**MOBX – GERENCIADOR DE ESTADO**

* **Dependencias**
  + Mobx
  + Flutter-mobx
  + Provider
* Dependências\_dev
  + Build\_runner
  + Mobx\_codegem
* **Princípios do MOBX**
  + Ele tem 3 pilares
    - **Actions – ação**
    - **Observables – ação**
    - **Reactions - reação**
  + Possui uma programação reativa pq primeiro ocorre uma mudança e quando modificar é disparado uma notificação a todos os interessados para q haja uma reação.
* **@observable**:
  + É um conceito que no TS chamamos de decoreator e no java annotation
  + Este é um conceito q não está disponível diretamente no dart, nativamente ele não interpreta isso
* **@action**:
  + Indica o local onde estou mudando o observable
* **Build\_runner + mobx\_codegen**: estas duas ferramentas (codegen vem de geração de código por mobx), vai gerar um código (daqueles decoration) para que aqueles decoration sejam realmente um observable/action
* **Como configurar**
  + Criamos uma **private abstract class** com mixin with Store p ex: \_cont
  + **Importamos** um arquivo que será gerado que terá o mesmo nome que criamos com uma extensão .g .dart e dentro deste arquivo vai ser criado um mixin que vai ter o mesmo nome da class so que com um $ na frente \_$cont, então usamos este mixin para gerar a classe que de fato iremos instanciar **class cont = \_cont with \_$cont**
  + Comando**: flutter pub run build\_runner watch**
    - Build: executa o build uma única vez
    - Clean: limpa qq build feito anteriormente
    - Run
    - Watch: fica monitorando qq mudança no arquivo e a partir das mudanças ele fica regerando as informações
  + **Envolve** o objeto/área que quer ser notificado com o **Observer**
* Com este gerenciador é possível fazer uma “simulação” do método setState apenas disponível na class State do Statefull