

TUTORIAL FLUTTER COM FIREBASE

VERSÃO DO ANDROID STUDIO: ANDROID STUDIO JELLYFISH | 2023.3.1

VERSÃO DO FLUTTER: 3.19.6

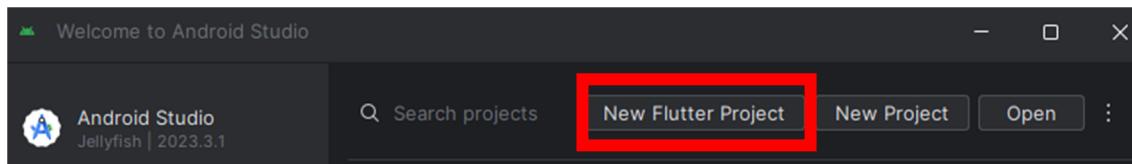
VERSÃO DO DART: 3.3.4

Link para o Firebase: <https://firebase.google.com/docs>

Link para o tutorial do FlutterFire:

<https://firebase.flutter.dev/docs/manual-installation/android>

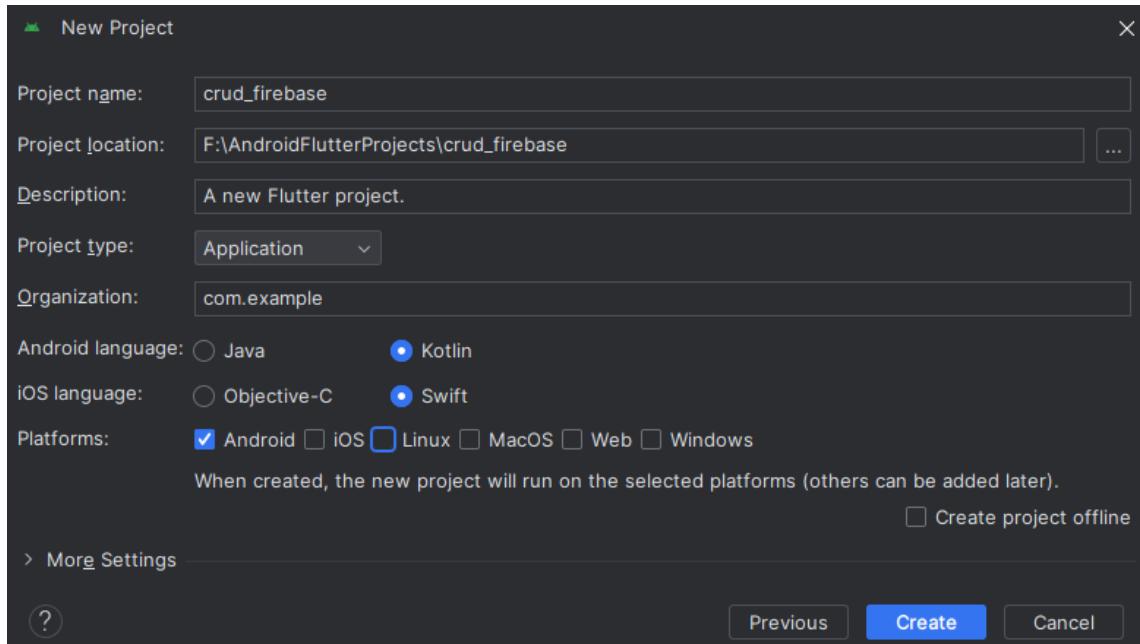
Crie um novo projeto Flutter no Android Studio:



Na próxima janela, certifique-se que o caminho (path) do Flutter SDK esteja indicado e clique em next:

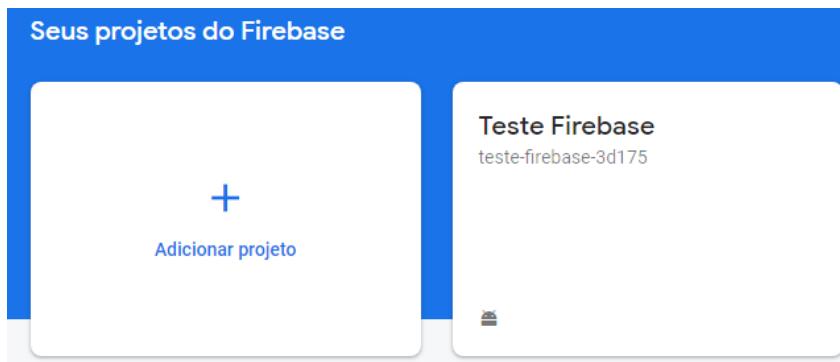


Agora você verá a tela de configuração do novo projeto, indique o nome (não pode conter letra inicial maiúscula, nem espaços em branco), selecione o local do projeto (Project Location), marque as plataformas desejadas e clique em "Create":



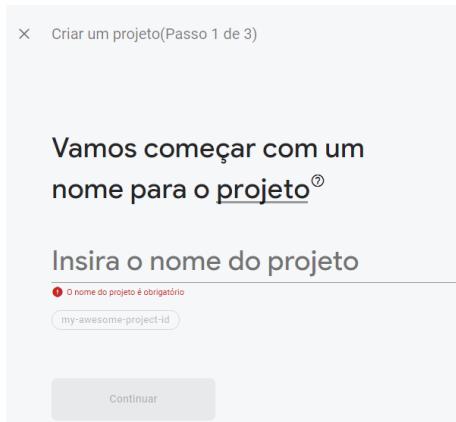
Após a criação do aplicativo, acesse o Firebase através do link <https://firebase.google.com> e crie uma conta, caso já possua, faça login.

Após o login, acesse o console e clique em **Adicionar Projeto**, conforme a imagem abaixo.



O Firebase irá instruir a criação do projeto:

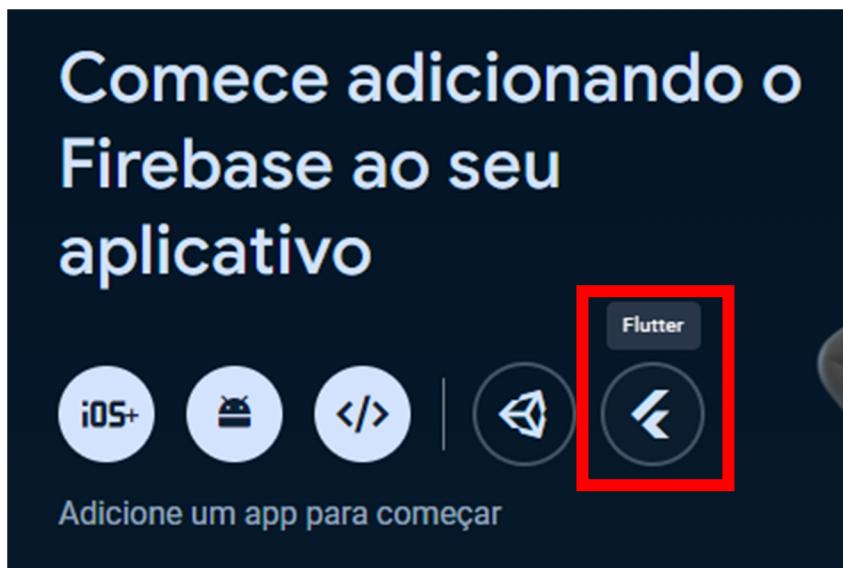
1- Inserir nome do Projeto e clicar em continuar;



2- Ativar o Analytics caso o projeto necessite. No caso deste tutorial, o Analytics será desativado, pois é apenas um teste. Logo em seguida clique em Criar projeto.

The screenshot shows the second step of the Firebase project creation process. At the top left, there's a back button labeled 'Voltar' and a help icon. The main title is 'Criar um projeto(Passo 2 de 2)'. Below it, the text reads 'Google Analytics para seu projeto do Firebase'. A descriptive paragraph explains that Google Analytics is a free, unlimited solution for analysis, mentioning products like Firebase Crashlytics, Cloud Messaging, App messaging, Remote Config, A/B Testing, Predictions, and Cloud Functions. A section titled 'O Google Analytics ativa:' lists several features: 'Teste A/B', 'Segmentação de usuários em produtos do Firebase', 'Previsão do comportamento de usuários', 'Usuários sem falhas', 'Gatilhos do Cloud Functions com base em eventos', and 'Geração de relatórios ilimitada gratuita'. A toggle switch is shown with the label 'Ativar o Google Analytics neste projeto' and the status 'Recomendado'. At the bottom, there are 'Anterior' and 'Criar projeto' buttons.

Após criar o projeto no Firebase, você precisará adicionar um app:



Escolha o Flutter, para utilizar o Firebase como um Framework e utilizar o CLI para configurá-lo no projeto (pode verificar a documentação aqui: <https://firebase.google.com/docs/flutter/setup?hl=pt-br&platform=android>).

Agora, irá aparecer a tela com instruções para adicionar o firebase ao app Flutter:

× Adicionar o Firebase ao app Flutter

1 Preparar seu espaço de trabalho

Usar a CLI do FlutterFire é o jeito mais fácil de começar.

Antes de continuar, certifique-se de:

- Instalar a [CLI do Firebase](#) e fazer login (execute `firebase login`)
- Instalar o [SDK do Flutter](#)
- Criar um projeto do Flutter (execute `flutter create`)

[Próxima](#)

2 Instalar e executar a CLI do FlutterFire

3 Inicializar o Firebase e adicionar plug-ins

Clique no link indicado para instalar a CLI do Firebase e siga suas instruções (a criação de um projeto já foi feita anteriormente, no início deste tutorial, e isso também já indica que a instalação do SDK do Flutter já foi feita).

Após instalar a CLI do Firebase, clique em Próxima para ir para o passo 2:

× Adicionar o Firebase ao app Flutter

Preparar seu espaço de trabalho

2 Instalar e executar a CLI do FlutterFire

Em qualquer diretório, execute o comando:

```
$ dart pub global activate flutterfire_cli
```

Em seguida, na raiz do diretório do seu projeto do Flutter, execute o comando:

```
$ flutterfire configure --project=crud-firebase-2d1f3
```

Com isso, seus apps são registrados automaticamente por plataforma com o Firebase, e um arquivo de configuração `lib.firebaseio_options.dart` é adicionado ao seu projeto do Flutter.

[Anterior](#)

[Próxima](#)

3 Inicializar o Firebase e adicionar plug-ins

Execute os comandos solicitados:

```
Administrator: Windows Pow X + - 
O Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

Instale o PowerShell mais recente para obter novos recursos e aprimoramentos! https://aka.ms/PSWindows

PS C:\Users\Tassiana> dart pub global activate flutterfire_cli
Package flutterfire_cli is currently active at version 1.0.0.
The package flutterfire_cli is already activated at newest available version.
To recompile executables, first run 'dart pub global deactivate flutterfire_cli'.
Installed executable flutterfire.
Activated flutterfire_cli 1.0.0.
PS C:\Users\Tassiana> F:
PS F:> cd .\AndroidFlutterProjects\
PS F:\AndroidFlutterProjects> cd .\crud_firebase\
PS F:\AndroidFlutterProjects\crud_firebase> flutterfire configure --project=crud-firebase-2d1f3
i Found 5 Firebase projects. Selecting project crud-firebase-2d1f3.
✓ Which platforms should your configuration support (use arrow keys & space to select)? · android
i Firebase android app com.example.crud_firebase is not registered on Firebase project crud-firebase-2d1f3.
i Registered a new Firebase android app on Firebase project crud-firebase-2d1f3.

Firebase configuration file lib.firebaseio_options.dart generated successfully with the following Firebase apps:

Platform  Firebase App Id
android   1:447757199866:android:56dbb0e8ec2ad7d1675c43

Learn more about using this file and next steps from the documentation:
> https://firebase.google.com/docs/flutter/setup
PS F:\AndroidFlutterProjects\crud_firebase> |
```

Ao executar o comando `flutterfire`, pode ocorrer o seguinte erro:

Warning: Pub installs executables into

C:\Users\PC\AppData\Local\Pub\Cache\bin, which is not on your path. You can fix that by adding that directory to your system's "Path" environment variable. A web search for "configure windows path" will show you how.

This means you need to add C:\Users*username*\AppData\Local\Pub\Cache\bin into your System's environment path. It would be better if you restart the computer after adding it to the path variable.

Isso significa que você precisa adicionar C:\Users*SEU_USUARIO*\AppData\Local\Pub\Cache\bin ao caminho do ambiente do seu sistema. Feche o terminal, abra novamente, e digite o comando.

Volte ao navegador para continuar a configuração do Firebase, no passo 2, clique em próximo, e irá abrir as configurações do passo 3:

Adicionar o Firebase ao app Flutter

- Preparar seu espaço de trabalho
- Instalar e executar a CLI do FlutterFire
- Inicializar o Firebase e adicionar plug-ins

Para inicializar o Firebase, chame `Firebase.initializeApp` no pacote `firebase_core` com a configuração do seu novo arquivo `firebase_options.dart`:

```
import 'package:firebase_core/firebase_core.dart';
import 'firebase_options.dart';

// ...

await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);
```

Depois adicione e comece a usar os [plug-ins do Flutter](#) para os produtos do Firebase que quiser.

Observação: se usar o Analytics ou o Monitoramento de desempenho, talvez você tenha que seguir algumas etapas adicionais de configuração.

[Anterior](#) [Continuar no console](#)

Vamos seguir as instruções do passo 3, que agora devem ser inseridas dentro do seu projeto. O arquivo “`firebase_options.dart`” foi criado automaticamente pelo CLI, portanto, agora é preciso adicionar as dependências do firebase ao projeto.

Acesse o arquivo `pubspec.yaml`, e adicione as dependências que você irá utilizar no projeto. Acesse <https://pub.dev> para verificar as versões atuais dos serviços: `cloud_firestore` e `firebase_core`, os quais **devem ser incluídos** no `pubspec.yaml`, para fazermos um teste de conexão com o banco.

Outros serviços podem ser adicionados também. Alguns exemplos de serviços e os link para saberem mais sobre cada um:

`firebase_core: # Recursos Básicos`

https://pub.dev/packages/firebase_core/install

`cloud_firestore: # Banco de Dados`

https://pub.dev/packages/cloud_firestore/install

`firebase_storage: # Armazenamento de Arquivos`

https://pub.dev/packages/firebase_storage/install

`firebase_auth: # Autenticação de Usuários`

https://pub.dev/packages/firebase_auth/install

Para incluir os serviços que utilizamos neste tutorial exemplo, pesquise as dependências indicadas (`firebase_core` e `cloud_firestore`) no site pub.dev, clique na dependência e vá para a aba “`Installing`”:

firebase_core 2.30.1

Published 8 days ago • firebase.google.com Dart 3 compatible

SDK FLUTTER PLATFORM ANDROID IOS MACOS WEB WINDOWS

3.3K

Readme Changelog Example **Installing** Versions Scores

Use this package as a library

Depend on it

Run this command:

With Flutter:

```
$ flutter pub add firebase_core
```

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

```
dependencies:  
  firebase_core: ^2.30.1
```

Você pode abrir o terminal dentro do Android Studio e rodar o comando indicado na imagem: Flutter pub add firebase_core, ou apenas copiar firebase_core: ^2.30.1 e inserir dentro de **dependencies**.

Após adicionar as duas dependências, seu pubspec.yaml ficará desta forma:

```
22  sdk: '>=3.3.4 <4.0.0'
23
24  # Dependencies specify other packages that
25  # To automatically upgrade your package do
26  # consider running `flutter pub upgrade` -
27  # dependencies can be manually updated by
28  # the latest version available on pub.dev
29  # versions available, run `flutter pub ou
30  dependencies:
31    flutter:
32      sdk: flutter
33
34
35  # The following adds the Cupertino Icons
36  # Use with the CupertinoIcons class for
37  cupertino_icons: ^1.0.6
38  | firebase_core: ^2.30.1
39  | cloud_firestore: ^4.17.2
40
41  dev_dependencies:
42    flutter_test:
43      sdk: flutter
44
45  # The "flutter_lints" package below can
```

Ainda no pubspec.yaml, clique em **Pub get**, no canto superior direito para que ele baixe e instale estas dependências, deixando-as prontas para utilização:

The Pub get button is highlighted in red.

Sempre que for utilizar as dependências instaladas, importar as bibliotecas que irá utilizar, como por exemplo:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_core/firebase_core.dart';
```

Agora, para podermos utilizar o Firebase, precisamos inicializá-lo, mas antes disso, precisamos acessar o arquivo main.dart e modificar o void main() para **Future<void> main() async**, e dentro da função **Future<void> main() async**, inicializamos o firebase inserindo os seguinte códigos (um deles é o `Firebase.initializeApp` que está no passo 3 da configuração do Firebase):

```
WidgetsFlutterBinding.ensureInitialized();

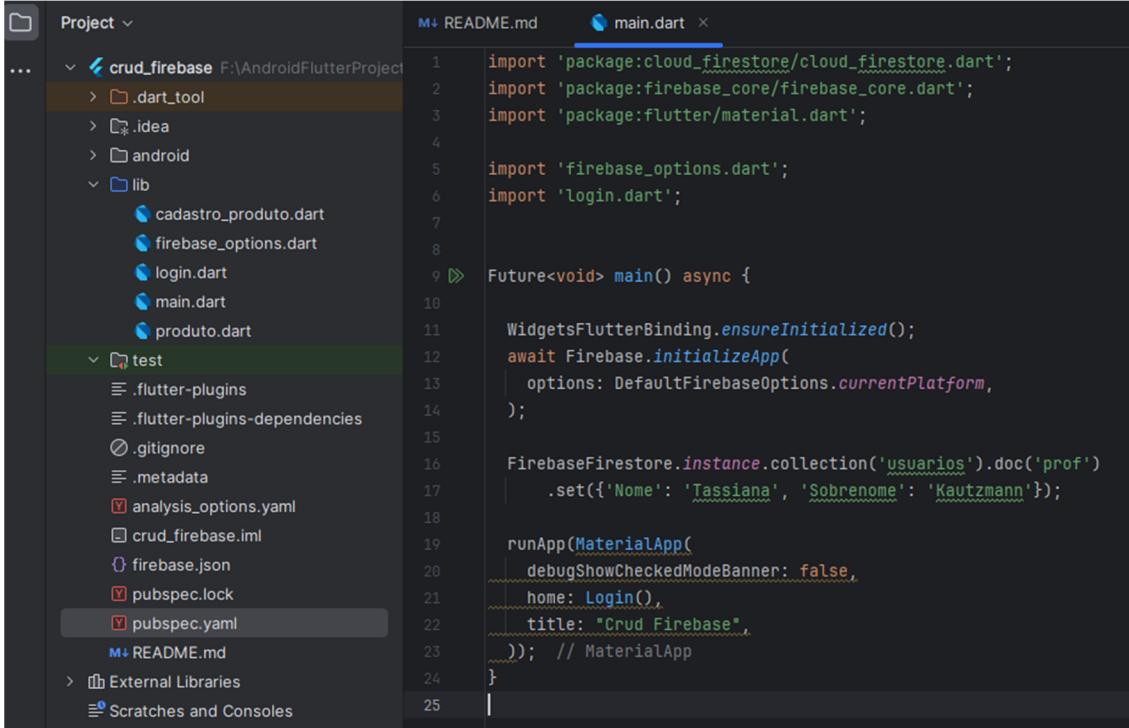
await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
);


```

Agora para podermos testar se a aplicação se conectou ao banco de dados, podemos fazer um teste, inserindo um código (dentro do **Future<void> main() async**) para a criação de uma coleção de dados no banco:

```
FirebaseFirestore.instance.collection('usuarios').doc('prof').set({'Nome': 'Tassiana', 'Sobrenome': 'Kautzmann'});
```

O “main.dart” ficará assim:



The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The project structure includes files like .gitignore, analysis_options.yaml, crud_firebase.iml, firebase.json, lib (containing cadastro_produto.dart, firebase_options.dart, login.dart, main.dart, and produto.dart), pubspec.lock, pubspec.yaml, README.md, and test (containing flutter-plugins, flutter-plugins-dependencies, .gitignore, .metadata, and analysis_options.yaml). The code editor displays the main.dart file with the following content:

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'firebase_options.dart';
import 'login.dart';

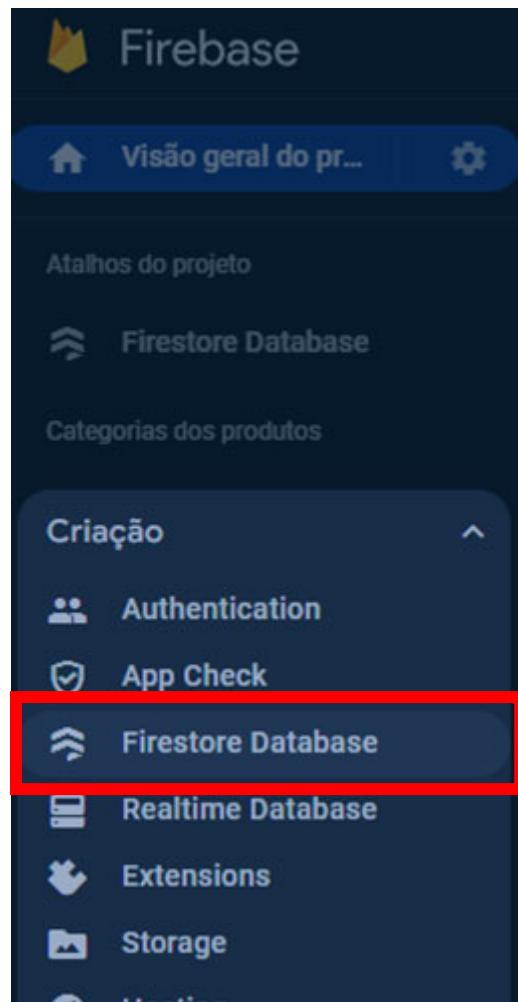
Future<void> main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp(
        options: DefaultFirebaseOptions.currentPlatform,
    );

    FirebaseFirestore.instance.collection('usuarios').doc('prof')
        .set({'Nome': 'Tassiana', 'Sobrenome': 'Kautzmann'});

    runApp(MaterialApp(
        debugShowCheckedModeBanner: false,
        home: Login(),
        title: "Crud Firebase",
    )); // MaterialApp
}
```

Volte ao site do Firebase, e no passo 3, clique em “Continuar no console”.

Acesse o Firestore Database no menu lateral:



E clique em Criar banco de dados:

Cloud Firestore

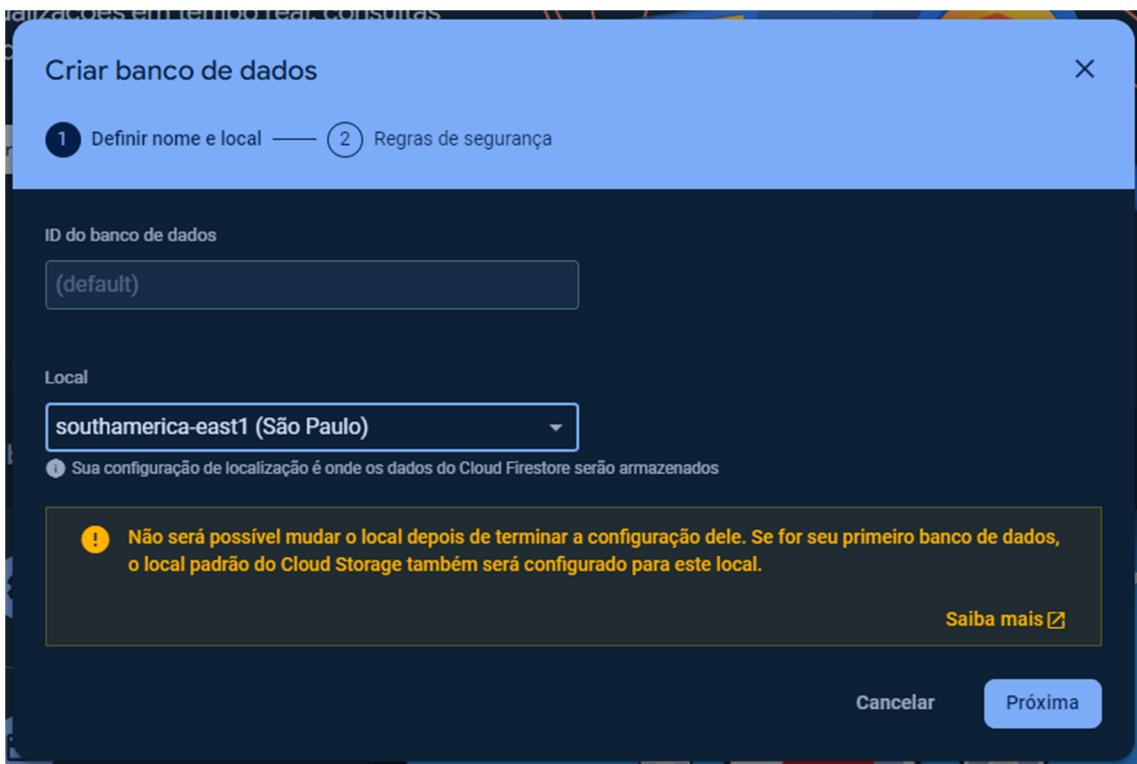
Atualizações em tempo real, consultas eficientes e escalonamento automático

Criar banco de dados

Isso fará com que abra a tela do Cloud Firestore, e você pode clicar em “Criar banco de dados”:



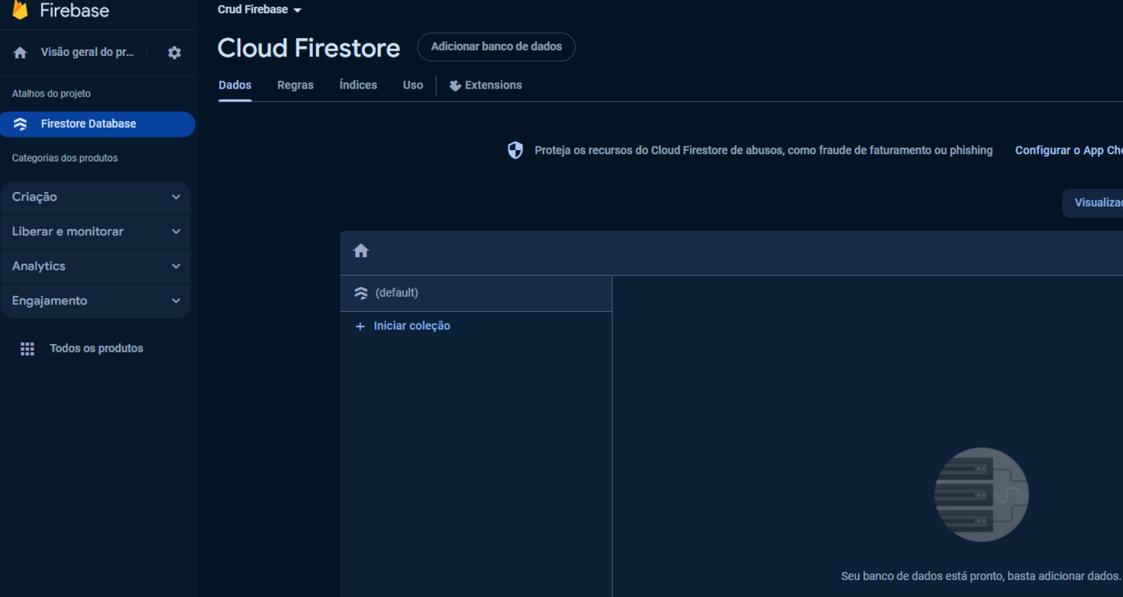
Ao clicar, um pop-up irá aparecer para começar a configurar o banco, selecione o servidor “southamerica-east1(São Paulo), conforme imagem abaixo e clique em Próxima.



Agora, selecione “Iniciar no modo de teste” e clique em “Criar”:



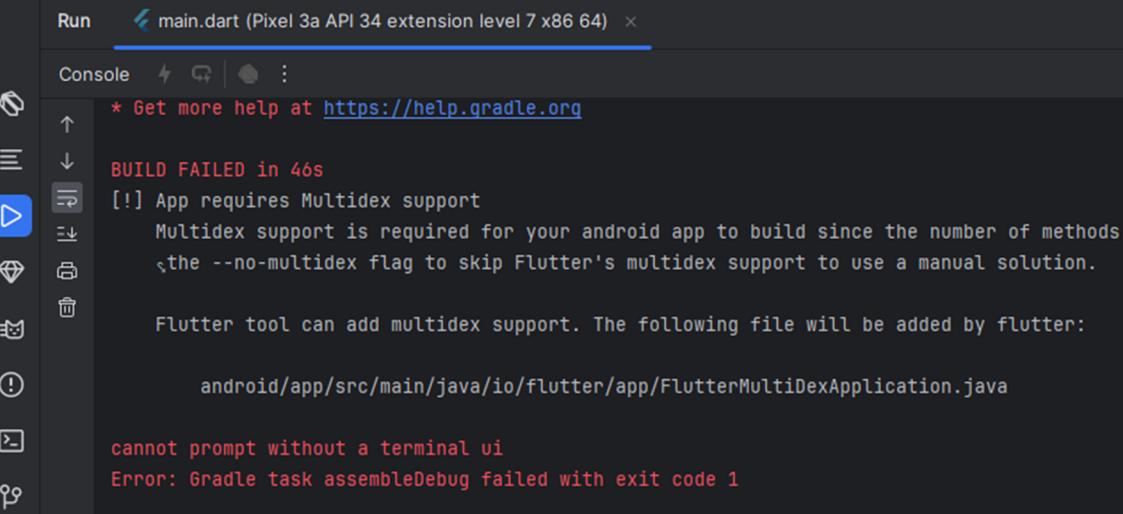
Agora seu banco de dados Cloud Firestore está configurado e pronto para ser utilizado:



The screenshot shows the Firebase Cloud Firestore interface. On the left, there's a sidebar with project settings like 'Visão geral do projeto' and 'Atalhos do projeto'. The main area is titled 'Cloud Firestore' with tabs for 'Dados', 'Regras', 'Índices', 'Uso', and 'Extensions'. A banner at the top right says 'Proteja os recursos do Cloud Firestore de abusos, como fraude de faturamento ou phishing' and 'Configurar o App Check'. Below the banner, there's a button for 'Visualização'. The central part of the screen shows a collection named '(default)' with a sub-item '+ Iniciar coleção'. At the bottom right, there's a circular icon with a server-like pattern and the text 'Seu banco de dados está pronto, basta adicionar dados.'

Volte ao Android Studio e execute o aplicativo criado.

Provavelmente, ao executar, aparecerá o seguinte erro:



The screenshot shows the Android Studio 'Run' tab with the 'main.dart' file selected. The 'Console' tab is active, displaying the following error message:

```
* Get more help at https://help.gradle.org
BUILD FAILED in 46s
[!] App requires Multidex support
    Multidex support is required for your android app to build since the number of methods
    <the --no-multidex flag to skip Flutter's multidex support to use a manual solution.

    Flutter tool can add multidex support. The following file will be added by flutter:

        android/app/src/main/java/io/flutter/app/FlutterMultiDexApplication.java

    cannot prompt without a terminal ui
Error: Gradle task assembleDebug failed with exit code 1
```

O app precisa de suporte Multidex. O Multidex é uma técnica usada em aplicativos Android para contornar a limitação do número máximo de métodos que um aplicativo pode ter.

Ele pode ser necessário se o seu aplicativo tem muitas dependências e bibliotecas, o que pode aumentar significativamente o número de métodos no aplicativo final.

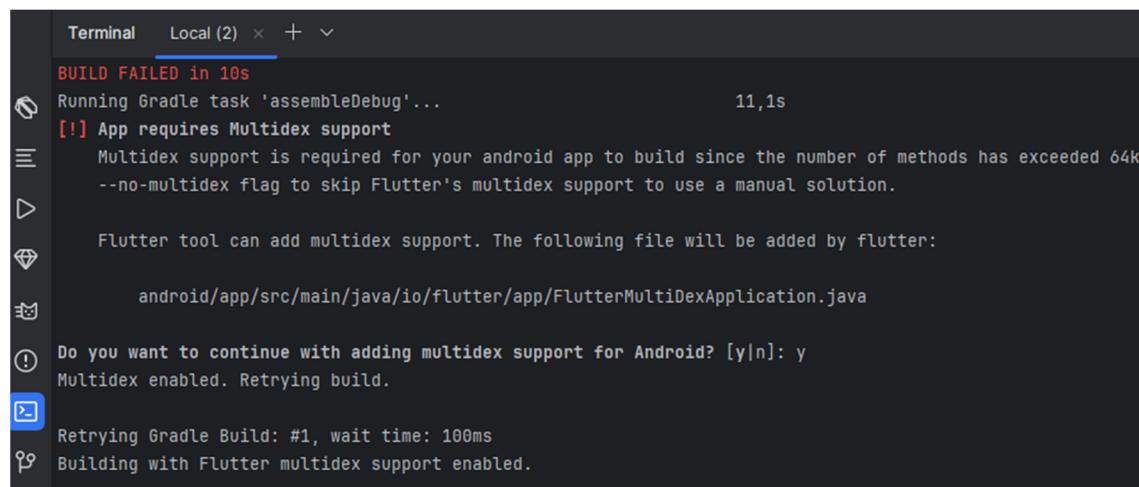
Para resolver (veja detalhes em

<https://docs.flutter.dev/deployment/android#enable-multidex-support>),

vamos executar o seguinte comando:

```
flutter run --debug
```

Isso fará com que o aplicativo execute novamente, porém, ao aparecer o erro, aparecerá uma mensagem perguntando se você deseja continuar e adicionar suporte ao MultiDex, digite Y e ele irá continuar e seu projeto irá rodar:



```
Terminal Local (2) × + ✓
BUILD FAILED in 10s
Running Gradle task 'assembleDebug'... 11,1s
[!] App requires Multidex support
Multidex support is required for your android app to build since the number of methods has exceeded 64k
--no-multidex flag to skip Flutter's multidex support to use a manual solution.

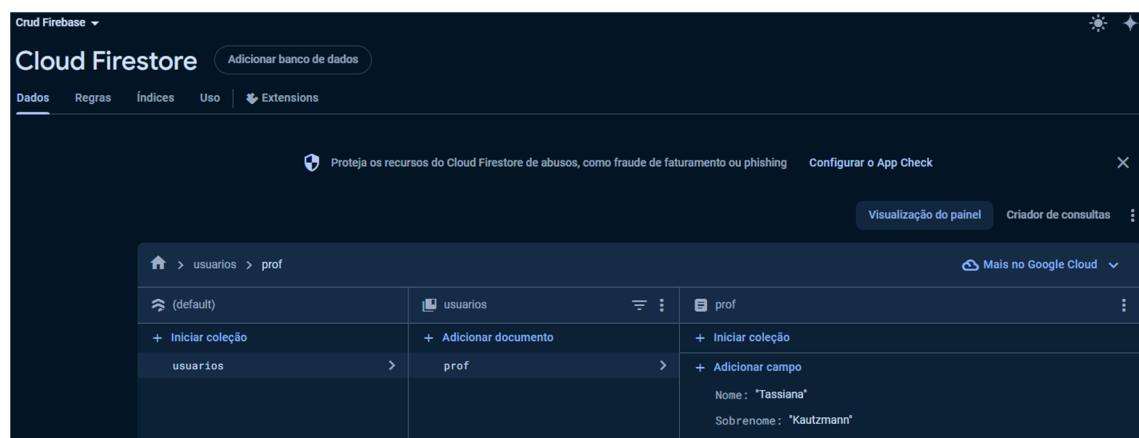
Flutter tool can add multidex support. The following file will be added by flutter:

    android/app/src/main/java/io/flutter/app/FlutterMultiDexApplication.java

Do you want to continue with adding multidex support for Android? [y|n]: y
Multidex enabled. Retrying build.

Retrying Gradle Build: #1, wait time: 100ms
Building with Flutter multidex support enabled.
```

Após o projeto aparecer no emulador, volte para o navegador, atualize a página do Cloud Firestore, e veja que a coleção foi criada no banco:



Cloud Firestore Adicionar banco de dados

Dados Regras Índices Uso Extensions

Proteja os recursos do Cloud Firestore de abusos, como fraude de faturamento ou phishing Configurar o App Check

Visualização do painel Criador de consultas

Mais no Google Cloud

(default) usuarios prof

+ Iniciar coleção	+ Adicionar documento	+ Iniciar coleção
usuarios >	prof >	Nome: "Tassiana" Sobrenome: "Kautzmann"

Agora seu aplicativo Flutter está pronto para executar operações com o banco de dados Firebase.

POR QUE UTILIZAMOS O FUTURE, ASYNC E AWAIT?

Processar recursos externos leva um certo tempo, e o app não pode parar de executar até que estes recursos sejam processados. Então utilizamos o conceito de programação assíncrona, onde o app continua executando diversas tarefas em paralelo.

Segundo Pinheiro (2022):

“O **async** determina que um método será assíncrono, ou seja, não irá retornar algo imediatamente, então o aplicativo pode continuar a execução de outras tarefas enquanto o processamento não é finalizado.

O **await** serve para determinar que o aplicativo deve esperar uma resposta de uma função antes de continuar a execução. Isso é muito importante pois há casos em que uma função depende do retorno de outra.

Já o **Future** determina que uma função irá retornar algo no “futuro”, ou seja, é uma função que levará um tempo até ser finalizada.”