

# Firestore Storage



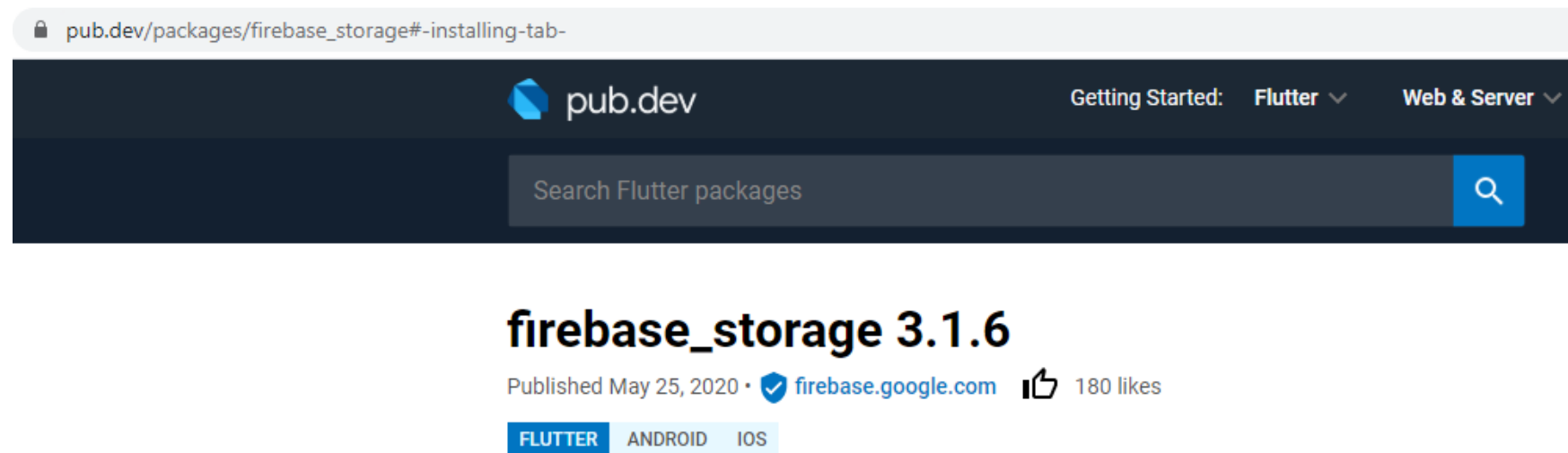
# Firestore Storage

- É o serviço de armazenamento de arquivos do Firestore.
- Podemos enviar imagens, áudios, vídeos e quaisquer tipos de arquivos para o Firestore Storage.
- No **FIIs Plan** utilizamos o Firestore Storage para armazenar o(s) arquivo(s) de banco de dados, bem como as imagens utilizadas.
- Imaginemos que o usuário do aplicativo resolva trocar de celular, como os dados são armazenados internamente no aplicativo, eles seriam perdidos. A ideia é ter rotinas de Backup e Restauração para que o usuário possa enviar seus dados para nuvem e recuperá-los de forma simples e rápida.



# Instalação do Plugin

- Precisaremos instalar o plugin do Firebase Storage para podermos gerenciar nossos arquivos usando Firebase.
- Esse plugin, assim como os demais, pode ser encontrado no site **pub.dev**.



# Console do Firebase Storage

- Assim como os outros serviços do Firebase, o Firebase Storage pode ser configurado a partir do console.

The diagram illustrates the configuration process for Firebase Storage in the console. It features three overlapping screenshots of the Firebase console interface, with red arrows indicating the sequence of steps.

**Step 1: Storage Overview**

The first screenshot shows the 'Storage' section in the Firebase console. The sidebar on the left lists various services: 'Desenvolver' (Develop), 'Authentication', 'Database', 'Storage', 'Hosting', 'Functions', and 'Machine Learning'. The main content area is titled 'Storage' and includes the description: 'Armazenar e recuperar arquivos gerados pelo usuário, como imagens, áudio e vídeos, sem o código do servidor'. A button labeled 'Primeiros passos' (Get started) is visible at the bottom of this section.

**Step 2: Configure Cloud Storage - Step 1**

The second screenshot shows the 'Configurar o Cloud Storage' (Configure Cloud Storage) wizard. It has two steps: '1. Regras seguras para o Cloud Storage' (Secure rules for Cloud Storage) and '2. Definir local do Cloud Storage' (Define Cloud Storage location). The first step is active. The text explains that by default, rules allow all reads and writes for authenticated users. It advises defining a data structure and creating rules to protect data, with a link to 'Saiba mais' (Learn more). A code editor shows the default security rules for Firebase Storage:

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

Buttons for 'Cancelar' (Cancel) and 'Próxima' (Next) are at the bottom.

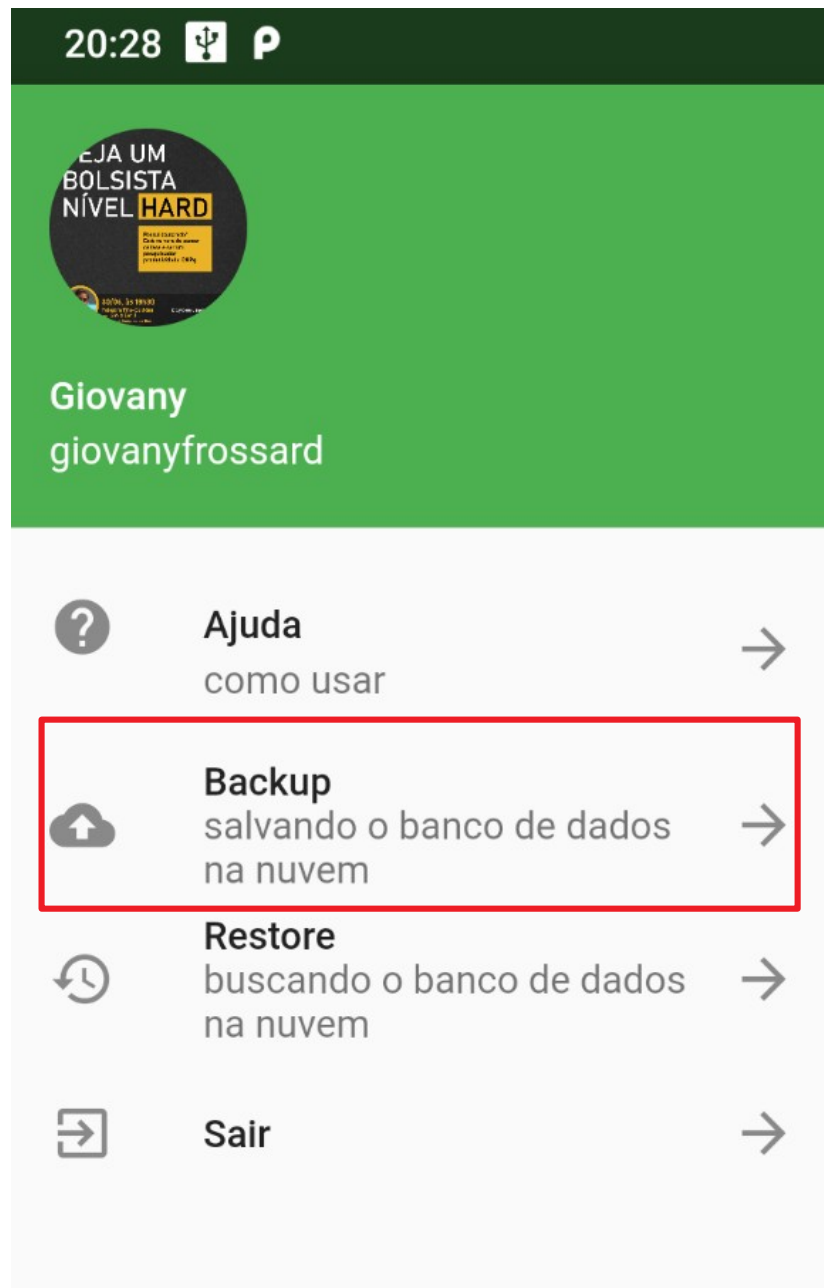
**Step 3: Configure Cloud Storage - Step 2**

The third screenshot shows the second step of the wizard: 'Definir local do Cloud Storage'. It explains that the local configuration is where the default bucket and data are stored. A text input field for 'Local no Cloud Storage' contains 'nam5 (us-central)'. Below this, it notes that Blaze plan clients can select other locations for additional intervals. Buttons for 'Cancelar' and 'Concluir' (Finish) are at the bottom.



# Backup do banco

```
ListTile _backup(BuildContext context) {  
  return ListTile(  
    leading: Icon(Icons.backup),  
    title: Text("Backup"),  
    subtitle: Text("salvando o banco de dados na nuvem"),  
    trailing: Icon(Icons.arrow_forward),  
    onTap: () async {  
      // Fechando o menu lateral  
      pop(context);  
  
      // Salvando o banco de dados na nuvem  
      DataBaseStorage.enviarBDParaStorage(usuario.login);  
    },  
  ); // ListTile  
}
```



```
static void enviarBDParaStorage(String nome_arquivo) async {  
    var storageRef = FirebaseStorage.instance.ref();  
    var bdRef = storageRef.child("/$nome_arquivo/$nome_arquivo.db");  
    StorageTaskSnapshot task = await bdRef  
        .putFile(File(  
            "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db"))  
        .onComplete;
```

- Primeiramente é criado um **StorageReference** inicializado na raiz do Firebase Storage Location.
- A partir da raiz é criada uma pasta com o nome **nome\_arquivo** e dentro dela um arquivo chamado **nome\_arquivo.bd** (nada foi enviado ainda).
- Na sequência, o arquivo **fundos.db** é enviado com o nome **nome\_arquivo.db**.
- Se o nome do arquivo for **admin**, por exemplo, é criada uma pasta **/admin** com um arquivo chamado **admin.db** cujo conteúdo é **fundos.db**.
- O caminho **/data/data/com.fundos.fundosimobiliarios/databases/** é o local onde o Android armazena o banco de dados (essa funcionalidade não foi implementada para iOS).



# Flutter 2.0

Praticamente não houve mudança,  
a linha do `getDownloadURL` pode  
ser retirada.

```
static void enviarBDParaStorage(String nome_arquivo) async {  
  var storageRef = FirebaseStorage.instance.ref();  
  var bdRef = storageRef.child("/$nome_arquivo/$nome_arquivo.db");  
  StorageTaskSnapshot task = await bdRef  
    .putFile(File(  
      "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db"))  
    .onComplete;
```

```
static void enviarBDParaStorage(String nome_arquivo) async {  
  var storageRef = FirebaseStorage.instance.ref("/$nome_arquivo/");  
  
  Reference storageReference = storageRef.child("$nome_arquivo.db");  
  await storageReference.putFile(File(banco));  
  await storageReference.getDownloadURL();
```

```
// Caminho em celulares reais  
static final String directorio_imagens = "/data/data/app.fiisplan.fiisplan2/app_flutter/";  
static final String banco = "/data/data/app.fiisplan.fiisplan2/databases/fundos.db";
```



- Também foram enviados os arquivos com extensão **db-shm** e **db-wal**.
- Verifiquei que o Android cria esses arquivos e enviar apenas o arquivo de extensão **.db** enviava um banco de dados não atualizado.

```
bdRef = storageRef.child("/$nome_arquivo/$nome_arquivo.db-shm");
task = await bdRef
    .putFile(File(
        "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db-shm"))
    .onComplete;
urlFile = await task.ref.getDownloadURL();
print("urlFile2 => $urlFile");

bdRef = storageRef.child("/$nome_arquivo/$nome_arquivo.db-wal");
task = await bdRef
    .putFile(File(
        "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db-wal"))
    .onComplete;
urlFile = await task.ref.getDownloadURL();
print("urlFile3 => $urlFile");
```

Device File Explorer

Motorola Motorola One Zoom Android 9, API 28

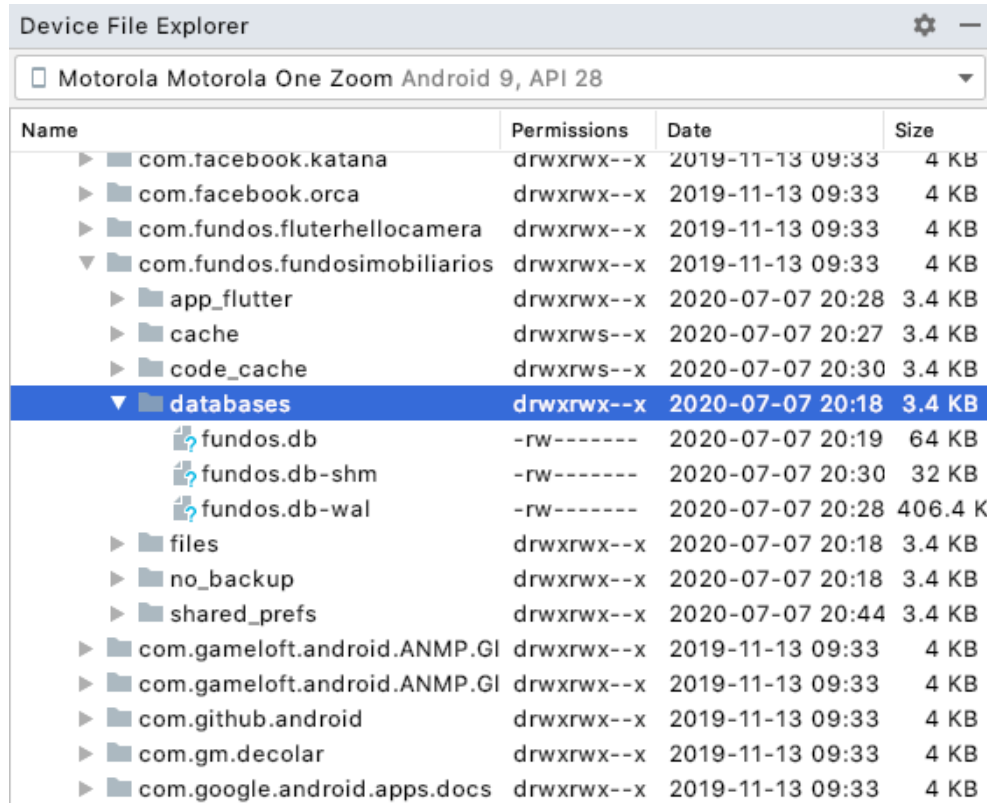
Name	Permissions	Date	Size
▶ com.facebook.katana	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.facebook.orca	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.fundos.fluterhellocamera	drwxrwx--x	2019-11-13 09:33	4 KB
▼ com.fundos.fundosimobiliarios	drwxrwx--x	2019-11-13 09:33	4 KB
▶ app_flutter	drwxrwx--x	2020-07-07 20:28	3.4 KB
▶ cache	drwxrws--x	2020-07-07 20:27	3.4 KB
▶ code_cache	drwxrws--x	2020-07-07 20:30	3.4 KB
▼ databases	drwxrwx--x	2020-07-07 20:18	3.4 KB
fundos.db	-rw-----	2020-07-07 20:19	64 KB
fundos.db-shm	-rw-----	2020-07-07 20:30	32 KB
fundos.db-wal	-rw-----	2020-07-07 20:28	406.4 K
▶ files	drwxrwx--x	2020-07-07 20:18	3.4 KB
▶ no_backup	drwxrwx--x	2020-07-07 20:18	3.4 KB
▶ shared_prefs	drwxrwx--x	2020-07-07 20:44	3.4 KB
▶ com.gameloft.android.ANMP.GI	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.gameloft.android.ANMP.GI	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.github.android	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.gm.decolar	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.google.android.apps.docs	drwxrwx--x	2019-11-13 09:33	4 KB

Androids mais antigos (9  
por exemplo)





# Flutter 2.0 – Device Explorer



Name	Permissions	Date	Size
▶ com.facebook.katana	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.facebook.orca	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.fundos.fluterhellocamera	drwxrwx--x	2019-11-13 09:33	4 KB
▼ com.fundos.fundosimobiliarios	drwxrwx--x	2019-11-13 09:33	4 KB
▶ app_flutter	drwxrwx--x	2020-07-07 20:28	3.4 KB
▶ cache	drwxrws--x	2020-07-07 20:27	3.4 KB
▶ code_cache	drwxrws--x	2020-07-07 20:30	3.4 KB
▼ databases	drwxrwx--x	2020-07-07 20:18	3.4 KB
fundos.db	-rw-----	2020-07-07 20:19	64 KB
fundos.db-shm	-rw-----	2020-07-07 20:30	32 KB
fundos.db-wal	-rw-----	2020-07-07 20:28	406.4 K
▶ files	drwxrwx--x	2020-07-07 20:18	3.4 KB
▶ no_backup	drwxrwx--x	2020-07-07 20:18	3.4 KB
▶ shared_prefs	drwxrwx--x	2020-07-07 20:44	3.4 KB
▶ com.gameloft.android.ANMP.GI	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.gameloft.android.ANMP.GI	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.github.android	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.gm.decolar	drwxrwx--x	2019-11-13 09:33	4 KB
▶ com.google.android.apps.docs	drwxrwx--x	2019-11-13 09:33	4 KB

```
/* Para ver o dispositivo no Device File Explorer é necessário:
=> Verificar se o Google USB Driver está instalado (SDK Manager->SDK Tools)
=> Verificar se o Android SDK está configurado no Projeto (Project Structure->
Project SDK
*/
```



# Inserindo as imagens

- O próximo passo é pegar as imagens dos usuários e enviar para o Firebase Storage.
- O método **filesInDirectory** irá obter uma lista dos arquivos dentro do diretório informado.
- Posteriormente, cada arquivo é enviado (dentro do for). A necessidade de usar **path.substring(52)** é para pegar o nome do arquivo (o path vem com o caminho inteiro).

```
Directory diretorio =  
    Directory("/data/data/com.fundos.fundosimobiliarios/app_flutter/");  
List<File> arquivos = await filesInDirectory(diretorio);  
for (int i = 0; i < arquivos.length; i++) {  
    bdRef = storageRef.child("/$nome_arquivo/imagens/${arquivos[i].path.substring(52)}");  
    task = await bdRef.putFile(arquivos[i]).onComplete;  
    urlFile = await task.ref.getDownloadURL();  
    print("arquivo$i => $urlFile");  
}  
}
```



# Flutter 2.0

```
// Caminho em celulares reais
static final String diretorio_imagens = "/data/data/app.fiisplan.fiisplan2/app_flutter/";
static final String banco = "/data/data/app.fiisplan.fiisplan2/databases/fundos.db";
```

```
Directory diretorio =
  Directory("/data/data/com.fundos.fundosimobiliarios/app_flutter/");
List<File> arquivos = await filesInDirectory(diretorio);
for (int i = 0; i < arquivos.length; i++) {
  bdRef = storageRef.child("/$nome_arquivo/imagens/${arquivos[i].path.substring(52)}");
  task = await bdRef.putFile(arquivos[i]).onComplete;
  urlFile = await task.ref.getDownloadURL();
  print("arquivo$i => $urlFile");
}
```

**O código é bastante parecido. Basicamente ocorreram acertos de tipagem e parâmetros.**

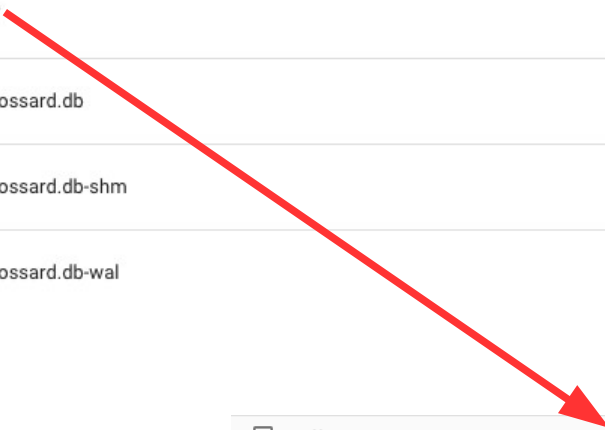
```
Directory diretorio =
  Directory(diretorio_imagens);
List<File> arquivos = await filesInDirectory(diretorio);
storageRef = FirebaseStorage.instance.ref("/$nome_arquivo/imagens/");
for (int i = 0; i < arquivos.length; i++) {
  // substring é utilizado para pegar o nome do arquivo de imagem
  // sem o restante do path
  storageReference = storageRef.child("${arquivos[i].path.substring(46)}");
  await storageReference.putFile(arquivos[i]);
  await storageReference.getDownloadURL();
}
MensagemAlerta("Backup executado com sucesso!!!");
```



# Resultado

<input type="checkbox"/>	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	giovaryfrossard/	—	Pasta	—

<input type="checkbox"/>	imagens/	—	Pasta	—
<input type="checkbox"/>	giovaryfrossard.db	64 KB		7 de jul. de 2020
<input type="checkbox"/>	giovaryfrossard.db-shm	32 KB		7 de jul. de 2020
<input type="checkbox"/>	giovaryfrossard.db-wal	406.4 KB		7 de jul. de 2020

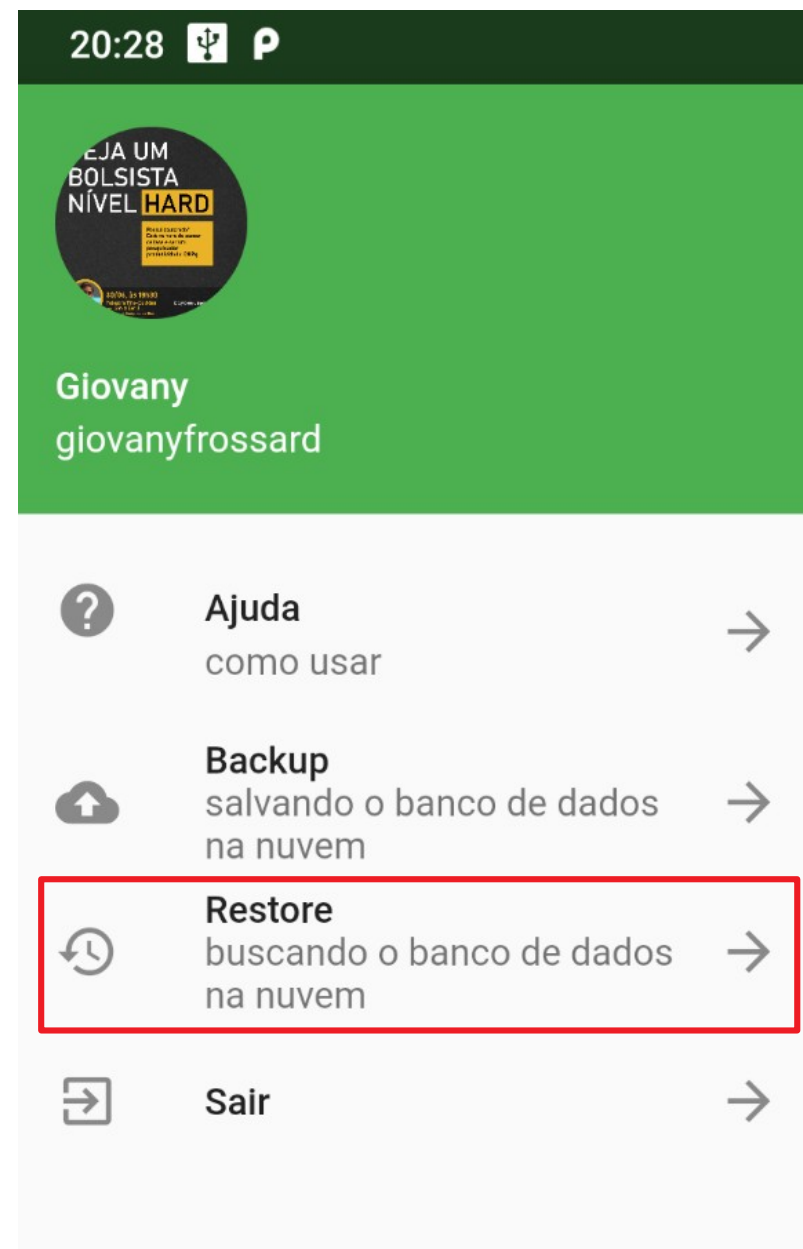


<input type="checkbox"/>	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	1593662715565.png	89.96 KB	image/png	7 de jul. de 2020
<input type="checkbox"/>	1593664116191.png	1.02 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	1594164068569.png	2.18 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	1594164091009.png	2.37 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	res_timestamp-1-1594166046016	—	Pasta	—



# Restore do banco

```
ListTile _restore(BuildContext context) {  
  return ListTile(  
    leading: Icon(Icons.restore),  
    title: Text("Restore"),  
    subtitle: Text("buscando o banco de dados na nuvem"),  
    trailing: Icon(Icons.arrow_forward),  
    onTap: () async {  
      // Fechando o menu lateral  
      pop(context);  
  
      // Salvando o banco de dados na nuvem  
      DataBaseStorage.buscarBDDoStorage(usuario.login);  
  
      // Sobrescrevendo a tela de Login  
      push(context, TelaLogin(), replace: true);  
  
      // Retirando o usuário das Shared Preferences  
      Usuario.limpar();  
    },  
  );  
}
```



```
static void buscarBDDoStorage(String nome_arquivo) {  
    String arquivo_db = "$nome_arquivo/$nome_arquivo.db";  
    StorageReference ref = FirebaseStorage.instance.ref().child(arquivo_db);  
    StorageFileDownloadTask task1 = ref.writeToFile(  
        File("/data/data/com.fundos.fundosimobiliarios/databases/fundos.db"));  
}
```

- Aqui utilizamos o caminho para o arquivo **.db** gerado pela rotina de Backup.
- É obtido então o **StorageReference** e a partir dele é sobrescrito o arquivo **fundos.db** do smartphone.
- A geração do **StorageFileDownloadTask** (task1) é importante pois precisaremos “pegar” as urls das imagens a partir dos usuários, nesse contexto, o banco de dados já precisa ter sido atualizado com a versão que está no Firebase Storage (essa questão ficará mais clara nos próximos slides).



# Flutter 2.0

```
static void buscarBDDoStorage(String nome_arquivo) {  
    String arquivo_db = "$nome_arquivo/$nome_arquivo.db";  
    StorageReference ref = FirebaseStorage.instance.ref().child(arquivo_db);  
    StorageFileDownloadTask task1 = ref.writeToFile(  
        File("/data/data/com.fundos.fundosimobiliarios/databases/fundos.db"));  
}
```

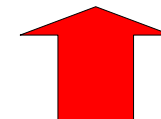
```
static Future<void> buscarBDDoStorage(String nome_arquivo) async {  
    var storageRef = FirebaseStorage.instance.ref("/$nome_arquivo/");  
    Reference storageReference = storageRef.child("$nome_arquivo.db");  
    TaskSnapshot task = await storageReference.writeToFile(File(banco));  
}
```

**Basicamente temos a troca do StorageFileDownloadTask pelo TaskSnapshot**



- Aqui temos a obtenção dos outros 2 arquivos para atualizar no smartphone. Ou seja, estamos "pegando" os arquivos de banco de dados do Firebase Storage e sobrescrevendo o banco local (dentro do smartphone).

```
ref = FirebaseStorage.instance.ref().child(arquivo_db + "-shm");  
StorageFileDownloadTask task2 = ref.writeToFile(File(  
    "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db-shm"));  
  
ref = FirebaseStorage.instance.ref().child(arquivo_db + "-wal");  
StorageFileDownloadTask task3 = ref.writeToFile(File(  
    "/data/data/com.fundos.fundosimobiliarios/databases/fundos.db-wal"));
```





- Aqui temos o uso do método **wait** para garantir que todas as tasks terminarão antes de se obter a lista de usuários. Dessa forma, quando os usuários forem obtidos, os arquivos de banco locais já terão sido sobrescritos pelos que estão no Firebase Storage.
- Na sequência temos os arquivos de imagem sendo escritos no dispositivo local.
- Se o usuário já tinha cadastrado imagens localmente, elas não serão apagadas (isso pode ser um problema pois ficará “lixo” para trás, entretanto, como o objetivo dessa rotina é fazer a restauração numa troca de smartphone, não foi implementada essa funcionalidade de “limpeza”).

```
Future.wait([task1.future, task2.future, task3.future]).then((value) {  
  Future<List<Usuario>> future = FabricaControladora.obterUsuarioControl()  
    .obterUsuarios();  
  future.then((usuarios) {  
    for (Usuario usuario in usuarios) {  
      if (usuario.urlFoto != null) {  
        String caminho_foto = "$nome_arquivo/imagens/${usuario.urlFoto  
          .substring(55)}";  
        ref = FirebaseStorage.instance.ref().child(caminho_foto);  
        ref.writeToFile(File(usuario.urlFoto));  
      }  
    }  
  });  
});  
});
```








# Flutter 2.0

```
task.ref.listAll().then((value) {  
  Future<List<Usuario>> future = FabricaControladora.obterUsuarioControl()  
    .obterUsuarios();  
  future.then((usuarios) async {  
    storageRef = FirebaseStorage.instance.ref("/$nome_arquivo/imagens/");  
    for (Usuario usuario in usuarios) {  
      if (usuario.urlFoto != null) {  
        String nome_foto = "${usuario.urlFoto!}";  
        String local_gravacao_imagem = "$diretorio_imagens${nome_foto.substring(48)}";  
        storageReference = storageRef.child(nome_foto.substring(48));  
        await storageReference.writeFile(File(local_gravacao_imagem));  
      }  
    }  
  });  
});
```

```
Future.wait([task1.future, task2.future, task3.future]).then((value) {  
  Future<List<Usuario>> future = FabricaControladora.obterUsuarioControl()  
    .obterUsuarios();  
  future.then((usuarios) {  
    for (Usuario usuario in usuarios) {  
      if (usuario.urlFoto != null) {  
        String caminho_foto = "/$nome_arquivo/imagens/${usuario.urlFoto  
          .substring(55)}";  
        ref = FirebaseStorage.instance.ref().child(caminho_foto);  
        ref.writeFile(File(usuario.urlFoto));  
      }  
    }  
  });  
});
```











# Resultado

<input type="checkbox"/>	Name	Tamanho	Tipo	Última modificação
<input type="checkbox"/>	 1593662715565.png	89.96 KB	image/png	7 de jul. de 2020
<input type="checkbox"/>	 1593664116191.png	1.02 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	 1594164068569.png	2.18 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	 1594164091009.png	2.37 MB	image/png	7 de jul. de 2020
<input type="checkbox"/>	 res_timestamp-1-1594166046016	—	Pasta	—

Device File Explorer

Emulator Pixel\_2\_API\_28 Android 9, API 28

Name	Permissions	Date	Size
com.example.calculadoraimcruiter	drwxrwx--x	2020-05-28 17:53	4 KB
com.example.somaweb service	drwxrwx--x	2020-05-28 17:53	4 KB
com.fundos.fundosimobiliarios	drwxrwx--x	2020-05-28 17:53	4 KB
▼ app_flutter	drwxrwx--x	2020-07-07 21:39	4 KB
flutter_assets	drwx-----	2020-07-07 21:35	4 KB
 1593662715565.png	-rw-----	2020-07-07 21:39	90 KB
 1593664116191.png	-rw-----	2020-07-07 21:39	1 MB
 1594164068569.png	-rw-----	2020-07-07 21:39	2.2 MB
 1594164091009.png	-rw-----	2020-07-07 21:39	2.4 MB
 res_timestamp-1-15941685255!	-rw-----	2020-07-07 21:35	0 B
cache	drwxrws--x	2020-07-07 21:35	4 KB
code_cache	drwxrws--x	2020-07-07 21:35	4 KB
▼ databases	drwxrwx--x	2020-07-07 21:35	4 KB
 fundos.db	-rw-----	2020-07-07 21:39	64 KB
 fundos.db-shm	-rw-----	2020-07-07 21:39	32 KB
 fundos.db-wal	-rw-----	2020-07-07 21:39	406.4 KB
files	drwxrwx--x	2020-07-07 21:35	4 KB
lib	lrwxrwxrwx	2020-07-07 21:35	72 B
no_backup	drwxrwx--x	2020-07-07 21:35	4 KB



# Dúvidas?

