Widgets são a hierarquia de classes central no framework Flutter. Um widget é uma descrição imutável de parte de uma interface de usuário. Widgets podem ser inflados em elementos, que gerenciam a árvore de renderização subjacente (que vem abaixo, na sequência). Os próprios widgets não têm estado mutável (todos os campos devem ser finais).

Material Design são boas práticas de design propostas pela google. As widgets segue esse padrão

**Widgets:** são componentes para construção da interface dos aplicativos. Abaixo algumas widgets.

**Container:** é uma widget para colocar outros componentes dentro (outros widgets). É um componente base/raiz. Ele se comportar de forma a adequar todos os filhos da melhor forma internamente.

**Row:** os componentes serão dispostos um ao lado do outro.

**Column:** os componentes serão dispostos um abaixo do outro.

Image:

Text:

Icon:

Os aplicativos Flutter podem incluir código e ativos (às vezes chamados de recursos). Um ativo é um arquivo que é empacotado e implantado com seu aplicativo e está acessível no tempo de execução. Tipos comuns de ativos incluem dados estáticos (por exemplo, arquivos JSON), arquivos de configuração, ícones e imagens. (fonte: <https://www.macoratti.net/>)

**s10-60\_espacamentos**

decoration: define uma borda e suas características;

padding: espaçamentos internos;

margin: espaçamentos externos;

**s10-61\_alinhamento**

mainAxisAlignment: alinhamento do eixo principal - define o alinhamento principal e depende de qual widtger está sendo aplicado;

crossAxisAlignment: alinhamento do eixo que cruza o eixo principal (mainAxis) - define o alinhamento principal;

Center(): coloca o objeto centralizado na tela;

width: double.infinity: “estica” o objeto 100% do espaço disponível na tela;

**s10-62\_imagens**

**s10-63\_layout base com scaffold**

scaffold (estrutura/armação/esqueleto): define uma estrutura base para montagem da tela;

*scafoold(*

*appBar: AppBar(),*

*body: Text(“asdf”),*

*bottomNavigationBar: BottomAppBar(),*

*)*

**s10-64\_stateless e statefull**

stateless: widgits que não podem ser alterados (constantes);

statefull: widgits que podem ser alterados (variáveis);

setState(): quando chamado atualiza somente o necessário e não toda a classe principal gerando economia de recurso e tempo de execução;

**s13-87\_Entendendo requisições e Api**

**Como funciona a Internet?**

Navegador -> Faz a requisição ao servidor -> Servidor responde com HTML/CSS -> Navegador exibe a resposta

**Requisições HTTP**

- HTTP (Hypertext Transfer Protocol): padronizou a comunicação entre navegadores e servidores

- Alguns tipos de requisições utilizando o protocolo HTTP

- Get -> recuperar dados no servidor;

- Post -> Criar dado novo no servidor;

- Put -> Atualizar dados no servidor;

- Delete -> Deletar dados no servidor.

**Alguns códigos de Status**

- 404 -> recurso não encontrado, ex.: links

- 200 -> ok

- 501 -> Bad Gateway

**Utilizando uma API**

- API (Application Programming Interface – Interface de Programação de Aplicações)

- Requisição: <http://site.com/clima?cidade=”palmas>” -> api interage com banco de dados -> api responde { “temperatura”: 20, “unidade”: “Celsius” }

- A resposta deve ser de baixa complexidade e simples interpretação para qualquer linguagem de programação. A resposta não deve ser dependente da linguagem de programação

**Tipos de retornos API**

- XML (eXtensible Markup Language)

<resultado>

<temperatura>20</temperatura>

<unidade>Celsius</unidade>

<resultado>