir_updated.emit(new_file)
            print(f"CONTROLE_DADOS atualizado para {new_file}")

    def update_controle_dados_pncp_dir(self, new_file):
        global CONTROLE_DADOS_PNCP
```

```
        if new_file != CONTROLE_DADOS_PNCP:
            CONTROLE_DADOS_PNCP = new_file
            save_config("CONTROLE_DADOS_PNCP", str(new_file))
            self.controle_dados_dir_updated.emit(new_file)
            print(f"CONTROLE_DADOS_PNCP atualizado para {new_file}")

    def update_contratacoes_diretas_database_dir(self, new_file):
        global CONTROLE_CONTRATACAO_DIRETAS
        CONTROLE_CONTRATACAO_DIRETAS = new_file
        save_config("CONTROLE_DADOS", str(new_file))
        self.controle_dir_updated.emit(new_file)

    def update_atas_dados_dir(self, new_file):
        global CONTROLE_ATAS_DADOS
        if new_file != CONTROLE_ATAS_DADOS:
            CONTROLE_ATAS_DADOS = new_file
            save_config("CONTROLE_DADOS", str(new_file))
            self.controle_dados_dir_updated.emit(new_file)
            print(f"CONTROLE_DADOS atualizado para {new_file}")

    def update_contratos_dados_dir(self, new_file):
        global CONTROLE_CONTRATOS_DADOS
        if new_file != CONTROLE_CONTRATOS_DADOS:
            CONTROLE_CONTRATOS_DADOS = new_file
            save_config("CONTROLE_DADOS", str(new_file))
            self.controle_dados_dir_updated.emit(new_file)
            print(f"CONTROLE_DADOS atualizado para {new_file}")

    def update_contratacoes_diretas_dados_dir(self, new_file):
        global CONTROLE_CONTRATACAO_DIRETAS
        if new_file != CONTROLE_CONTRATACAO_DIRETAS:
            CONTROLE_CONTRATACAO_DIRETAS = new_file
            save_config("CONTROLE_DADOS", str(new_file))
            self.controle_dados_dir_updated.emit(new_file)
            print(f"CONTROLE_DADOS atualizado para {new_file}")

    def update_sicaf_dir(self, new_dir):
        self.sicaf_dir_updated.emit(new_dir)

    def update_relatorio_path(self, new_dir):
        global RELATORIO_PATH
        RELATORIO_PATH = new_dir
        self.relatorio_path_updated.emit(new_dir)

# Instância global do gerenciador de eventos
global_event_manager = EventManager()
```