Guia Firebase

Guia feito com referência na criação de um app de mensagens

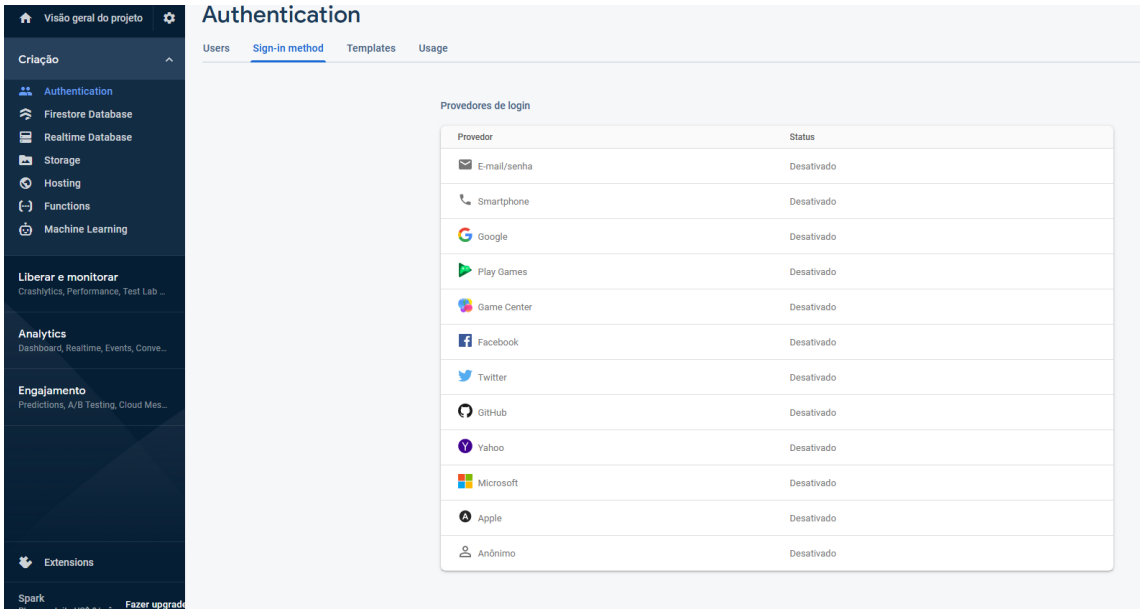
1 O que é o Firebase?

Firebase é uma plataforma de desenvolvimento da **Google** que fornece uma série de ferramentas para a comunicação do seu aplicativo com serviços externos, como *Cloud Storage, Authenticator, Analytics, etc.*

2 Criando e configurando um projeto no Firebase

Entre no site https://firebase.google.com, faça Login com sua conta Google e clique em "Ir para Console" e logo em seguida, "Criar novo projeto". Após a criação do projeto, você será redirecionado para a tela de controle dos dados do seu aplicativo.

Com o seu projeto já criado, clique na aba "Criação" e configure o que achar necessário, por exemplo, métodos de Login que serão realizados pelo aplicativo.



Authentication

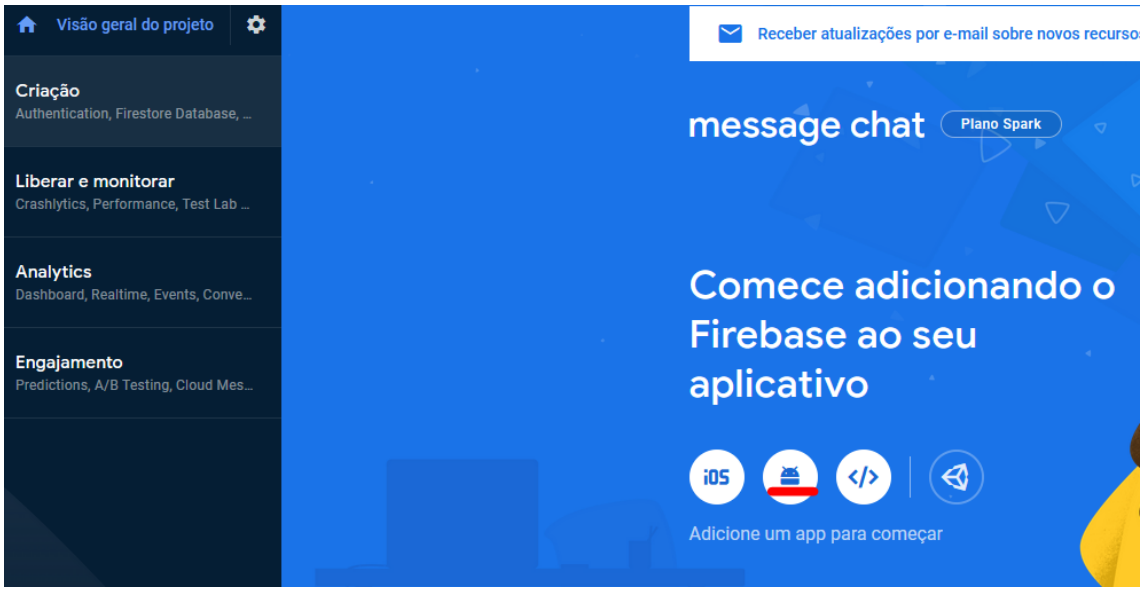
Abaixo dessa opção, clique em "Database" e crie um banco de dados.Nessa etapa de criação de um BD tem dois modos de proteção, o modo produção, que impossibilita tanto escrita quanto leitura no BD, e o modo teste, que permite escrita até um certo período e depois se torna um BD no modo produção.

2.1 Entendendo o Banco de Dados

O banco de dados é formado por coleções, para cada coleção exitem uma lista de outros objetos (documentos) pertencentes à mesma coleção.Por exemplo, ao criar uma coleção "Mensagens",esta estará agrupando todos os objetos que forem uma mensagem, de modo análogo seria uma coleção "Usuários".Cada documento possui os seus respectivos dados (campos) e também podem armazenar outra coleção dentro deles.

2.2 Integrando o Firebase ao projeto no Flutter

Feitas as configurações iniciais do projeto, precisamos integrar ele ao Flutter, para isso, basta clicar na página inicial do projeto no Firebase e clicar no botão referente ao SO, no caso, Android.



Integração ao Flutter

Isso irá abrir uma tela pedindo o *Package name* do aplicativo, que pode ser encontrado no arquivo "AndroidManifest.xml".

No campo que solicita o certificado de assinatura de depuração SHA-1 (necessário caso for realizar Login com o Google), é necessario achar a pasta do Java Runtime Enviroment (JRE) no computador. Se o android studio estiver instalado, basta procurar na pasta dele.Abrindo a pasta jre, entre em bin e procure por **keytool**, se estiver na pasta, abra o CMD no caminho dela e execute o comando "keytool -list -v -keystore "%USERPROFILE%àndroid\debug.keystore" -alias androiddebugkey -storepass android -keypass android " feito isso, basta copiar a chave

e registar o app.Realize o download do arquivo "google-services.json" em sequência, como é pedido, e coloque no local especificado.Após a realização do download e com o arquivo já na pasta do projeto, colocar as dependências, do mesmo modo que é pedido no próprio site.Clique em próximo e configurar o projeto, então tudo está pronto.

Agora, é necessário adicionar os plugins que serão utilizados, no caso, ***cloud_firestore, image_picker, google_sign_in, firebase_storage e firebase_auth*** nas versoes mais recentes.

2.3 Testando o Firebase e resolvendo possíveis erros

Para testar se o app consegue se comunicar com o Firebase, adicione a seguinte package "package:cloud_firestore/cloud_firestore.dart" e o seguinte código após o runApp(),
"FirebaseFirestore.instance
.collection("col")
.doc("doc")
.set({"texto": "teste"});"

e caso dê o erro *The number of method references in a .dex file cannot exceed 64K.* entre em [app][src][build.gradle] e digite "multiDexEnabled true" abaixo de "minSdkVersion".Caso ocorra um erro de Autenticação do Firestore/Firebase, pode ser que seja a localização do BD ao criar o projeto no site.

3 Conceitos do Firebase

3.1 Escrever dados

Para escrever dados no Firebase é utilizado o comando Firestore.instance, através dele, especificamos uma coleção e um documento para ser escrito no BD, caso a coleção ou documento fornecido já exista, ele irá reescrever os valores dentro do objeto, caso não exista, ele irá criar um.Por exemplo, se quisermos criar uma coleção "Mensagens" e colocar, dentro dela, um documento "doc1", seria:

Firestore.instance.collection("Mensagens").document("doc1").setData({Map()});

onde o setData({Map()}) irá escrever as informações lá dentro.Caso o documento já exista e só precisemos atualizar/criar um dado específico, basta trocar o setData por updateData.Há também a possibilidade de adicionar um documento diretamente, pelo comando:

Firestore.instance.collection("col").addData(data)

desse modo, o documento é criado e adicionado com o dado especificado em addData().
OBS: Caso necessite que cada objeto tenha uma "key" específica, basta não passar nenhum parâmetro dentro do document, desse modo, cada documento dentro de uma coleção terá um ID único

3.2 Ler dados

A leitura de dados pode ser feita de duas maneiras, podem ser lidos dados uma única vez, ou pode-se adicionar um listener que irá atualizar sempre que um dado for modificado.

3.3 Lendo dados uma única vez

Para ler dados uma única vez, pode ser feito o comando

Firestore.instance.collection("name").getDocuments()

que será armazenado em uma variável do tipo **QuerySnapshot** e,nesse caso, precisamos solicitar um objeto Future, pois leva um tempo para carregar as informações.

```
QuerySnapshot dados = await
Firestore.instance.collection("name").getDocuments();
dados.documents.forEach((document){
print(document.data); //Imprime as informações dentro do documento
}
)
```

Para um documento apenas, o tipo da variável seria **DocumentSnapshot** e o método a ser chamado seria

Firestore.instance.collection("name").document("ID").get()

O ID do documento pode ser obtido também através do atributo .documentID

Atualizando elementos dentro dos documentos

Para atualizar/criar um elemento dentro de um documento, podemos utilizar a seguinte expressão:

docuement.reference.updateData("data": "value")

4 Lendo documentos em tempo real

Caso queira receber a informação dentro de um documento ou coleção sempre que esta receber uma modificação, é necessário utilizar o método snapshot().listen((value) //Function//).Por exemplo, quero receber um print dos dados dentro da coleção "Mensagens" sempre que um dele sofrer uma modificação:

```
Firestore.instance.collection("Mensagens").snapshot().listen( ( data ){
data.documents.forEach( (d){
print(d.data) } )
} )
```

Isso pode ser feito com objetos document() da mesma maneira que os da collection().

5 Enviando imagens para o Firebase

Para enviar imagens é necessário, primeiramente, criar um FirebaseStorage, no próprio site do Firebase, abaixo do BD.Após criar o Storage, deve ser criado um Objeto do tipo StorageUploadTask, que será responsável por adicionar a imagem ao Storage.Ao instanciar esse objeto, ele vai receber outro objeto da classe FirebaseStorage, e para que possamos criar um link entre o FirebaseStorage e a imagem que está sendo enviada, devemos usar o comando:

FirebaseStorage.instance.ref().child("NOME DO ARQUIVO").putFile("ARQUIVO")

onde o ref() é para pegar a referência do Storage criado, o child() se refere ao nome do arquivo/pasta (esse método pode ser encadeado, ex: child("pasta1").child("nome do arquivo")), e o putFile() armazena o arquivo que será enviado.

Precisamos então criar um objeto do tipo StorageTaskSnapshot que armazenará informações da nossa imagem, um deles é o URL para download.Ele receberá um objeto da classe StorageUploadTask utilizando o método onComplete() que retorna um objeto Future¡¿, portanto a função deve ser **async**.

StorageTaskSnapshot info = **await** (StorageUploadTask).onComplete

e para obter o URL da imagem enviada, utilizamos o comando:

String url = **await** info.ref.getDownloadURL()

Inserindo a função no botão de foto

Para que o envio de imagens funcione, é preciso adicionar uma função que busque a imagem no aparelho,isso é feito através do seguinte código no atributo onPressed() do botão:

final PickedFile img = **await** ImagePicker().getImage(source: ImageSource.camera ou ImageSourcer.gallery) File imgFile = File(img.path);

o processo é assíncrono pois deve-se esperar a escolha da imagem. O método getImage() retorna um Future¡PickedImage¿, mas para adicionar ao Storage deve ser armazenada na forma de File, por isso, adicionamos a linha File(img.path).

5.1 Implementação do Login com o Google

Primeiro, criaremos um objeto do tipo **final GoogleSignIn signin = Gooogle-SignIn()** e, em uma função separada, realizar a tentativa de Login (será feita em um bloco try-catch).Devemos pegar o Login feito com o Google e depois autenticar no Firebase.

Para realizar o Login através do Google, cria-se um objeto GoogleSignInAccount que irá pegar a conta que está sendo logada e então jogar essas informações em uma autenticação, pelo objeto GoogleSignInAuthentication, feito isso, agora precisamos fornecer ao objeto da classe abstrata AuthCredential as credenciais validadas pelo autenticador da Google para realizarmos a conexão com o Firebase, por fim, relizamos a conexão com o Firebase através do objeto AuthResult.O código abaixo mostra como fazer:

final GoogleSignInAccount accountGoogle = await googleSignIn.signIn();
final GoogleSignInAuthentication authenticationGoogle = await accountGoogle.authentication;
final AuthCredential credentialGoogle = GoogleAuthProvider.getCredential(
idToken: authenticationGoogle.idToken,
accessToken: authenticationGoogle.accessToken);
final AuthResult firebaseAuth = await FirebaseAuth.instance.signInWithCredential(credentialGoogle);
final FirebaseUser user = firebaseAuth.user;

e através da última linha, você valida o acesso ao Firebase e o acesso também ao BD.

Verificando se o usuário está logado

Para verificar, precisamos sobrescrever o método initState() da nossa classe e dentro dele colocar a linha abaixo:

FirebaseAuth.instance.onAuthStateChanged.listen((user) { _currentUser = user; });

e o atributo onAuthStateChanged é um stream, podendo ser adicionado a ele o método listen(), desse modo, sempre que houver uma troca de usuários a variável(global) _currentUser será atualizada.Agora, na mesma função com o bloco try-catch, retornamos o usuário atual, colocando o tipo de retorno da função como Future¡FirebaseUser¿.

6 Regras do Firebase

As regras no Firebase são uma forma de limitar quem pode ler, escrever, modificar, etc os dados no BD, um exemplo do código utilizado no app de mensagens abaixo:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read: if true;
      allow create: if request.auth.uid != null;
      allow update: if request.auth.uid == resource.data.uid;
    }
  }
}
```

Regras do Firebase