Sistema de Estoque — Documentação do Projeto

Este documento descreve a arquitetura, organização de pastas, padrões de UI/UX e como executar o projeto.

Sumário

- Visão geral
- Arquitetura e principais componentes
- Navegação e rotas
- Internacionalização (i18n)
- Estilos e temas
- Estrutura de pastas
- Convenções e dicas de layout (Flet)
- Como executar
- Roadmap (próximos passos)

Visão geral

Aplicativo de controle de estoque feito em Python com Flet. A navegação é baseada em um "shell" principal (Main View) com barra lateral (Sidebar). O conteúdo principal é carregado dinamicamente no centro sem troca de rota para a maioria das interações; rotas são usadas para estados principais como Login e a View principal.

Arquitetura e principais componentes

- App (**lib/app.py**)

- Configura título e preferências de janela/tema via **AppConfig**.

- Inicializa navegação com **PageManager**.

- Core (**lib/src/core/**)

- **page_manager.py**: responde a mudanças de rota (event.route), instancia e injeta Views.

- **routes.py**: centraliza constantes de rota (ex.: **/login**, **/main_view**).

- Views (**lib/src/app/views/**)

- **widgets/side_bar.py**: NavigationRail com labels traduzidas; emite seleção para atualizar conteúdo central.

- **pages/**: páginas de conteúdo como **welcome.py** e **products.py** (renderizadas dentro do shell principal).

- **login.py**: tela de autenticação.

- **home.py**/**main_view.py** (shell): layout com Row (Sidebar + área de conteúdo). A Sidebar troca o conteúdo central via callback.

- i18n (**lib/utils/**)

- **label_keys.py**: enum com chaves (APP_TITLE, MENU_HOME, WELCOME_TITLE etc.).

- **labels.py**: dicionários de traduções (pt/en) e helper **Labels.t**.

- Estilos (**lib/src/app/styles/**)

- **theme.py**: **ThemeManager** com cores e temas (dark/light).

- **image.py**: **ImagesAssets** e resolução de logos conforme tema.

Navegação e rotas

- As rotas (Login/Main View) usam **PageManager** com **event.route** e **page.go(...)**.

- Dentro da Main View, a Sidebar não muda a rota; apenas notifica a seleção (ex.: **home**, **products**) para que o shell atualize o container central.

- Benefícios: histórico e deep-link para estados macro (login/main), com UX fluida no conteúdo interno.

Internacionalização (i18n)

- Novas chaves para menu e WelcomePage: **MENU_***, **WELCOME_TITLE**, **WELCOME_INSTRUCTION**.

- **Labels.t(LabelKey.X)** retorna a string traduzida conforme **AppConfig.default_lang**.

Estilos e temas

- Uso de **ThemeManager** para cores, fundo, botões e fontes.

- Logos alternam conforme tema (claro/escuro).

- API atualizada do Flet: **ft.Colors** em vez de **ft.colors**.

Estrutura de pastas (resumo)

config.ini

main.py

lib/

app.py

src/

app/

styles/

views/

pages/

widgets/

core/

page_manager.py

routes.py

config/

app_config.py

utils/

label_keys.py

labels.py

assets/

images/

storage/

data/

temp/

Convenções e dicas de layout (Flet)

- Para ocupar todo o espaço disponível:

- Em layouts com **Row**, use **vertical_alignment=ft.CrossAxisAlignment.STRETCH**.

- Em **Column**, use **expand=True** e **horizontal_alignment=ft.CrossAxisAlignment.STRETCH**.

- Para conteúdo que deve crescer, use **ft.Expanded** ou **expand=True** e envolva tabelas em **ft.ListView**/**ft.Container** com **expand=True** para scroll.

- Sidebar

- Alterna **label_type** (NONE/ALL) no hover; labels vêm do i18n.

Como executar

- Requisitos: Python 3.11+ e Flet.

- Opção 1 (CLI do Flet):

- Dentro do diretório do projeto, execute: **flet run**

- Opção 2 (Python):

- Ative a venv e rode **python main.py**.

Roadmap (próximos passos)

- Mover páginas para **views/pages** e criar controladores por domínio (ex.: Produtos).

- Paginação/ordenção/CRUD na tabela de produtos.

- Testes de UI e linter/formatador (ruff/black) no repositório.