Flutter Firebase

Платформа для разработки приложений Backend-as-a-Service (BaaS), которая предоставляет размещенные серверные службы, такие как база данных в реальном времени, облачное хранилище, аутентификация, отчеты о сбоях, машинное обучение, удаленная конфигурация и хостинг для ваших статических файлов.

Installation

pubspec.yaml:

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: "0.7.0"
```

Android configuration

Download and add to project: google-services.json

android/build.gradle

```
buildscript {
  dependencies {
    // ... other dependencies
    classpath 'com.google.gms:google-services:4.3.3'
  }
}
```

android/app/build.gradle

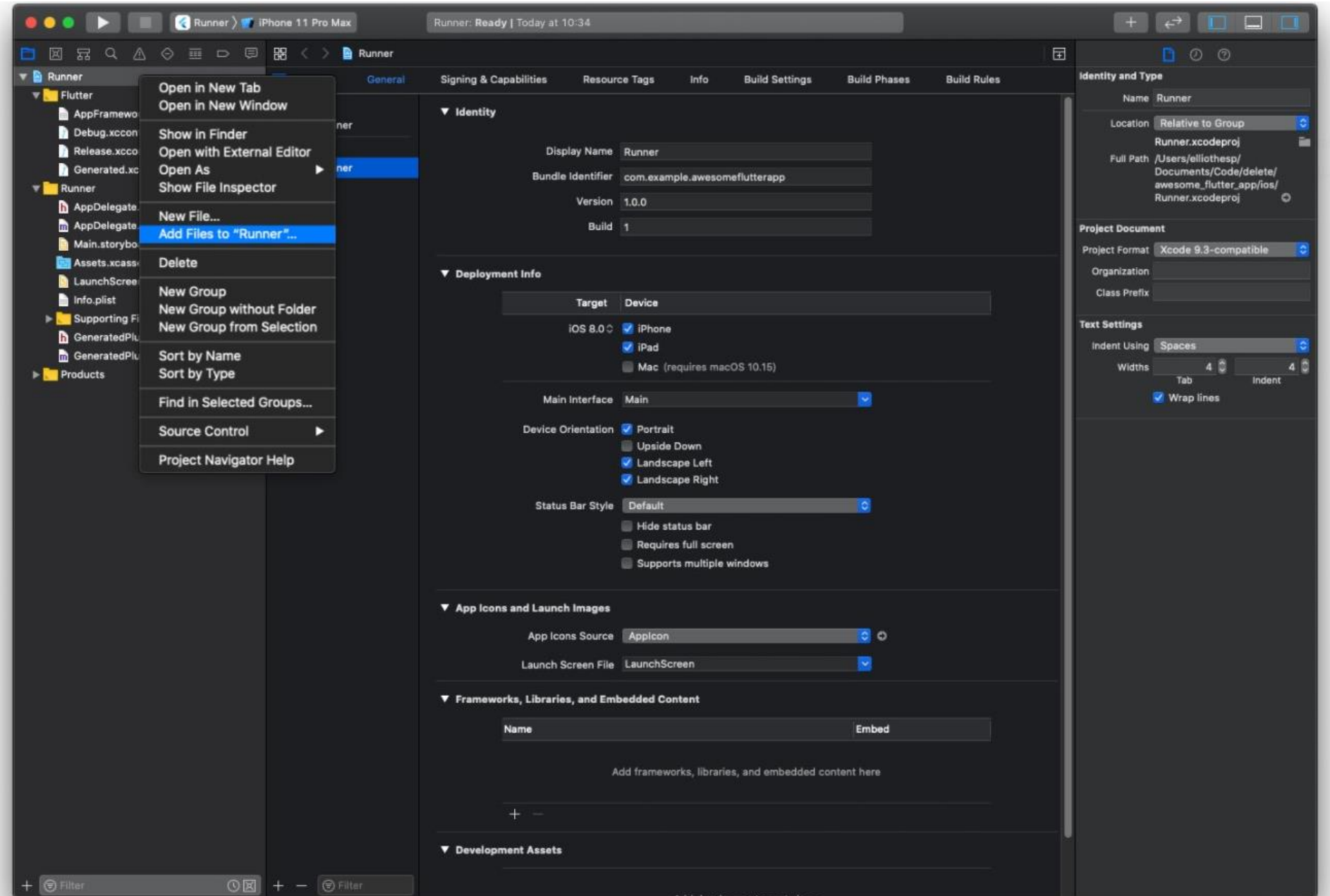
```
apply plugin: 'com.google.gms.google-services'
android {
defaultConfig {
// ...
minSdkVersion 16
targetSdkVersion 28
multiDexEnabled true
}
}
dependencies {
implementation 'com.android.support:multidex:1.0.3'
}
```

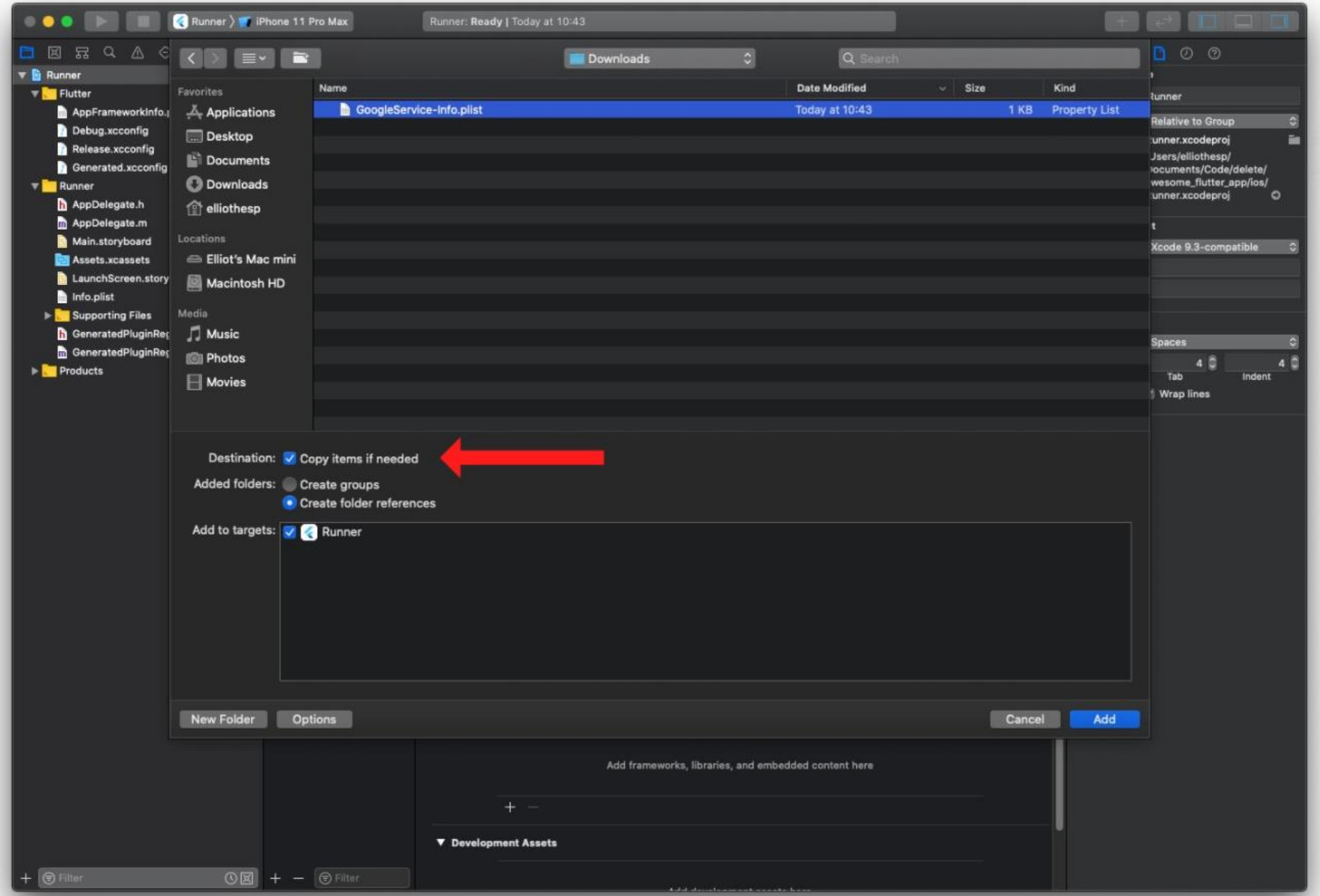
AndroidManifest.xml

```
<application usesCleartextTraffic="true">
<!-- possibly other elements -->
</application>
```

iOS configuration

Download and add to project: GoogleService-Info.plist




Runner
Flutter
AppFrameworkInfo.
Debug.xcconfig
Release.xcconfig
Generated.xcconfig
Runner
AppDelegate.h
AppDelegate.m
Main.storyboard
Assets.xcassets
LaunchScreen.story
Info.plist
Supporting Files
GeneratedPluginReg
GeneratedPluginReg
Products
Downloads
Search
Favorites
Applications
Desktop
Documents
Downloads
elliothesp
Locations
Elliot's Mac mini
Macintosh HD
Media
Music
Photos
Movies
Name
Date Modified
Size
Kind
GoogleService-Info.plist
Today at 10:43
1 KB
Property List
Destination:
Copy items if needed
Added folders:
Create groups
Create folder references
Add to targets:
Runner
New Folder
Options
Cancel
Add
Add frameworks, libraries, and embedded content here
Development Assets
Relative to Group
Xcode 9.3-compatible
Spaces
Tab
Indent
Wrap lines

Enabling use of Firebase Emulator Suite

Info.plist

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsLocalNetworking</key>
    <true/>
</dict>
```

Initializing Firebase

```
final Future<FirebaseApp> _initialization = Firebase.initializeApp();

@override
Widget build(BuildContext context) {
return FutureBuilder(
future: _initialization,
builder: (context, snapshot) {
if (snapshot.hasError) {
return SomethingWentWrong();
}
if (snapshot.connectionState == ConnectionState.done) {
return MyAwesomeApp();
}
return Loading();
},
);
}
```

Instruments

Analytics

Authentication

Cloud Firestore

Cloud Functions

Cloud Messaging

Cloud Storage

Crashlytics

Realtime Database

Performance Monitoring

Authentication

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: "^0.7.0"
  firebase_auth: "^0.20.1"
```

Get Instance

```
FirebaseAuth auth = FirebaseAuth.instance;
```

Authentication state

```
FirebaseAuth.instance
  .authStateChanges()
  .listen((User user) {
    if (user == null) {
      print('User is currently signed out!');
    } else {
      print('User is signed in!');
    }
  });
```

Sign-in methods

Anonymous sign-in

```
UserCredential userCredential = await FirebaseAuth.instance.signInAnonymously();
```

Email/Password Registration & Sign-in

Social Auth

Phone Auth

Email/Password Registration & Sign-in

Registration

```
try {
  UserCredential userCredential = await FirebaseAuth.instance.createUserWithEmailAndPassword(
    email: "barry.allen@example.com",
    password: "SuperSecretPassword!"
  );
} on FirebaseAuthException catch (e) {
  if (e.code == 'weak-password') {
    print('The password provided is too weak.');
  } else if (e.code == 'email-already-in-use') {
    print('The account already exists for that email.');
  }
} catch (e) {
  print(e);
}
```

Sign-in

```
try {
  UserCredential userCredential = await FirebaseAuth.instance.signInWithEmailAndPassword(
    email: "barry.allen@example.com",
    password: "SuperSecretPassword!"
  );
} on FirebaseAuthException catch (e) {
  if (e.code == 'user-not-found') {
    print('No user found for that email.');
  } else if (e.code == 'wrong-password') {
    print('Wrong password provided for that user.');
  }
}
```

Verifying a users email

```
User user = FirebaseAuth.instance.currentUser;

if (!user.emailVerified) {
  await user.sendEmailVerification();
}
```

Open link in app

```
User user = FirebaseAuth.instance.currentUser;

if (!user.emailVerified) {

var actionCodeSettings = ActionCodeSettings(
        url: 'https://www.example.com/?email=${user.email}',
        dynamicLinkDomain: "example.page.link",
        android: {
          "packageName": "com.example.android",
          "installApp": true,
          "minimumVersion": "12"
        },
        iOS: {"bundleId": "com.example.ios"},
        handleCodeInApp: true);

await user.sendEmailVerification(actionCodeSettings);

}
```

```
FirebaseAuth auth = FirebaseAuth.instance;

//Get actionCode from the dynamicLink
final Uri deepLink = dynamicLink?.link;
var actionCode = deepLink.queryParameters['oobCode'];

try {
  await auth.checkActionCode(actionCode);
  await auth.applyActionCode(actionCode);

  // If successful, reload the user:
  auth.currentUser.reload();
} on FirebaseAuthException catch (e) {
  if (e.code == 'invalid-action-code') {
    print('The code is invalid.');
  }
}
```

Cloud Firestore

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: "^0.7.0"
  cloud_firestore: "^0.16.0+1"
```

Get Instance

```
FirebaseFirestore firestore = FirebaseFirestore.instance;
```

Collections & Documents

```
class AddUser extends StatelessWidget {
final String fullName;
final String company;
final int age;
AddUser(this.fullName, this.company, this.age);
@override
Widget build(BuildContext context) {
CollectionReference users = FirebaseFirestore.instance.collection('users'); // Create a CollectionReference called users that references the firestore collection
Future<void> addUser() {// Call the user's CollectionReference to add a new user
return users
.add({
'full_name': fullName, // John Doe
'company': company, // Stokes and Sons
'age': age // 42
})
.then((value) => print("User Added"))
.catchError((error) => print("Failed to add user: $error"));
}
return TextButton(
onPressed: addUser,
child: Text(
"Add User",
),);}}
```

Read Data

```
class GetUserName extends StatelessWidget {
  final String documentId;
  GetUserName(this.documentId);
  @override
  Widget build(BuildContext context) {
    CollectionReference users = FirebaseFirestore.instance.collection('users');
    return FutureBuilder<DocumentSnapshot>(
      future: users.doc(documentId).get(),
      builder:
          (BuildContext context, AsyncSnapshot<DocumentSnapshot> snapshot) {
        if (snapshot.hasError) {
          return Text("Something went wrong");
        }
        if (snapshot.connectionState == ConnectionState.done) {
          Map<String, dynamic> data = snapshot.data.data();
          return Text("Full Name: ${data['full_name']} ${data['last_name']}");
        }
        return Text("loading");
      }, );}}
```

Realtime changes

```
class UserInformation extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    CollectionReference users = FirebaseFirestore.instance.collection('users');
    return StreamBuilder<QuerySnapshot>(
      stream: users.snapshots(),
      builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {
        if (snapshot.hasError) {
          return Text('Something went wrong');
        }
        if (snapshot.connectionState == ConnectionState.waiting) {
          return Text("Loading");
        }
        return new ListView(
          children: snapshot.data.docs.map((DocumentSnapshot document) {
            return new ListTile(
              title: new Text(document.data()['full_name']),
              subtitle: new Text(document.data()['company']),
            );
          }).toList(),
        );},);}}
```

Document & Query Snapshots

QuerySnapshot результат запроса коллекции и позволяет вам проверять коллекцию, например, сколько документов существует в ней, дает доступ к документам в коллекции, видеть любые изменения с момента последнего запроса и многое другое.

```
FirebaseFirestore.instance
.collection('users')
.get()
.then((QuerySnapshot querySnapshot) => {
querySnapshot.docs.forEach((doc) {
print(doc["first_name"]);
});
});
```

DocumentSnapshot возвращается из запроса или путем прямого доступа к документу. Даже если в базе данных нет документа, всегда будет возвращаться моментальный снимок.

```
FirebaseFirestore.instance
.collection('users')
.doc(userId)
.get()
.then((DocumentSnapshot documentSnapshot) {
if (documentSnapshot.exists) {
print('Document exists on the database');
}
});
```

Querying

Filtering

```
FirebaseFirestore.instance
.collection('users')
.where('age', isGreaterThan: 20)
//.where('language', arrayContainsAny: ['en', 'it'])
.get()
.then(...);
```

Limiting

```
FirebaseFirestore.instance
.collection('users')
.limit(2)
//.limitToLast(2)
.get()
.then(...);
```

Ordering

```
FirebaseFirestore.instance
  .collection('users')
  .orderBy('age', descending: true)
  .get()
  .then(...);
```

Start & End Cursors

```
FirebaseFirestore.instance
  .collection('users')
  .orderBy('age')
  .orderBy('company')
  .startAt([4, 'Alphabet Inc.'])
  .endAt([21, 'Google LLC’])
//.startAfterDocument(documentSnapshot)
  .get()
  .then(...);
```

Adding Documents

```
CollectionReference users = FirebaseFirestore.instance.collection('users');
Future<void> addUser() {
// Call the user's CollectionReference to add a new user
return users
.add({
'full_name': fullName, // John Doe
'company': company, // Stokes and Sons
'age': age // 42
})
.then((value) => print("User Added"))
.catchError((error) => print("Failed to add user: $error"));
}
```

Метод add добавляет новый документ в вашу коллекцию с уникальным автоматически сгенерированным идентификатором. Если вы хотите указать свой собственный идентификатор, вместо этого вызовите метод set в DocumentReference:

```
CollectionReference users = FirebaseFirestore.instance.collection('users');


Future<void> addUser() {
  return users
    .doc('ABC123')
    .set({
      'full_name': "Mary Jane",
      'age': 18
    })
    .then((value) => print("User Added"))
    .catchError((error) => print("Failed to add user: $error"));
}
```

Updating documents

```
CollectionReference users = FirebaseFirestore.instance.collection('users');

Future<void> updateUser() {
  return users
    .doc('ABC123')
    .update({'company': 'Stokes and Sons’})
//.update({'info.address.zipcode': 90210})
    .then((value) => print("User Updated"))
    .catchError((error) => print("Failed to update user: $error"));
}
```

Removing Data

```
CollectionReference users = FirebaseFirestore.instance.collection('users');

Future<void> deleteUser() {
  return users
    .doc('ABC123')
    .delete()
//.update({'age': FieldValue.delete()})
    .then((value) => print("User Deleted"))
    .catchError((error) => print("Failed to delete user: $error"));
}
```

Transactions

```
DocumentReference documentReference = FirebaseFirestore.instance
  .collection('users')
  .doc(documentId);

return Firestore.instance.runTransaction((transaction) async {
  // Get the document
  DocumentSnapshot snapshot = await transaction.get(documentReference);

  if (!snapshot.exists) {
    throw Exception("User does not exist!");
  }

  // Update the follower count based on the current count
  // Note: this could be done without a transaction
  // by updating the population using FieldValue.increment()

  int newFollowerCount = snapshot.data()['followers'] + 1;

  // Perform an update on the document
  transaction.update(documentReference, {'followers': newFollowerCount});

  // Return the new count
  return newFollowerCount;
})
.then((value) => print("Follower count updated to $value"))
.catchError((error) => print("Failed to update user followers: $error"));
```

Cloud Storage

```
dependencies:
  flutter:
    sdk: flutter
  firebase_core: "^0.7.0"
  firebase_storage: "^7.0.0"
```

Example

Get Instance:

```
import 'package:firebase_storage/firebase_storage.dart' as firebase_storage;
firebase_storage.FirebaseStorage storage =
  firebase_storage.FirebaseStorage.instance;
```

References

```
firebase_storage.Reference ref =
  firebase_storage.FirebaseStorage.instance.ref('/notes.txt’);

firebase_storage.Reference ref = firebase_storage.FirebaseStorage.instance
    .ref('images/defaultProfile.png');
// or
firebase_storage.Reference ref = firebase_storage.FirebaseStorage.instance
    .ref()
    .child('images')
    .child('defaultProfile.png');
```

Listing files & directories

```
Future<void> listExample() async {
  firebase_storage.ListResult result =
      await firebase_storage.FirebaseStorage.instance.ref().listAll();

  result.items.forEach((firebase_storage.Reference ref) {
    print('Found file: $ref');
  });

  result.prefixes.forEach((firebase_storage.Reference ref) {
    print('Found directory: $ref');
  });
}
```

```
 Future<void> listExample() async {
   firebase_storage.ListResult result = await firebase_storage
       .FirebaseStorage.instance
       .ref()
       .list(firebase_storage.ListOptions(maxResults: 10));
   // ...
 }
```

```
Future<void> listExample() async {
  firebase_storage.ListResult result = await firebase_storage
      .FirebaseStorage.instance
      .ref()
      .list(firebase_storage.ListOptions(maxResults: 10));

  if (result.nextPageToken != null) {
    firebase_storage.ListResult additionalResults = await firebase_storage
        .FirebaseStorage.instance
        .ref()
        .list(firebase_storage.ListOptions(
          maxResults: 10,
          pageToken: result.nextPageToken,
        ));
  }
}
```

Download URLs

```
Future<void> downloadURLExample() async {
  String downloadURL = await firebase_storage.FirebaseStorage.instance
      .ref('users/123/avatar.jpg')
      .getDownloadURL();

}
```

Uploading Files

```
import 'package:path_provider/path_provider.dart';
Future<void> uploadExample() async {
Directory appDocDir = await getApplicationDocumentsDirectory();
String filePath = '${appDocDir.absolute}/file-to-upload.png';
// ...
// e.g. await uploadFile(filePath);
}

Future<void> uploadFile(String filePath) async {
File file = File(filePath);
try {
await firebase_storage.FirebaseStorage.instance
.ref('uploads/file-to-upload.png')
.putFile(file);
} on firebase_core.FirebaseException catch (e) {
// e.g, e.code == 'canceled'
}
}
```

Downloading Files

```
import 'package:path_provider/path_provider.dart';

Future<void> downloadFileExample() async {
  Directory appDocDir = await getApplicationDocumentsDirectory();
  File downloadToFile = File('${appDocDir.path}/download-logo.png');

  try {
    await firebase_storage.FirebaseStorage.instance
        .ref('uploads/logo.png')
        .writeToFile(downloadToFile);
  } on firebase_core.FirebaseException catch (e) {
    // e.g, e.code == 'canceled'
  }
}
```

Handling Tasks

```
Future<void> handleTaskExample1() async {
  firebase_storage.UploadTask task = firebase_storage.FirebaseStorage.instance
      .ref('uploads/hello-world.txt')
      .putString('Hello World');

  try {
    // Storage tasks function as a Delegating Future so we can await them.
    firebase_storage.TaskSnapshot snapshot = await task;
    print('Uploaded ${snapshot.bytesTransferred} bytes.');
  } on firebase_core.FirebaseException catch (e) {
    // The final snapshot is also available on the task via `.snapshot`,
    // this can include 2 additional states, `TaskState.error` & `TaskState.canceled`
    print(task.snapshot);

    if (e.code == 'permission-denied') {
      print('User does not have permission to upload to this reference.');
    }
    // ...
  }
}
```

```
Future<void> handleTaskExample2(String filePath) async {
 File largeFile = File(filePath);
 firebase_storage.UploadTask task = firebase_storage.FirebaseStorage.instance
     .ref('uploads/hello-world.txt')
     .putFile(largeFile);
 task.snapshotEvents.listen((firebase_storage.TaskSnapshot snapshot) {
  print('Task state: ${snapshot.state}');
  print('Progress: ${(snapshot.bytesTransferred / snapshot.totalBytes) * 100} %');
 }, onError: (e) {
  // The final snapshot is also available on the task via `.snapshot`, this can include 2 additional states, `TaskState.error` & `TaskState.canceled`
  print(task.snapshot);
  if (e.code == 'permission-denied') {
   print('User does not have permission to upload to this reference.');
  }
 });
 // We can still optionally use the Future alongside the stream.
 try {
  await task;
  print('Upload complete.');
 } on firebase_core.FirebaseException catch (e) {
  if (e.code == 'permission-denied') {
   print('User does not have permission to upload to this reference.');
 }}}
```

```
Future<void> handleTaskExample3(String filePath) async {
  File largeFile = File(filePath);

  firebase_storage.UploadTask task = firebase_storage.FirebaseStorage.instance
      .ref('uploads/hello-world.txt')
      .putFile(largeFile);

  // Pause the upload.
  bool paused = await task.pause();
  print('paused, $paused');

  // Resume the upload.
  bool resumed = await task.resume();
  print('resumed, $resumed');

  // Cancel the upload.
  bool cancelled = await task.cancel();
  print('cancelled, $cancelled');

  // ...
}
```