Projeto KivyMD – Gestão de Utilizadores

Este notebook contém toda a informação partilhada nesta conversa sobre a criação de uma aplicação com KivyMD com:

- Abas de navegação
- Formulário com validação
- Gravação de dados em SQLite
- Visualização de dados
- Exportação para CSV
- Alternância de tema escuro/claro
- Empacotamento para Windows, Android e iOS

1. Estrutura do Projeto

```
📁 kivy_form_tabs/
├── main.py
├── main.kv
├── database.py
├── iniciar_app.bat
├── main.spec
└── buildozer.spec
```

2. Código Python – `main.py`

```
from kivymd.app import MDApp
from kivy.lang import Builder
from database import criar_tabela, inserir_utilizador, listar_utilizadores, exportar_csv
import re

class AppFormulario(MDApp):
    def build(self):
        criar_tabela()
        self.tema_claro = True
        return Builder.load_file("main.kv")

    def gravar_dados(self):
        nome = self.root.ids.nome_input.text.strip()
        email = self.root.ids.email_input.text.strip()
        idade = self.root.ids.idade_input.text.strip()

        if not nome or not email or not idade:
            self.mostrar_snackbar("Preenche todos os campos!")
```

```
        elif not idade.isdigit() or int(idade) <= 0:
            self.mostrar_snackbar("Idade inválida!")
        elif not re.match(r"[^@]+@[^@]+\.[^@]+", email):
            self.mostrar_snackbar("E-mail inválido!")
        else:
            inserir_utilizador(nome, email, int(idade))
            self.mostrar_snackbar("Utilizador guardado!")
            self.limpar_campos()
            self.carregar_utilizadores()

    def limpar_campos(self):
        self.root.ids.nome_input.text = ""
        self.root.ids.email_input.text = ""
        self.root.ids.idade_input.text = ""

    def mostrar_snackbar(self, texto):
        self.root.ids.snackbar.text = texto
        self.root.ids.snackbar.open()

    def carregar_utilizadores(self):
        lista = self.root.ids.lista_utilizadores
        lista.clear_widgets()
        for nome, email, idade in listar_utilizadores():
            lista.add_widget(
                self.criar_linha_utilizador(nome, email, idade)
            )

    def criar_linha_utilizador(self, nome, email, idade):
        from kivymd.uix.list import ThreeLineListItem
        return ThreeLineListItem(
            text=nome,
            secondary_text=email,
            tertiary_text=f"Idade: {idade}"
        )

    def exportar_csv(self):
        exportar_csv()
        self.mostrar_snackbar("Exportado para utilizadores.csv")

    def alternar_tema(self):
        self.tema_claro = not self.tema_claro
        self.theme_cls.theme_style = "Light" if self.tema_claro else "Dark"

AppFormulario().run()
```

3. Base de Dados – `database.py`

```
import sqlite3
import csv

def criar_tabela():
    conn = sqlite3.connect("utilizadores.db")
```

```
    conn = sqlite3.connect("utilizadores.db")
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS utilizadores (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nome TEXT,
            email TEXT,
            idade INTEGER
        )
    """)
    conn.commit()
    conn.close()

def inserir_utilizador(nome, email, idade):
    conn = sqlite3.connect("utilizadores.db")
    cursor = conn.cursor()
    cursor.execute("INSERT INTO utilizadores (nome, email, idade) VALUES (?, ?, ?)", (
    conn.commit()
    conn.close()

def listar_utilizadores():
    conn = sqlite3.connect("utilizadores.db")
    cursor = conn.cursor()
    cursor.execute("SELECT nome, email, idade FROM utilizadores")
    dados = cursor.fetchall()
    conn.close()
    return dados

def exportar_csv(caminho="utilizadores.csv"):
    dados = listar_utilizadores()
    with open(caminho, "w", newline="", encoding="utf-8") as f:
        writer = csv.writer(f)
        writer.writerow(["Nome", "Email", "Idade"])
        writer.writerows(dados)
```

4. Script Windows – `iniciar_app.bat`

```
@echo off
python main.py
pause
```

5. PyInstaller – `main.spec`

```
# -*- mode: python ; coding: utf-8 -*-
block_cipher = None
a = Analysis(
    ['main.py'],
    pathex=[],
    binaries=[],
    datas=[('main.kv', '.'), ('database.py', '.')],
```

```
    datas [( [illegible] )],
    hiddenimports=['kivymd'],
    hookspath=[],
    runtime_hooks=[],
    excludes=[],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=block_cipher,
)
pyz = PYZ(a.pure, a.zipped_data, cipher=block_cipher)
exe = EXE(
    pyz,
    a.scripts,
    [],
    exclude_binaries=True,
    name='GestaoUtilizadores',
    debug=False,
    bootloader_ignore_signals=False,
    strip=False,
    upx=True,
    console=False,
)
coll = COLLECT(
    exe,
    a.binaries,
    a.zipfiles,
    a.datas,
    strip=False,
    upx=True,
    name='GestaoUtilizadores'
)
```

6. Buildozer – `buildozer.spec`

```
[app]
title = GestaoUtilizadores
package.name = gestao
package.domain = org.exemplo
source.dir = .
source.include_exts = py,kv,db
version = 1.0
requirements = python3,kivy,kivymd,sqlite3
orientation = portrait
osx.kivy_version = 2.1.0

[buildozer]
log_level = 2
warn_on_root = 1

[android]
android.api = 31
android.minapi = 21
```

```
android.permissions = INTERNET
```

7. Instruções Windows

1. Instala Python 3.10+
2. Corre `iniciar_app.bat`
3. Ou usa `pyinstaller main.spec` para gerar `.exe`

8. Instruções Android (APK)

1. Instala Buildozer
2. Substitui `buildozer.spec`
3. Corre `buildozer -v android debug`

9. Instruções iOS

1. Usa um Mac com Xcode
2. Instala `kivy-ios`
3. Corre `toolchain.py create ios_app`
4. Copia o código e compila no Xcode