

**Aplicativo para alterar o tema da aplicação**

**ICEI – PUC Minas**

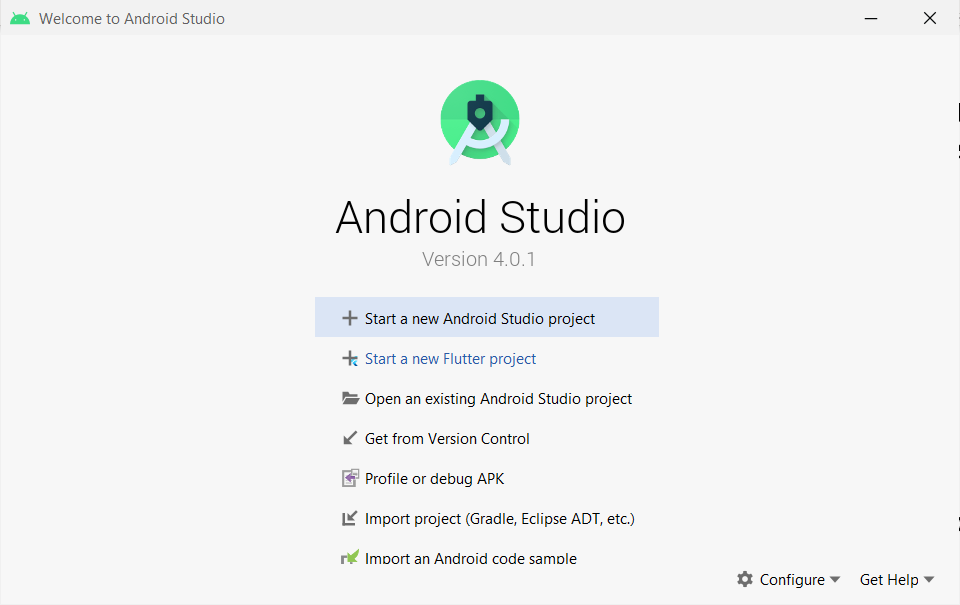
***SharedPreferences***

**Especificação do Aplicativo**

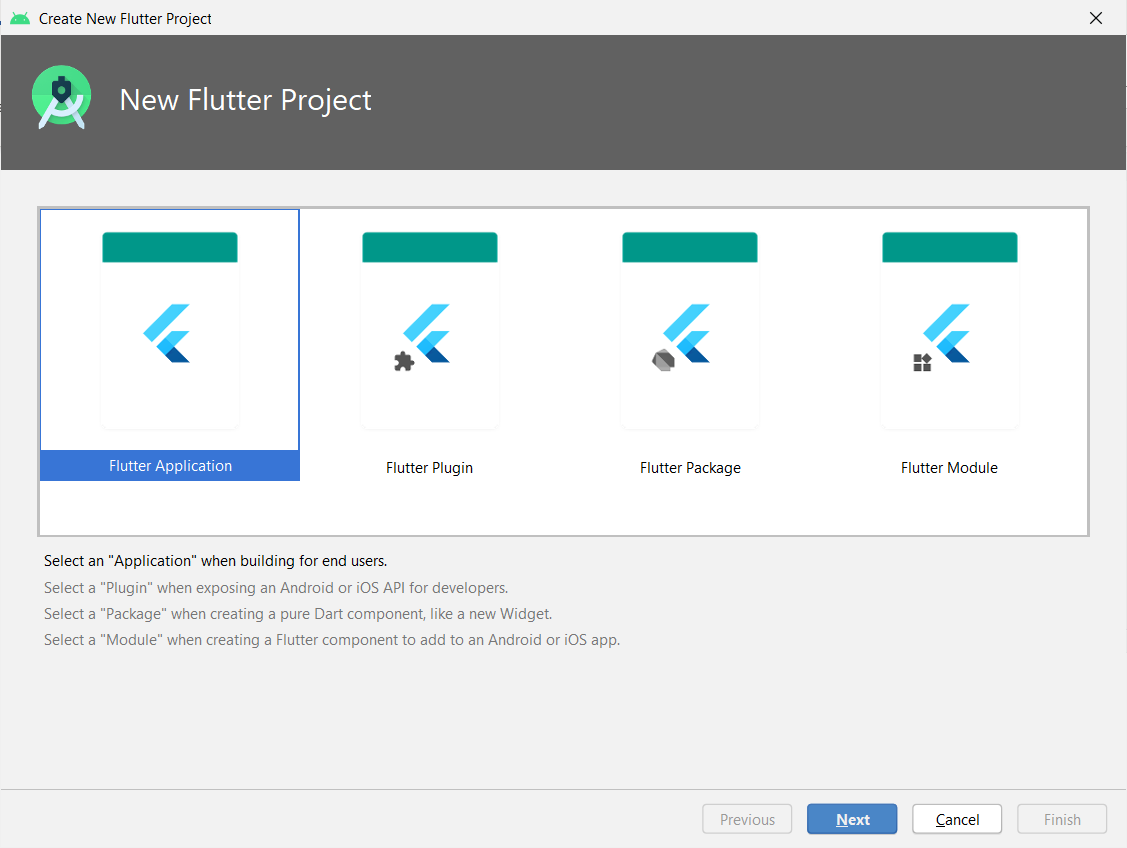
Nesta atividade iremos desenvolver um aplicativo que permite alterar o tema da aplicação e armazenar as alterações realizadas usando a persistência de dados com o *SharedPreferences.*

**Criando o Projeto**

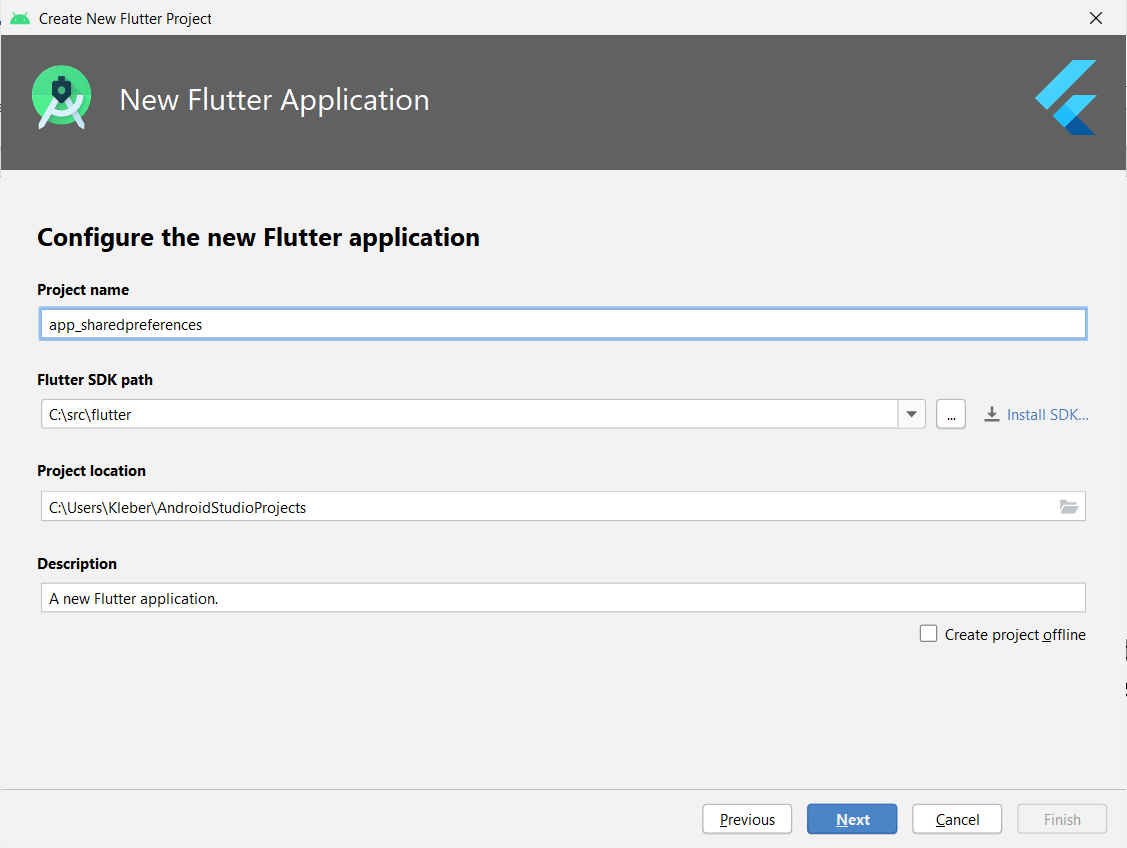
No Android Studio clique em “***Start a new Flutter project***”.

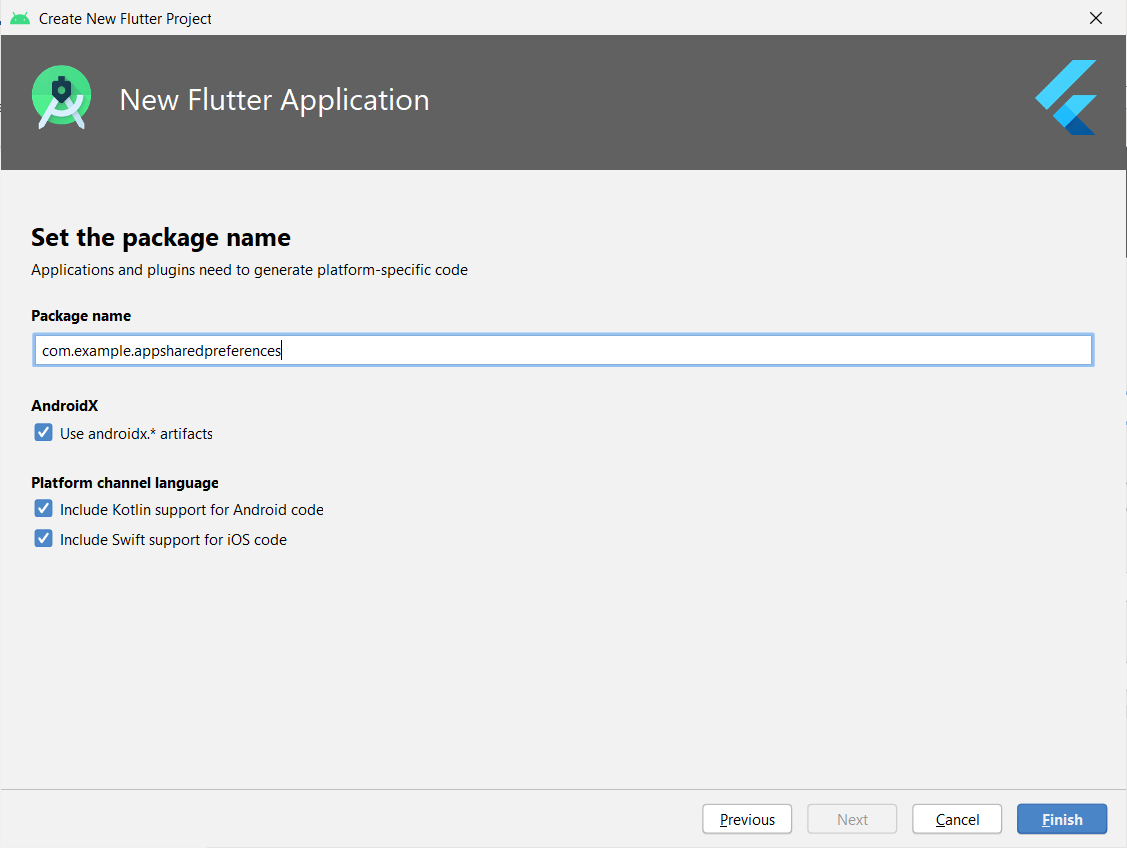


Depois selecione “***Flutter Application***” e clique em “***Next***”.



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





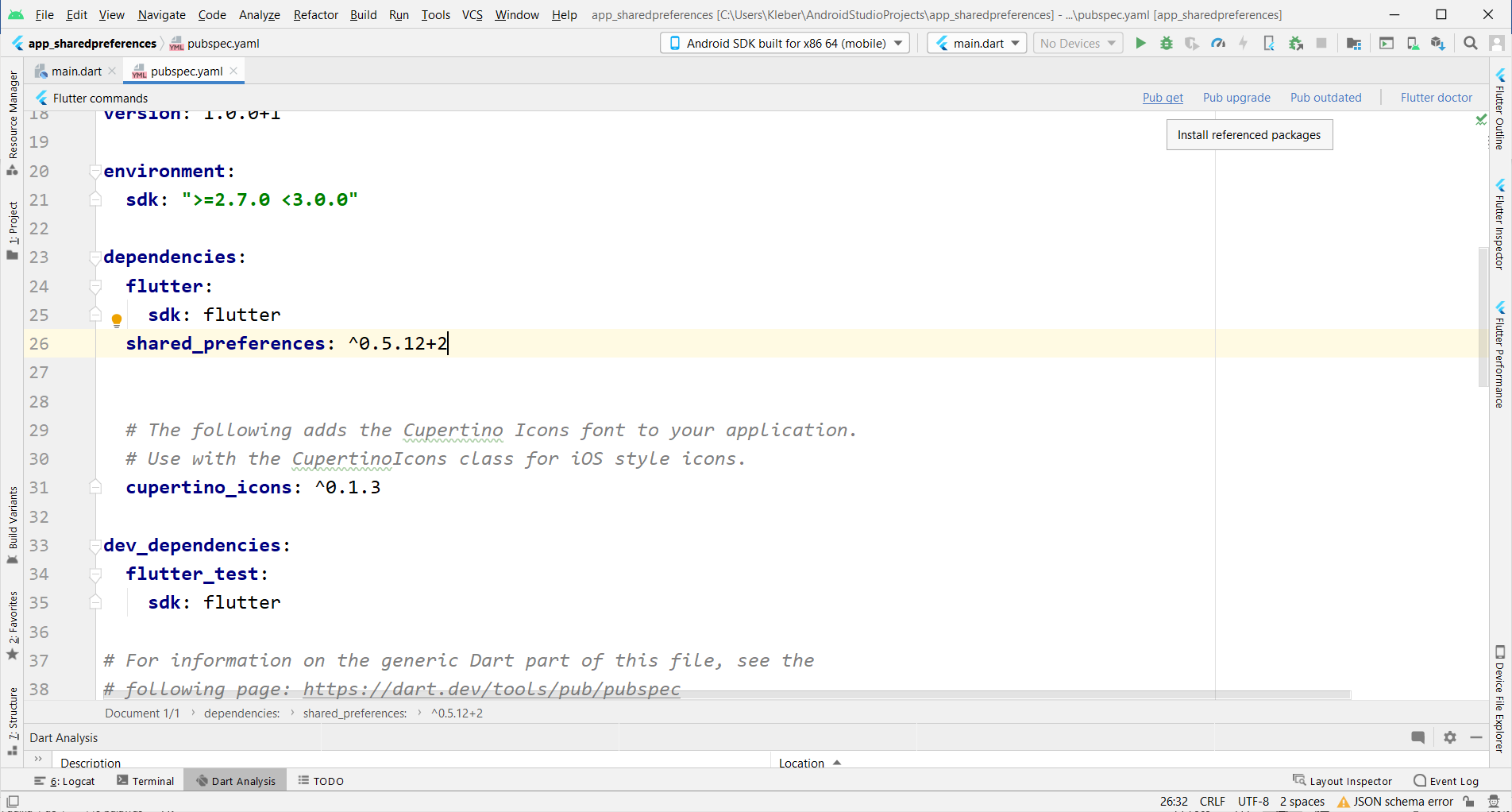
**Configurando o *SharePreferences***

Para usarmos o recurso *SharePreferences*, é necessário incluir a dependência no projeto. Acesse o site: <https://pub.dev/packages/shared_preferences> para usar a versão mais atual deste plugin.

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

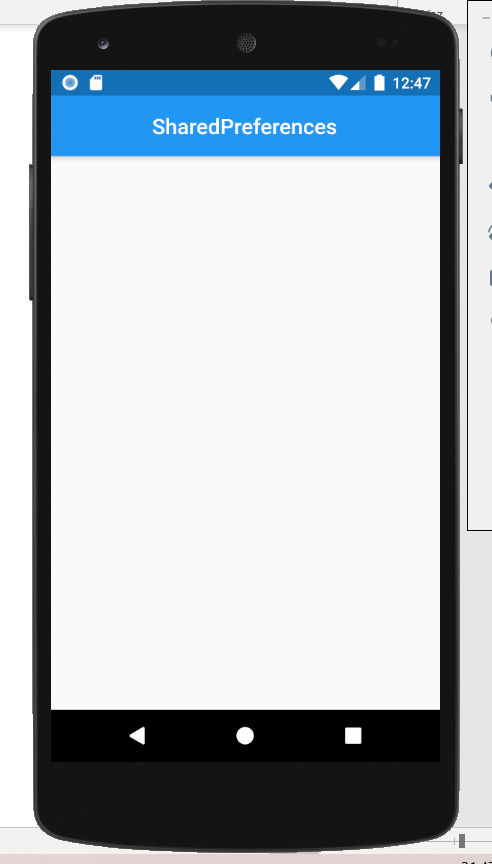
**dependencies**:  
 **flutter**:  
 **sdk**: flutter  
 **shared\_preferences**: ^0.5.12+2

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



**Iniciando o projeto**

Após criar e configurar as dependências o projeto, digite o código abaixo para iniciarmos nosso aplicativo:

**import 'package:flutter/material.dart'**;  
  
**void** main() {  
 runApp(MainApp());  
}  
  
**class** MainApp **extends** StatefulWidget {  
 @override   
 \_MainAppState createState() => \_MainAppState();  
}  
  
**class** \_MainAppState **extends** State<MainApp> {  
 @override  
 Widget build(BuildContext context) {  
 **return** MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text(**'SharedPreferences'**),  
 centerTitle: **true**,  
 ),  
 ),  
 );  
 }  
}

Iremos utilizar um Widget “MainApp” principal com estado, para que seja possível alterar o tema da aplicação de acordo com o desejo do usuário.

**Usando *SharePreferences***

Para usar o recurso de *SharePreferences* precisamos importar a referência em nosso arquivo. Digite no início do arquivo “main.dart”:

**import 'package:shared\_preferences/shared\_preferences.dart'**;

Após fazer a importação, implemente os métodos responsáveis por alterar o tema da aplicação usando o *SharePreferences* para persistir a alteração, após o aplicativo ser fechado. Digite o código na classe “\_**MainAppState**”.

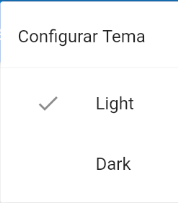
String **\_theme** = **'Light'**;  
**var \_themeData** = ThemeData.light();

@override  
**void** initState() {  
 **super**.initState();  
 \_loadTheme();  
}  
  
*// Carregando o tema salvo pelo usuário*\_loadTheme() **async** {  
 SharedPreferences prefs = **await** SharedPreferences.*getInstance*();  
 setState(() {  
 **\_theme** = (prefs.getString(**'theme'**) ?? **'Light'**);  
 **\_themeData** = **\_theme** == **'Dark'** ? ThemeData.dark() : ThemeData.light();  
 });  
}  
  
*// Carregando o tema salvo pelo usuário*\_setTheme(theme) **async** {  
 SharedPreferences prefs = **await** SharedPreferences.*getInstance*();  
 setState(() {  
 **\_theme** = theme;  
 **\_themeData** = theme == **'Dark'** ? ThemeData.dark() : ThemeData.light();  
 prefs.setString(**'theme'**, theme);  
 });  
}

O método “\_loadTheme” é responsável por abrir o recurso *SharedPreferences* e recuperar o tema armazenado anteriormente, caso não tenha sido definido ele assume o tema “Light” como padrão. Para usar este método ao iniciar o aplicativo, ele é invocado dentro do método “initState()” do Widget principal.

Já o método “\_setTheme” recebe o tema selecionado pelo usuário e faz a alteração tanto da variável “\_themeData” que armazena o tema a ser utilizado na aplicação quanto no recurso *SharedPreferences*, fazendo assim a persistência da informação.

**Desenvolvendo o Menu**

Para darmos a opção do usuário alterar o tema, crie um PopupMenuButton. Digite o método abaixo, responsável por criar e configurar o Widget PopupMenuButton. Este método cria um menu com um título superior e duas opções no formato de “CheckedPopupMenuItem”.

No método “onSelected” do “PopupMenuButton”,

configuramos para executar o método “setTheme”.

\_PopupMenuButton(){  
 **return** PopupMenuButton(  
 onSelected: (value) => \_setTheme(value) ,  
 itemBuilder: (context) {  
 **var** list = List<PopupMenuEntry<Object>>();  
 list.add(  
 PopupMenuItem(  
 child: Text(**"Configurar Tema"**)  
 ),  
 );  
 list.add(  
 PopupMenuDivider(  
 height: 10,  
 ),  
 );  
 list.add(  
 CheckedPopupMenuItem(  
 child: Text(**"Light"**),  
 value: **'Light'**,  
 checked: **\_theme** == **'Light'**,  
 ),  
 );  
 list.add(  
 CheckedPopupMenuItem(  
 child: Text(**"Dark"**),  
 value: **'Dark'**,  
 checked: **\_theme** == **'Dark'**,  
 ),  
 );  
 **return** list;  
 },  
 );  
}

Para finalizar nossa aplicação, basta chamar os métodos desenvolvidos no método “**build**” do Widget principal.

@override  
Widget build(BuildContext context) {  
 **return** MaterialApp(  
 debugShowCheckedModeBanner: **false**,  
 theme: **\_themeData**,  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text(**'SharedPreferences'**),  
 centerTitle: **true**,  
 actions: [  
 \_PopupMenuButton()  
 ],  
 ),  
 ),  
 );  
}

**Testando a aplicação**

Execute a aplicação e clique no menu para alterar o tema. Depois feche a aplicação e abra novamente e verifique se o tema permaneceu alterado 😊.

