

## Arquivo: edit\_OM.py

```
from PyQt6.QtWidgets import *
from functools import partial
import json
from modules.utils.add_button import add_button_func
from modules.utils.linha_layout import linha_divisoria_sem_spacer_layout
from paths import ORGANIZACOES_FILE

from PyQt6.QtWidgets import QLabel, QScrollArea, QVBoxLayout, QWidget
from PyQt6.QtCore import Qt
import json
from paths import ORGANIZACOES_FILE
def show_organizacoes_widget(content_layout, icons, parent):
    """Exibe o widget para Alteração das Organizações Militares com funcionalidade de edição."""
    # Limpa o layout de conteúdo
    while content_layout.count():
        item = content_layout.takeAt(0)
        widget = item.widget()
        if widget:
            widget.deleteLater()
        elif item.layout():
            clear_layout(item.layout())

    def clear_layout(layout):
        """Recursivamente limpa um layout."""
        while layout.count():
            item = layout.takeAt(0)
            widget = item.widget()
            if widget:
                widget.deleteLater()
            elif item.layout():
                clear_layout(item.layout())

    # Carregar dados do arquivo JSON
    try:
        with open(ORGANIZACOES_FILE, 'r', encoding='utf-8') as file:
            config_data = json.load(file)
    except (FileNotFoundException, json.JSONDecodeError):
        config_data = {}

    # Widget principal para o conteúdo
    scroll_widget = QWidget()
    layout = QVBoxLayout(scroll_widget)

    # Título
    title = QLabel("Alteração das Organizações Militares")
    title.setStyleSheet("font-size: 20px; font-weight: bold; color: #4E648B")
    layout.addWidget(title)

    # Adiciona o widget de edição logo após o título
    edit_widget = EditOMWidget("organizacoes", config_data, parent)
```

```

layout.addWidget(edit_widget)

# Scroll Area para incluir o layout principal
scroll_area = QScrollArea()
scroll_area.setWidgetResizable(True)
scroll_area.setWidget(scroll_widget)

# Adiciona o Scroll Area ao layout principal
content_layout.addWidget(scroll_area)

class EditOMWidget(QWidget):
    def __init__(self, categoria, config_data, parent=None):
        super().__init__(parent)
        self.categoria = categoria
        self.config_data = config_data

        layout = QVBoxLayout(self)

        # Título
        title = QLabel(f"Edição de {categoria.replace('_', ' ').capitalize()}")
        title.setStyleSheet("font-size: 18px; font-weight: bold;")
        layout.addWidget(title)

        # Lista de itens
        self.list_widget = QListWidget()
        if categoria in config_data:
            for item in config_data[categoria]:
                item_text = f"UASG: {item['UASG']} - {item['Nome']} - {item['Sigla']} - {item['Indicativo']} - {item['Cidade']}"
                self.list_widget.addItem(item_text)
        layout.addWidget(self.list_widget)

        # Campos de edição
        layout.addWidget(QLabel("Nome:"))
        self.nome_input = QLineEdit()
        self.nome_input.setPlaceholderText("Digite o nome, Exemplo: Centro de Intendência da Marinha em Brasília")
        self.nome_input.textChanged.connect(self.forcar_caixa_alta)
        layout.addWidget(self.nome_input)

        layout.addWidget(QLabel("Sigla:"))
        self.sigla_input = QLineEdit()
        self.sigla_input.setPlaceholderText("Digite a Sigla, Exemplo: CeIMBra")
        self.sigla_input.textChanged.connect(self.forcar_caixa_alta)
        layout.addWidget(self.sigla_input)

        layout.addWidget(QLabel("UASG:"))
        self.uasg_input = QLineEdit()
        self.uasg_input.setPlaceholderText("Digite a UASG, Exemplo: 787010")
        layout.addWidget(self.uasg_input)

        layout.addWidget(QLabel("Indicativo:"))

```

```

self.indicativo_input = QLineEdit()
self.indicativo_input.setPlaceholderText("Digite o Indicativo, Exemplo: CITBRA")
layout.addWidget(self.indicativo_input)

layout.addWidget(QLabel("Cidade:"))
self.cidade_input = QLineEdit()
self.cidade_input.setPlaceholderText("Digite a Cidade, Exemplo: Brasília-DF")
layout.addWidget(self.cidade_input)

# Botões para adicionar e remover itens
button_layout = QBoxLayout()
add_btn = QPushButton("Adicionar")
add_btn.clicked.connect(self.adicionar_item)
button_layout.addWidget(add_btn)

remove_btn = QPushButton("Remover")
remove_btn.clicked.connect(self.remover_item)
button_layout.addWidget(remove_btn)

layout.addLayout(button_layout)

# Botão de salvar
save_btn = QPushButton("Salvar")
save_btn.clicked.connect(self.salvar)
layout.addWidget(save_btn)

# Conecta a seleção na lista para preencher os campos
self.list_widget.itemClicked.connect(self.preencher_campos)

def forcar_caixa_alta(self):
    """Garante que o nome seja sempre em caixa alta."""
    self.nome_input.setText(self.nome_input.text().upper())

def preencher_campos(self, item):
    """Preenche os campos de edição com os valores do item selecionado."""
    partes = item.text().split(" - ")
    if len(partes) == 5:
        self.uasg_input.setText(partes[0].split(":")[1].strip())
        self.nome_input.setText(partes[1].strip())
        self.sigla_input.setText(partes[2].strip())
        self.indicativo_input.setText(partes[3].strip())
        self.cidade_input.setText(partes[4].strip())

def adicionar_item(self):
    """Adiciona um novo item à lista."""
    nome = self.nome_input.text().strip()
    sigla = self.sigla_input.text().strip()
    uasg = self.uasg_input.text().strip()
    indicativo = self.indicativo_input.text().strip()
    cidade = self.cidade_input.text().strip()

    if not nome or not sigla or not uasg or not indicativo or not cidade:
        QMessageBox.warning(self, "Aviso", "Todos os campos devem ser preenchidos.")

```

```

    return

item_text = f"UASG: {uasg} - {nome} - {sigla} - {indicativo} - {cidade}"
self.list_widget.addItem(item_text)

# Limpa os campos após adicionar
self.nome_input.clear()
self.sigla_input.clear()
self.uasg_input.clear()
self.indicativo_input.clear()
self.cidade_input.clear()

def remover_item(self):
    """Remove o item selecionado da lista."""
    selected_item = self.list_widget.currentItem()
    if selected_item:
        self.list_widget.takeItem(self.list_widget.row(selected_item))

def salvar(self):
    """Salva as alterações na configuração."""
    items = []
    for i in range(self.list_widget.count()):
        partes = self.list_widget.item(i).text().split(" - ")
        if len(partes) == 5:
            items.append({
                "UASG": partes[0].split(":")[1].strip(),
                "Nome": partes[1].strip(),
                "Sigla": partes[2].strip(),
                "Indicativo": partes[3].strip(),
                "Cidade": partes[4].strip()
            })
    self.config_data[self.categoria] = items

    with open(ORGANIZACOES_FILE, 'w', encoding='utf-8') as file:
        json.dump(self.config_data, file, ensure_ascii=False, indent=4)

QMessageBox.information(self, "Sucesso", "Dados salvos com sucesso.")

```