

**Aplicativo para gerenciar uma lista de compras**

**ICEI – PUC Minas**

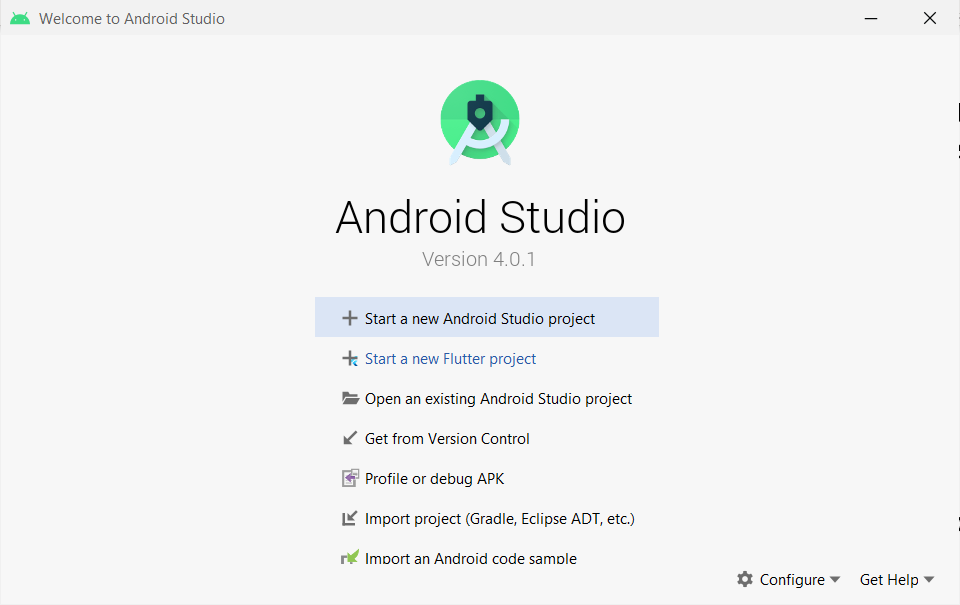
MobX

**Especificação do Aplicativo**

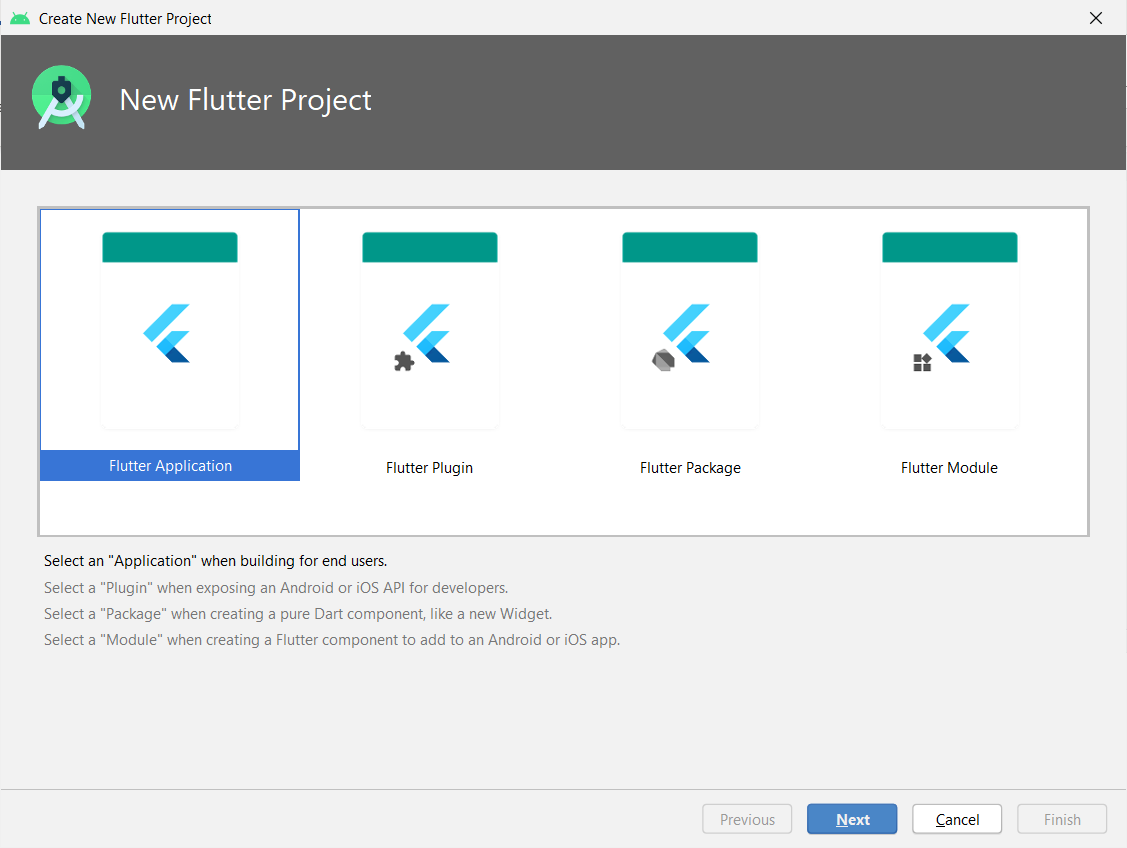
Nesta atividade iremos desenvolver um aplicativo para gerenciar uma lista de compras utilizando o MobX para realizar o gerenciamento de estado*.*

**Criando o Projeto**

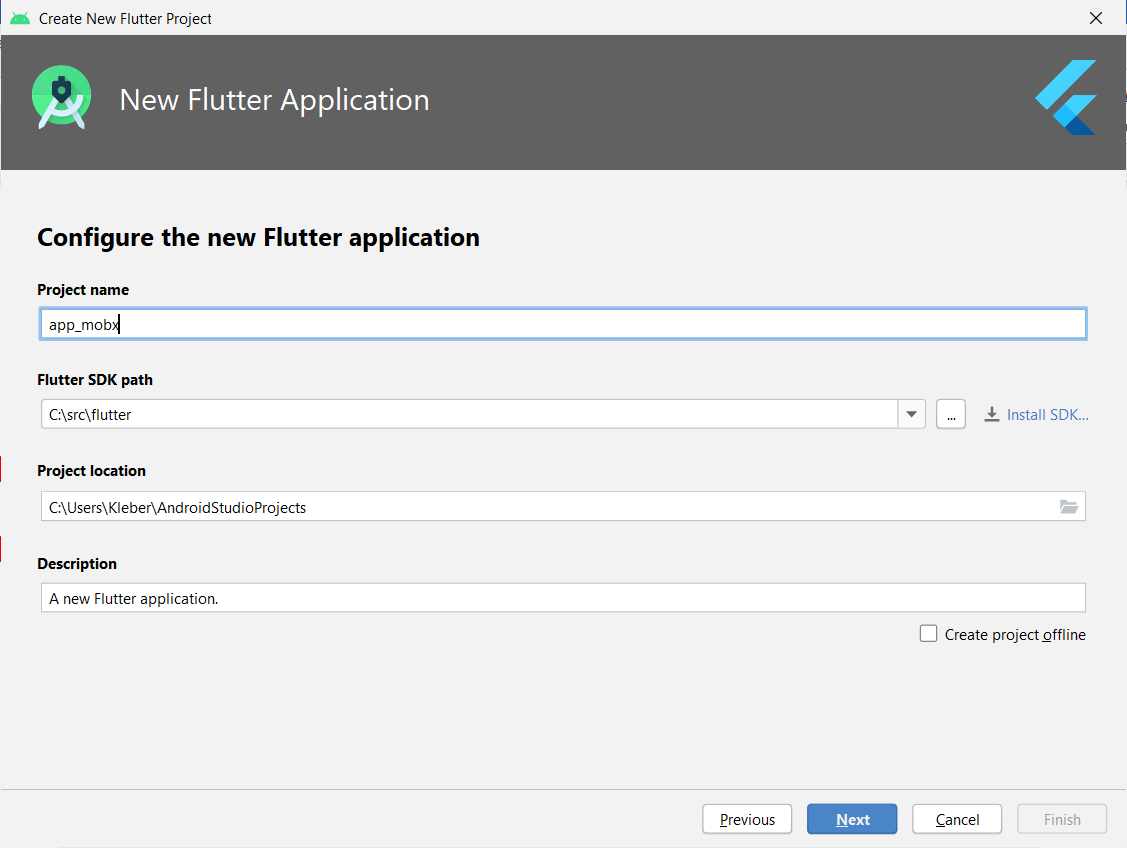
No Android Studio clique em “***Start a new Flutter project***”.

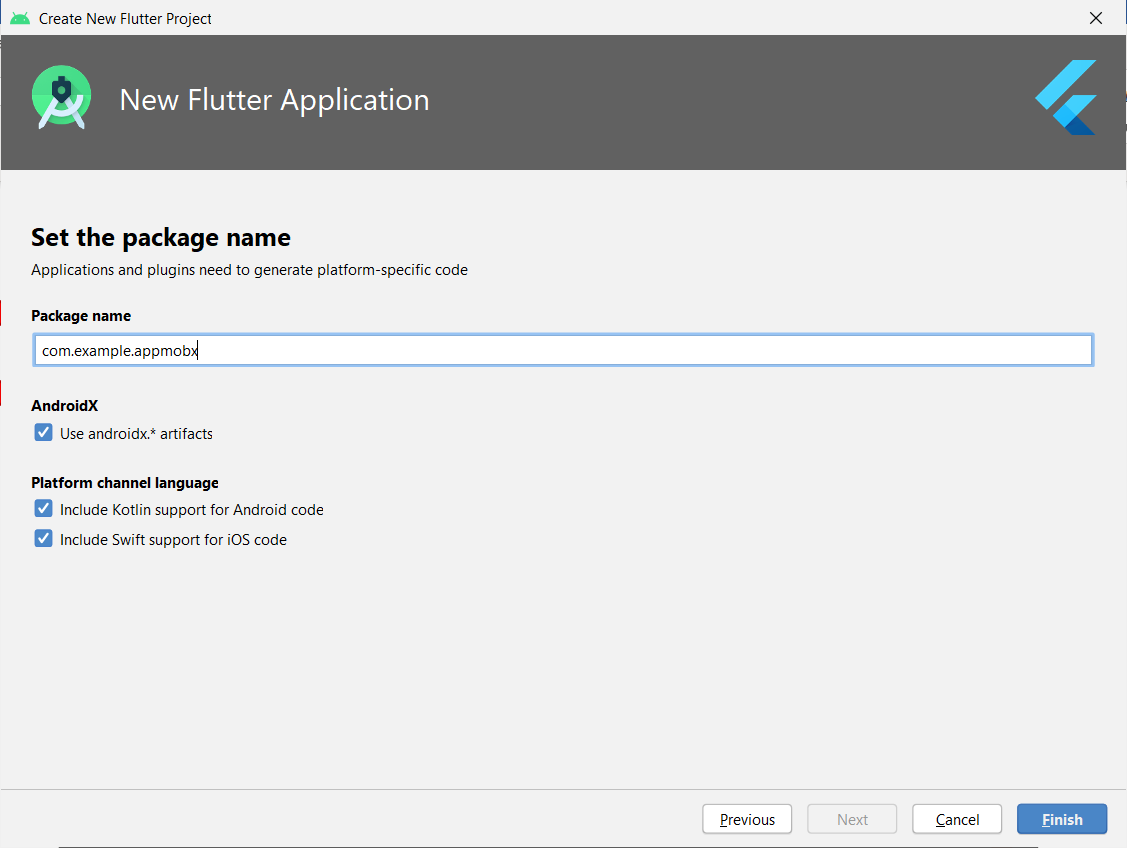


Depois selecione “***Flutter Application***” e clique em “***Next***”.



Preencha as informações do projeto e depois clique em “**Next**”. Na próxima tela, basta clicar em “***Finish***”.





**Configurando o MobX**

Para usarmos o recurso MobX, é necessário incluir as dependências no projeto.

Para adicionar a dependência abra o arquivo “**pubspec.yaml**” e adicione na seção *dependencies*:

**dependencies:  
 flutter:  
 sdk: flutter  
 mobx: ^1.2.1+3  
 flutter\_mobx: ^1.1.0+2**

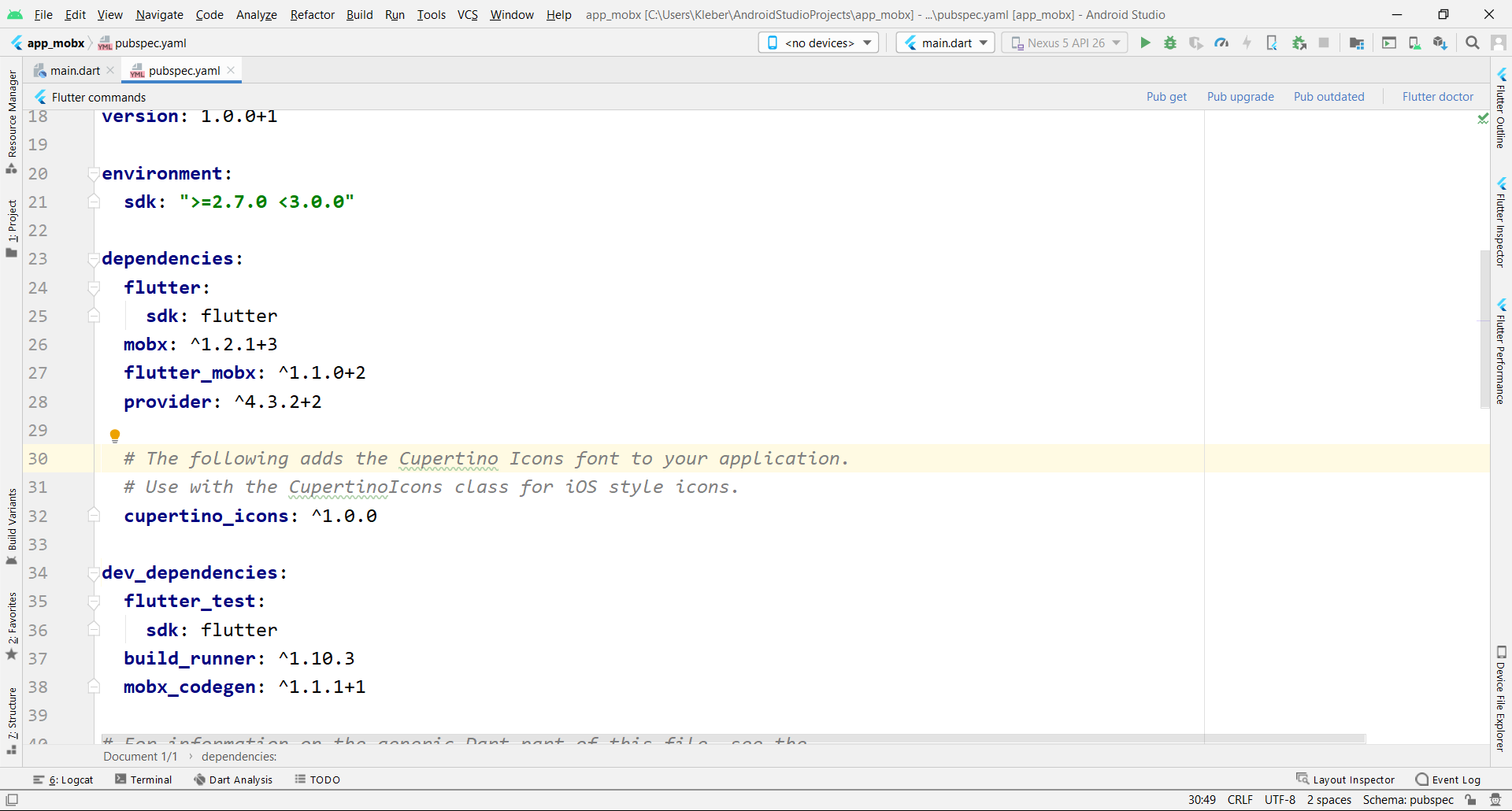
**provider**: **^4.3.2+2**

Além disso, também precisamos adicionar as dependências de desenvolvimento.

Para adicionar as dependências de desenvolvimento abra o arquivo “**pubspec.yaml**” e adicione na seção *dependencies*:

**dev\_dependencies:  
 flutter\_test:  
 sdk: flutter  
 build\_runner: ^1.10.3  
 mobx\_codegen: ^1.1.1+1**

Após realizar a alteração do arquivo clique em “Pub get”, para atualizar as dependências do projeto.



**Estruturando o Projeto**

Vamos organizar o projeto usando o padrão de projeto MVC, para isso, crie os seguintes pacotes dentro da lib:

* lib
  + controllers
  + models
  + repositories
  + views

Para fazer a gestão global do estado do aplicativo iremos um enumerador que armazenará o estado atual da aplicação. Para isso, na pasta “lib” crie o arquivo “app\_status.dart”, e digite o seguinte código dentro dele:

**enum** AppStatus {**loading**, **success**, **error**, **none**}  
  
**extension** AppStatusExt **on** AppStatus{  
 **static var** *\_value*;  
 **get value** => *\_value*;  
 **set value**(value) => *\_value* = value;  
}

O enumerador “AppStatus” apresenta todas as possibilidades de estados da nossa aplicação e a extensão “AppStatusExt” permite armazenar valores nesse estado, que podem auxiliar no armazenamento de mensagem de estados.

**Configurando o Model**

Iremos definir uma classe Produto, para armazenar as informações dos produtos que iremos adicionar em nossa lista de compras. Para isso, na pasta “models” crie o arquivo “produto.model.dart”, e digite o seguinte código dentro dele:

**class** Produto {

String **nome**;  
 bool **concluido**;  
  
 Produto({**this**.**nome**, **this**.**concluido**});  
  
 Produto.fromJson(Map<String, **dynamic**> json) {  
 **nome** = json[**'nome'**];  
   
 **concluido** = json[**'concluido'**];  
 }  
  
 Map<String, **dynamic**> toJson() {  
 **final** Map<String, **dynamic**> data = **new** Map<String, **dynamic**>();  
 data[**'nome'**] = **this**.**nome**;  
 data[**'concluido'**] = **this**.**concluido**;  
 **return** data;  
 }  
}

Iremos armazenar as informações do nome, quantidade e estado de concluído, do produto da lista de compras.

**Configurando o Repository**

Como nosso objetivo é exercitar o gerenciamento de dados, iremos usar uma lista de Produtos na memória, posteriormente, podemos adicionar uma camada de persistência de dados em nossa aplicação. Para isso, na pasta “repositories” crie o arquivo “produto.repository.dart”, e digite o seguinte código dentro dele:

**import 'package:app\_mobx/models/produto.model.dart'**;  
  
**class** ProdutoRepository{  
  
 List<Produto> **list**;  
  
 ProdutoRepository(){  
 **list** = **new** List<Produto>();  
 **list**.add(Produto(nome:**"Arroz"**, concluido: **false**));  
 **list**.add(Produto(nome:**"Feijão"**, concluido: **false**));  
 }  
  
 Future<List<Produto>> getAll() **async** {  
 **try** {  
 **await new** Future.delayed(**const** Duration(seconds : 1));  
 **list**.sort((a, b){  
 **if**(a.**concluido**&& !b.**concluido**) **return** 1;  
 **else if**(!a.**concluido** && b.**concluido**) **return** -1;  
 **else return** 0;  
 });  
 **return list**;  
 } **catch** (e) {  
 **return null**;  
 }  
 }  
  
 Future<bool> create(Produto p) **async** {  
 **try** {  
 **await new** Future.delayed(**const** Duration(seconds : 1));  
 **list**.add(p);  
 **return true**;  
 } **catch** (e) {  
 **return false**;  
 }  
 }  
  
 Future<bool> update(int id, Produto p) **async** {  
 **try** {  
 **await new** Future.delayed(**const** Duration(seconds : 1));  
 **list**[id] = p;  
 **return true**;  
 } **catch** (e) {  
 **return false**;  
 }  
 }  
  
 Future<bool> delete(int id) **async** {  
 **try** {  
 **await new** Future.delayed(**const** Duration(seconds : 1));  
 **list**.removeAt(id);  
 **return true**;  
 } **catch** (e) {  
 **return false**;  
 }  
 }  
}

O código “**await new** Future.delayed(**const** Duration(seconds : 1));” serve apenas para criar um atraso de 1 segundo na operação, mas simular um acesso ao banco de dados ou uma API Web.

**Configurando o Controller**

Em nosso Controller que iremos adicionar a lógica de gestão de estado para nossa aplicação. Para isso, na pasta “controllers” crie o arquivo “produto.controller.dart”, e digite o seguinte código dentro dele:

**import 'package:app\_mobx/repositories/produto.repository.dart'**;  
**import 'package:mobx/mobx.dart'**;  
**import 'package:app\_mobx/models/produto.model.dart'**;  
  
**import '../app\_status.dart'**;  
**part 'produto.controller.g.dart'**;  
  
**class** ProdutoController = \_ProdutoController **with** \_$ProdutoController;  
  
**abstract class** \_ProdutoController **with** Store{  
  
 ProdutoRepository **repository**;  
  
 \_ProdutoController(){  
 **repository** = **new** ProdutoRepository();  
 \_init();  
 }  
  
 \_init() **async** {  
 getAll();  
 }  
  
 @observable  
 AppStatus **status** = AppStatus.**none**;  
  
 @observable  
 ObservableList<Produto> **list** = ObservableList<Produto>();  
  
 @action  
 Future<**void**> getAll() **async** {  
 **status** = AppStatus.**loading**;  
 **try** {  
 **final** allList = **await repository**.getAll();  
 **list**.clear();  
 **list**.addAll(allList);  
 **status** = AppStatus.**success**;  
 }**catch**(e){  
 **status** = AppStatus.**error**..**value** = e;  
 }  
 }  
  
 @action  
 Future<**void**> create(Produto p) **async** {  
 **status** = AppStatus.**loading**;  
 **try** {  
 **await repository**.create(p);  
 **await** getAll();  
 **status** = AppStatus.**success**;  
 }**catch**(e){  
 **status** = AppStatus.**error**..**value** = e;  
 }  
 }  
  
 @action  
 Future<**void**> update(int id, Produto p) **async** {  
 **status** = AppStatus.**loading**;  
 **try** {  
 **await repository**.update(id, p);  
 **await** getAll();  
 **status** = AppStatus.**success**;  
 }**catch**(e){  
 **status** = AppStatus.**error**..**value** = e;  
 }  
 }  
  
 @action  
 Future<**void**> delete(int id) **async** {  
 **status** = AppStatus.**loading**;  
 **try** {  
 **await repository**.delete(id);  
 **await** getAll();  
 **status** = AppStatus.**success**;  
 }**catch**(e){  
 **status** = AppStatus.**error**..**value** = e;  
 }  
 }  
  
}

A variável “status”, que é do tipo “AppStatus” irá armazenar o estado da nossa aplicação. Assim, em cada operação podemos alterar e controlar o seu estado.

@observable  
AppStatus **status** = AppStatus.**none**;

Para que o MobX gere todo o código de gerenciamento de estado, execute o código abaixo:

flutter packages pub run build\_runner build

Caso deseje realizar alterações e não precisar ficar executando este código, toda vez que alterar o Controller, você pode usar o seguinte código:

flutter packages pub run build\_runner watch

Assim, irá ser gerado o arquivo “produto.controller.g.dart” com o código necessário para realizar o gerenciamento de código da nossa aplicação.

**Configurando View**

Agora iremos adicionar a camada View em nossa aplicação. Para isso, na pasta “views” crie o arquivo “produto.view.dart”, e digite o seguinte código dentro dele:

**import 'package:app\_mobx/controllers/produto.controller.dart'**;  
**import 'package:app\_mobx/models/produto.model.dart'**;  
**import 'package:flutter/material.dart'**;  
**import 'package:flutter\_mobx/flutter\_mobx.dart'**;  
**import 'package:provider/provider.dart'**;  
  
**import '../app\_status.dart'**;  
  
**class** ProdutoListView **extends** StatefulWidget {  
 @override  
 \_ProdutoListViewState createState() => \_ProdutoListViewState();  
}  
  
**class** \_ProdutoListViewState **extends** State<ProdutoListView> {  
 **final \_formKey** = GlobalKey<FormState>();  
 **var \_itemController** = TextEditingController();  
 ProdutoController **\_controller** = **null**;  
  
 @override  
 **void** initState() {  
 **super**.initState();  
 WidgetsBinding.*instance*.addPostFrameCallback((\_) **async** {  
 **await \_controller**.getAll();  
 });  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 **\_controller** = Provider.*of*<ProdutoController>(context);  
  
 **return** Scaffold(  
 appBar: AppBar(title: Text(**'Lista de Compras'**), centerTitle: **true**),  
 body: Scrollbar(  
 child: Observer(builder: (\_) {  
 **if** (**\_controller**.**status** == AppStatus.**loading**) {  
 **return** Center(  
 child: CircularProgressIndicator(),  
 );  
 } **else if** (**\_controller**.**status** == AppStatus.**success**) {  
 **return** ListView(  
 children: [  
 **for** (int i = 0; i < **\_controller**.**list**.**length**; i++)  
 ListTile(  
 title: CheckboxListTile(  
 controlAffinity: ListTileControlAffinity.**leading**,  
 title: **\_controller**.**list**[i].**concluido** ? Text(  
 **\_controller**.**list**[i].**nome**,  
 style: TextStyle(  
 decoration: TextDecoration.*lineThrough*),  
 )  
 : Text(**\_controller**.**list**[i].**nome**),  
 value: **\_controller**.**list**[i].**concluido**,  
 secondary: IconButton(  
 icon: Icon(  
 Icons.*delete*,  
 size: 20.0,  
 color: Colors.*red*[900],  
 ),  
 onPressed: () **async** {  
 **await \_controller**.delete(i);  
 },  
 ),  
 onChanged: (c) **async** {  
 Produto edited = **\_controller**.**list**[i];  
 edited.**concluido** = c;  
 **await \_controller**.update(i, edited);  
 },  
 )),  
 ],  
 );  
 }**else**{  
 **return** Text(**"Erro"**);  
 }  
 }),  
 ),  
 floatingActionButton: FloatingActionButton(  
 child: Icon(Icons.*add*),  
 onPressed: () => \_displayDialog(context),  
 ),  
 );  
 }  
  
 \_displayDialog(context) **async** {  
 **return** showDialog(  
 context: context,  
 builder: (context) {  
 **return** AlertDialog(  
 content: Form(  
 key: **\_formKey**,  
 child: TextFormField(  
 controller: **\_itemController**,  
 validator: (s) {  
 **if** (s.**isEmpty**)  
 **return "Digite o produto."**;  
 **else  
 return null**;  
 },  
 keyboardType: TextInputType.*text*,  
 decoration: InputDecoration(labelText: **"Produto"**),  
 ),  
 ),  
 actions: <Widget>[  
 FlatButton(  
 child: **new** Text(**'CANCEL'**),  
 onPressed: () {  
 Navigator.*of*(context).pop();  
 },  
 ),  
 FlatButton(  
 child: **new** Text(**'SALVAR'**),  
 onPressed: () {  
 **if** (**\_formKey**.**currentState**.validate()) {  
 **\_controller**.create(Produto(nome: **\_itemController**.**text**, concluido: **false**));  
 **\_itemController**.**text** = **""**;  
 Navigator.*of*(context).pop();  
 }  
 },  
 )  
 ],  
 );  
 });  
 }  
}

Ansalisando o código você irá verificar que usamos o Widget “**Observer**” para monitorar o estado da lista de produtos.

**Configurando o Main**

Após configurar todas as camandas da nossa aplicação, precisamos definir a função principal para iniciar todos os serviços da nossa aplicação. Para isso, na pasta “lib” altere o arquivo “main.dart”, digitando o seguinte código dentro dele:

**import 'package:app\_mobx/views/produto.view.dart'**;  
**import 'package:flutter/material.dart'**;  
**import 'package:provider/provider.dart'**;  
  
**import 'controllers/produto.controller.dart'**;  
  
**void** main() => runApp(MyApp());  
  
**class** MyApp **extends** StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 **return** MultiProvider(  
 providers:[  
 Provider<ProdutoController>.value(  
 value: ProdutoController(),)  
 ],  
 child: MaterialApp(  
 title: **'Lista'**,  
 debugShowCheckedModeBanner: **false**,  
 home: ProdutoListView(),  
 )  
 );  
 }  
}

Observe que usamos o recurso “Provider” para gerenciar o nosso controle, garantido que ele seja acessível em toda a aplicação sem precisar ficar gerando várias instâncias.

Agora você pode realizar teste e alterações na aplicação 😊

