Firestore CRUD and Firebase Storage with Dart

March 26, 2024

Contents

1 Introduction to Firebase

Firebase is a comprehensive platform provided by Google for mobile and web application development. It offers various services including real-time database, authentication, cloud storage, machine learning, and more. Firebase allows developers to build high-quality apps quickly, with features like real-time synchronization, offline support, and easy integration with popular frameworks like Flutter.

2 Firestore Overview

Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. It is a NoSQL document database that lets you store, sync, and query data for your applications. Firestore offers seamless integration with other Firebase services and provides real-time updates for data changes.

2.1 Features of Firestore

- Real-time updates: Changes made to the database are automatically propagated to all connected clients in real-time.
- Scalability: Firestore automatically scales to handle millions of users and petabytes of data.
- Offline support: Firestore provides offline data persistence, allowing users to access data even when offline.
- Strong consistency: Firestore ensures strong consistency for reads and writes across all regions.
- Security rules: Firestore allows you to define security rules to control access to your data.
- Integration with Firebase ecosystem: Firestore seamlessly integrates with other Firebase services such as Authentication, Cloud Functions, and Firebase Storage.

3 CRUD Operations in Firestore

CRUD stands for Create, Read, Update, and Delete. Firestore provides APIs to perform these operations on your data.

3.1 Create

To add a new document to Firestore, you can use the `add()` method on a collection reference. Below is an example of adding a new user document:

```
Future<void> addUser(String name, String email) {
  return _users.add({
    'name': name,
    'email': email,
    'average': 0.0,
    // Additional fields...
  });
}
```

Listing 1: Adding a new user document

3.2 Read

Firestore allows you to retrieve documents from a collection using queries. You can use methods like `get()` or `where()` to retrieve documents based on certain criteria. Below is an example of retrieving users sorted by their average score:

```
Future<List<User>> initUsers() async {
  QuerySnapshot users = await _users.orderBy("average",
   descending: false).get();
  List<User> usersList = [];
  for (var doc in users.docs) {
    User temp = User.fromFirestore(doc);
    usersList.add(temp);
  }
  return usersList;
}
```

Listing 2: Retrieving users sorted by average score

3.3 Update

You can update existing documents in Firestore using the `update()` method. Below is an example of updating the average score of a user:

```
Future<void> setAverage(double average, String id) async {
  DocumentSnapshot user = await FirebaseFirestore.instance.
   collection("users").doc(id).get();
  await user.reference.update({"average": average});
}
```

Listing 3: Updating the average score of a user

3.4 Delete

To delete a document from Firestore, you can use the `delete()` method. Below is an example of deleting a user document:

```
Future<void> deleteUser(String id) async {
  return await _users.doc(id).delete();
}
```

Listing 4: Deleting a user document

4 Firebase Storage

Firebase Storage is a powerful, simple, and cost-effective object storage service provided by Google. It allows you to store and serve user-generated content such as photos and videos.

4.1 Features of Firebase Storage

- Scalability: Firebase Storage automatically scales to handle large amounts of data.
- Security: Firebase Storage provides security features like user authentication and access control.
- Integration: Firebase Storage seamlessly integrates with other Firebase services like Firestore and Firebase Authentication.

- File Metadata: Firebase Storage allows you to store metadata along with files, such as file name, content type, and custom metadata.
- Resumable Uploads: Firebase Storage supports resumable uploads, allowing users to pause and resume uploads without losing data.
- File Versioning: Firebase Storage supports file versioning, allowing users to access previous versions of files and restore them if needed.

4.2 Uploading Files to Firebase Storage

To upload files to Firebase Storage, you can use the Firebase Storage SDK. Below is an example of uploading a file:

```
void uploadFile() {
File file = File('path/to/file.jpg');
firebase_storage.Reference ref = firebase_storage.
    FirebaseStorage.instance.ref('path/to/upload.jpg');
ref.putFile(file);
}
```

Listing 5: Uploading a file to Firebase Storage

4.3 Downloading Files from Firebase Storage

Downloading files from Firebase Storage is straightforward using the SDK. Below is an example of downloading a file:

```
void downloadFile() {
firebase_storage.Reference ref = firebase_storage.
    FirebaseStorage.instance.ref('path/to/file.jpg');
ref.getData();
}
```

Listing 6: Downloading a file from Firebase Storage

5 Integrating Firebase with Dart/Flutter

Integrating Firebase with Dart/Flutter is essential for building powerful mobile and web applications. Firebase provides official plugins for Dart/Flutter that simplify the integration process.

5.1 Setting up Firebase in a Dart/Flutter Project

To integrate Firebase with your Dart/Flutter project, follow these steps:

1. Create a Firebase project on the Firebase Console.
2. Add your Dart/Flutter project to the Firebase project.
3. Download and add the `google-services.json` (for Android) or `GoogleService-Info.plist` (for iOS) file to your Dart/Flutter project.
4. Add the necessary Firebase plugins to your `pubspec.yaml` file.
5. Initialize Firebase in your Dart/Flutter project.

5.2 Initializing Firebase in Dart/Flutter

To initialize Firebase in your Dart/Flutter project, you need to call the `Firebase.initializeApp()` method. Below is an example of initializing Firebase in a Flutter project:

```
void main() async {
WidgetsFlutterBinding.ensureInitialized();
await Firebase.initializeApp();
runApp(MyApp());
}
```

Listing 7: Initializing Firebase in a Flutter project

5.3 Using Firebase Services in Dart/Flutter

Once Firebase is initialized in your Dart/Flutter project, you can start using Firebase services such as Firestore, Firebase Authentication, and Firebase Storage. Below is an example of using Firestore in a Flutter project:

```
void addUserToFirestore() {
FirebaseFirestore.instance.collection('users').add({
'name': 'John Doe',
'email': 'john.doe@example.com',
'age': 30,
});
}
```

Listing 8: Using Firestore in a Flutter project

6 Advanced Firestore Operations

Firestore provides advanced features such as querying, indexing, and transactions to perform complex operations efficiently.

6.1 Querying Documents

You can query documents in Firestore using methods like `where()`, `orderBy()`, and `limit()` to filter, sort, and limit the results.

6.2 Indexing

Firestore automatically indexes fields used in queries. However, for complex queries, you may need to define composite indexes.

6.3 Transactions

Firestore transactions ensure the consistency of data by allowing you to perform multiple read and write operations atomically.

7 Firebase Authentication

Firebase Authentication provides easy-to-use SDKs and ready-made UI libraries to authenticate users to your app.

7.1 Authentication Methods

Firebase Authentication supports various authentication methods such as email/password, phone number, Google Sign-In, Facebook Login, and more.

7.2 Integration with Firestore

You can integrate Firebase Authentication with Firestore to control access to data based on user authentication.

8 Firebase Security Rules

Firebase Security Rules allow you to control access to your Firestore data. You can define rules to restrict access based on user authentication, data validation, and conditions.

8.1 Example Security Rules

Below are example security rules for a Firestore database:

```
service cloud.firestore {
match /databases/{database}/documents {
match /users/{userId} {
allow read, write: if request.auth != null && request.auth.
    uid == userId;
}
}
}
```

Listing 9: Example Firestore Security Rules

9 Advanced Firebase Features

9.1 Cloud Functions for Firebase

Cloud Functions for Firebase allows you to run backend code in response to events triggered by Firebase features and HTTPS requests. You can write functions using Node.js, Python, Go, or Java.

9.2 Firebase Remote Config

Firebase Remote Config allows you to dynamically change the behavior and appearance of your app without publishing app updates. You can use Remote Config to customize your app for different user segments or to run A/B tests.

9.3 Firebase Analytics

Firebase Analytics provides detailed insights into user behavior and app performance. You can track user engagement, retention, and conversion rates, and use this data to optimize your app's performance.

9.4 Firebase Cloud Messaging (FCM)

Firebase Cloud Messaging allows you to send push notifications to users across platforms (iOS, Android, and web). You can target specific user segments and personalize notifications based on user behavior.

9.5 Firebase Performance Monitoring

Firebase Performance Monitoring allows you to monitor your app's performance in real-time. You can track app startup time, network latency, and UI rendering performance to identify and fix performance issues.

10 Firebase Machine Learning

10.1 Firebase ML Kit

Firebase ML Kit provides ready-to-use machine learning models and APIs for common use cases such as text recognition, face detection, barcode scanning,

and image labeling. You can integrate ML Kit into your app with just a few lines of code and leverage the power of machine learning without needing expertise in data science or machine learning algorithms.

10.2 Custom Machine Learning Models

In addition to pre-built models provided by ML Kit, Firebase allows you to deploy custom machine learning models using Firebase ML Custom. You can train your models using TensorFlow Lite and deploy them to Firebase for inference on-device or in the cloud.

10.3 Firebase AutoML

Firebase AutoML enables you to train high-quality custom machine learning models with minimal effort and expertise. You can upload your training data to Firebase, and AutoML automatically trains and optimizes machine learning models tailored to your specific use case.

11 Firebase Cloud Firestore Security Rules

11.1 Role-Based Access Control (RBAC)

Firebase Cloud Firestore Security Rules support role-based access control, allowing you to define different levels of access for different user roles. You can specify read and write permissions based on user authentication and custom user attributes.

11.2 Granular Access Control

With Firebase Security Rules, you can enforce granular access control at the document and field levels. You can restrict access to sensitive documents or fields based on user roles or conditions.

11.3 Validation Rules

Firebase Security Rules support validation rules, allowing you to enforce data validation at the time of write operations. You can ensure that data meets certain criteria before allowing it to be written to the database.

12 Firebase Realtime Database

12.1 Real-time Data Synchronization

Firebase Realtime Database provides real-time data synchronization, allowing multiple clients to listen for data changes and receive updates in real-time. This enables collaborative and interactive app experiences.

12.2 Offline Persistence

Firebase Realtime Database supports offline persistence, allowing users to access and modify data even when offline. Data changes made offline are synchronized with the server once the device reconnects to the internet.

12.3 Scalability and Performance

Firebase Realtime Database automatically scales to handle thousands of concurrent connections and petabytes of data. It provides low-latency data access and real-time synchronization, ensuring optimal performance for your app.

13 Firebase Cloud Functions

13.1 Serverless Computing

Firebase Cloud Functions allows you to run server-side code without managing servers. You can write functions using JavaScript, TypeScript, or any Node.js runtime, and deploy them to Firebase with a simple command.

13.2 Event-Driven Architecture

Firebase Cloud Functions follows an event-driven architecture, allowing you to trigger functions in response to Firebase events such as database writes, authentication events, or HTTP requests. This enables you to automate tasks and integrate third-party services with your Firebase app.

13.3 Integration with Firebase Services

Firebase Cloud Functions seamlessly integrates with other Firebase services such as Firestore, Firebase Authentication, and Firebase Storage. You can use functions to extend the functionality of your Firebase app and automate backend tasks.

14 Firebase Hosting

14.1 Static Web Hosting

Firebase Hosting allows you to host your web app's static assets (HTML, CSS, JavaScript) with a global content delivery network (CDN). This ensures fast loading times and high performance for your web app users worldwide.

14.2 Custom Domain Support

Firebase Hosting supports custom domain names, allowing you to serve your web app from your own domain. You can easily connect your domain to Firebase Hosting using DNS configuration.

14.3 SSL Encryption

Firebase Hosting provides SSL encryption for all hosted sites by default, ensuring that data transmitted between the user's browser and your web app is secure. This improves user trust and protects sensitive information.

15 Conclusion

Firebase offers a comprehensive set of tools and services for building modern mobile and web applications. From real-time database synchronization to machine learning capabilities, Firebase provides developers with the tools they need to create engaging and scalable apps. By leveraging Firebase's cloud infrastructure and developer-friendly APIs, developers can focus on building great user experiences without worrying about managing servers or infrastructure.

16 References

1. Firebase Documentation: `https://firebase.google.com/docs`
2. FlutterFire Documentation: `https://firebase.flutter.dev`
3. Dart Documentation: `https://dart.dev`