**FLUTTER E WIDGETS INTERNAMENTE**

* **60FPS É UM PROBLEMA?**
  + É A taxa de atualização do framework de 60 quadros por segundo, ou seja, ele vai pintar os pixels na tela 60x a cada segundo.
  + Não é um problema, é muito rápido
  + Problema seria se fosse necessário o flutter recalcular tudo , definir temas, aplicar os temas aos componentes 60x p s
* **ÁRVORE DE WIDGETS E ÁRVORE ELEMENTOS**
  + Sua aplicação é composta por um conjunto de widgets, componentes e esses componentes vao fazendo composição entre si para formar outros componentes q formam outros componentes ... formando uma arvore
    - **Composição agressiva**: pq existe uma quantidade enorme de componente para formar a app.
  + **Árvores** 
    - **Widgets:** define a estrutura da aplicação, é a arvore de configuração
      * Reconstruído frequentemente
      * Imutável – mesmo se for um componente statefull, o statefull widget é imutável apesar que tem o estado, o estado pode evoluir, mas o widget é reconstruído quando há mudança
      * Quando é marcado como sujo, no próximo frame é recriado, pq é imutável
    - **Elementos:** liga o widget com o objeto renderizado na tela
      * Raramento reconstruído
      * Estrutura logica: ex o estado do componente de um widget statefull
      * Para todo widget temos um elemento com uma referencia para ele.
      * Os elementos não são recriados como os widgets
      * As referencias é que sõa atualizadas, deixando de apontar para o componente velho para o novo componente e vai gerar impacto na arvore de renderizacao.
      * **Keys chaves:** vai impedir determinados problemas, via especificar o widget e a referencia não vai “errar” o caminho. Com a key vc vai conseguir identificar o elemento dentro unicamente dentro de um determinado contexto. Use o **ValueKey() –** (chave local)**, GlobalObjectKey()** – (vai ser única dentro de toda arvore) para criar a chave
    - **Render :** objeto renderizado na tela
      * Raramente reconstruído
      * O que vc ve na tela
      * Mais baixo nível
  + **Quando o método BUILD é chamado?**
    - Quando monta a tela pela 1 vez
    - Com setState
      * Quando altera o estado de um objeto, este obj é marcado como sujo (alterado) e ainda não foi renderizado, então próximo ciclo (60fps) ele vai perceber que tem um objeto sujo, ou seja, q não foi renderizado, então ele vai chamar toda a arvore de widget para atualização, depois atualizar as referencias do elementos (pq não serão recriados), ai fará a alteração na arvore render (pixels)
    - Com o MediaQuery.of
      * Quando há uma mudança na orientação do dispositivo p ex fica claro que os pixels precisam ser redesenhado na rela.
      * Tudo q envolve o MediaQuery vai afetar, como tamanho, espaçamento...
  + **Construtores e widgets constantes**
    - **Variável const:** uma variável constante em tempo de compilação,
    - **Variável final:** uma variável constante em runtime(tempo de execução)
    - **Construtor const:**  para ser marcado como const, todos os atributos devem ser final
    - **Q**uando um widget tem a possibilidade de ser marcado como constante, o flutter não terá o trabalho de reconstruí lo
  + **Ciclo de Vida dos Widgets**
    - **Stateless**
      * Construtor – cria o objeto
      * Build – renderiza a arvore
    - **Statefull**
      * Construtor
      * initState – inicializar o estado do componente é chamado este método e aqui vc pode colocar algum caso queira.
      * Build
      * setState
      * didUpdateWidget- é chamando logo após alguma mudança. Aqui vc pode fazer alguma comparação entre o old e new widget
      * build
      * dispose: quando o componente sai da tela
  + **Ciclo de vida da Aplicação:** os métodos do ciclo de vida da própria aplicação em flutter
    - **Ciclos:**
      * **Inativo:** app está inativa, nenhuma entrada do usuário é recebida. Ocoree na transição de uma estado para outro
      * **Paused:** app não visível, executando em background. Quando sai da aplicação p ex e vai para a tela home. Ela continua rodando e vc pode usar esse ciclo de vida para fazer alguma limpeza.
      * **Resumed:** app novamente visível e respondendo ao usuário. A aplicação volta a ficar ativa, ou seja, fica novamente visível. Vc deixa em background depois vc volta.
      * **Suspending:** app será suspensa(sair).
    - **Mixim:** é mistura, quando vc quer reaproveitar código quando sem necessariamente precisar da herança, vc pega um código e anexa onde vc quer. **with WidgetsBindingObserver**
    - **Método: didChangeAppLifecycleState(AppLifecycleState state)** este método é chamado sempre q o estado da aplicação for mudada. Sendo que precisamos registrar esta classe como observer para receber notificao sempre q for mudado
  + **Entendendo o Contexto BuildContext**
    - Para todo meotodo build é passado para ele um buildContext
    - O buildContext é único para cada elemento.
    - Como a estrutura é em forma de arvore, cada BuildCOntext tem sua forma de comunicar e referencia o BuildCOntext do pai/filho
    - **Sçao META informações sobre o widget e sua localização na árvore de componentes. Exemplo de quem usa:** media query, navegator, theme
    - **InheritedWidget: componente herdado**
      * Ele vai ajudar para se comunicar com widget mais distante
      * **Qunando vc diz que um componente x é herdado, então torna –se possível ter um túnel de comunicao direta entre o componente y que vc precisa recuperar uma informação do componente x que tem a informação.** 
        + Exemplo de uso: media query – é um componente herdado, ele herda do **InheritedWidget**. E usando o método estático of consegue ter as informações do media query