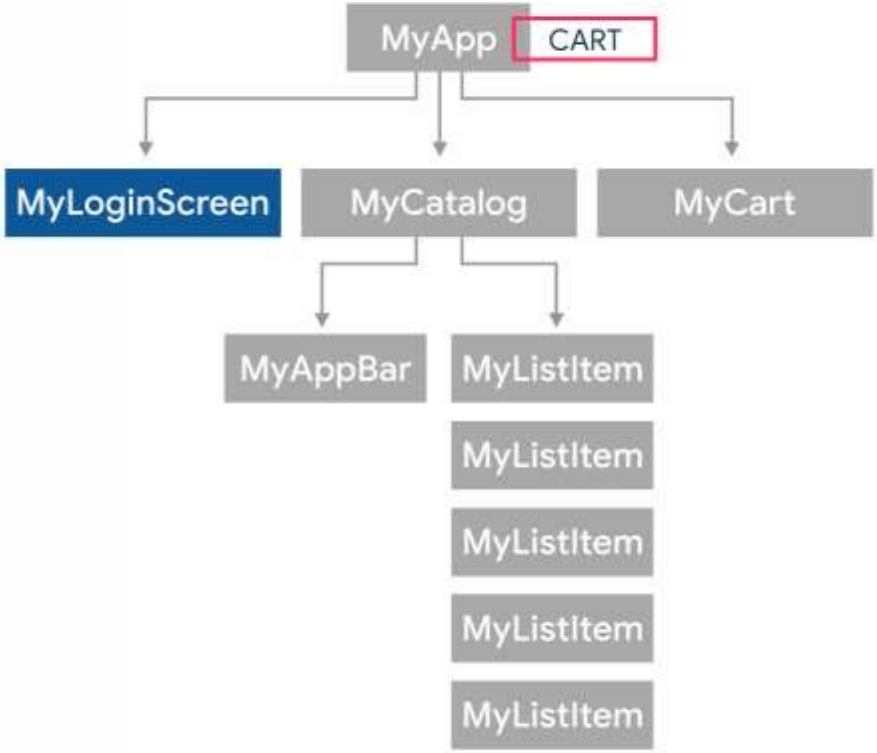
Gerenciamento de Estado


Welcome
Login
Password
Enter


MyApp
CART
MyLoginScreen
MyCatalog
MyCart
MyAppBar
MyListItem
MyListItem
MyListItem
MyListItem
MyListItem

Declarative UI

Declarative UI

- Isso significa que o **Flutter** constrói sua interface de usuário para **refletir** o estado atual do seu aplicativo
- Sempre que o estado da aplicação muda, os widgets são reconstruídos para atender aquele novo estado

Declarative UI

- No **Flutter**, não há problema em reconstruir partes de sua Interface do zero em vez de modificá-la. **Flutter** é rápido o suficiente para fazer isso, mesmo em todos os quadros, se necessário.

https://flutter.dev/docs/development/data-and-backend/state-mgmt/declarative

setState()

setState()

- O método ***setState(),*** que é exclusivo ao widgets do tipo *Stateful*, é o mais simples, básico e fácil de entender, quando se deseja adicionar interatividade em um aplicativo.

```
int _counter = 0;

void _incrementCounter() {
  setState(() {
    _counter++;
  });
}
```

setState()

- Ao chamar o método ***setState****(),* nosso widget será reconstruído já com o novo valor da variável contador tendo sido incrementado.
- O grande problema com ele é que à medida que o aplicativo for crescendo e ganhando complexidade, pode ser necessário alterações em vários widgets diferentes, e controlar as chamadas do ***setState()*** em todas elas acaba ficando inviável.

setState()

- O ***setState()*** é ideal para o estado local de um widget, como marcar/desmarcar um *checkbox*/*switch*.
- Para o estado global da aplicação (ex: usuário logado, carrinho de compras) há outras opções mais robustas e que permitem atualizar vários widgets quando o estado sofrer alterações.

Gerenciamento de Estado

- setState
- InheritedWidget & InheritedModel
- Provider
- Redux
- MobX

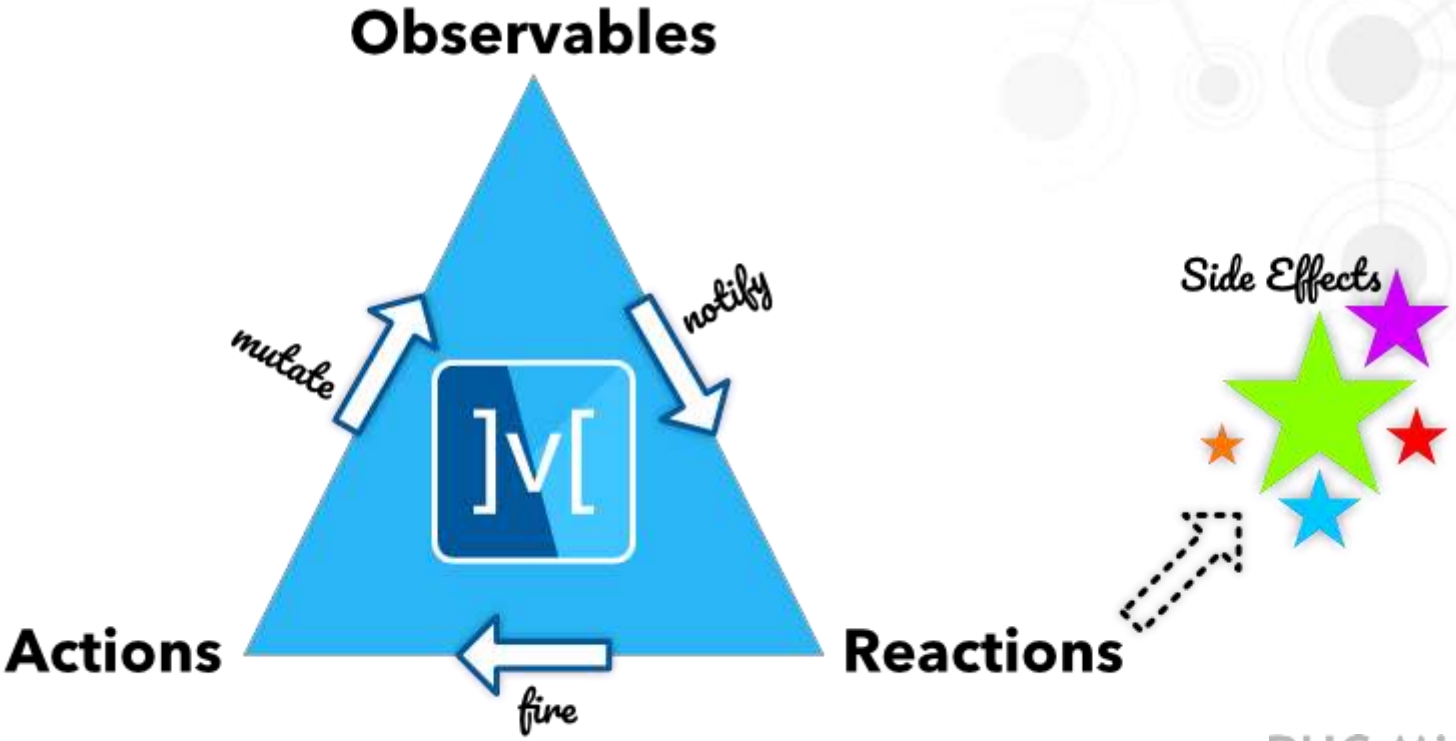
https://flutter.dev/docs/development/data-and-backend/state-mgmt/options

MobX

MobX

- **MobX** é uma biblioteca de gerenciamento de estado que simplifica a conexão dos dados reativos de seu aplicativo.
- MobX simplifica o gerenciamento de estado criando um **Observable** ao redor de um *Widget* que queremos manter sempre atualizado.

MobX



Fonte: https://pub.dev/packages/mobx

MobX

- Para usar este recurso você deve adicionar as dependências “**mobx[1]**” e “**flutter_mobx[2]**” no arquivo “pubspec.yaml”

```
dependencies:
  flutter:
    sdk: flutter
  mobx: ^1.2.1+3
  flutter_mobx: ^1.1.0+2
```

[1] https://pub.dev/packages/mobx
[2] https://pub.dev/packages/flutter_mobx

MobX

- Você também deve adicionar as dependências de desenvolvimento "**build_runner**[1]" e "**mobx_codegen**[2]" no arquivo "pubspec.yaml"

```
dev_dependencies:
  flutter_test:
    sdk: flutter
  build_runner: ^1.10.3
  mobx_codegen: ^1.1.1+1
```

[1] https://pub.dev/packages/build_runner
[2] https://pub.dev/packages/mobx_codegen

Observables

- ***Observables*** representam o estado reativo de seu aplicativo.
- Quando os ***Observables*** mudam, todas as reações são executadas novamente. O que é interessante é que essas reações podem ser qualquer coisa, desde um simples log do console, uma chamada de rede para renderizar novamente a IU.

Observables

```dart
import 'package:mobx/mobx.dart';

part 'counter.g.dart';

class Counter = CounterBase with _$Counter;

abstract class CounterBase with Store {
  @observable
  int value = 0;

  @action
  void increment() {
    value++;
  }
}
```

Observables

```
@override
Widget build(BuildContext context) => Scaffold(
  appBar: AppBar(
    title: const Text('Counter'),
  ),
  body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        const Text(
          'You have pushed the button this many times:',
        ),
        Observer(
            builder: (_) => Text(
              '${_counter.value}',
              style: const TextStyle(fontSize: 20),
            )),
      ],
    ),
  ),
  ...
);
}
```

Referências Bibliográficas

- **Flutter:**
  - https://flutter.dev/
- **State management:**
  - **https://flutter.dev/docs/development/data-and-backend/state-mgmt/intro**
- **Start thinking declaratively:**
  - **https://flutter.dev/docs/development/data-and-backend/state-mgmt/declarative**
- **Github MobX:**
  - **https://github.com/mobxjs/mobx.dart**
- **MobX.dart:**
  - **https://mobx.netlify.app/getting-started/**