



Cloud Firestore: O Banco de Dados NoSQL Flexível e Escalável do Google

O Cloud Firestore é um banco de dados de nuvem NoSQL flexível e escalável, construído sobre a robusta infraestrutura do Google Cloud. Ele permite armazenar e sincronizar dados de forma eficiente para o desenvolvimento de aplicativos móveis, web e de servidor, oferecendo recursos poderosos e integração perfeita com o ecossistema Firebase e Google Cloud.

Por Que Escolher o Cloud Firestore?



Focado em Mobile e Web

Desenvolvido para aplicações modernas, garantindo sincronização em tempo real e suporte off-line.



Sincronização em Tempo Real

Mantém seus dados atualizados em todos os clientes conectados através de listeners em tempo real.



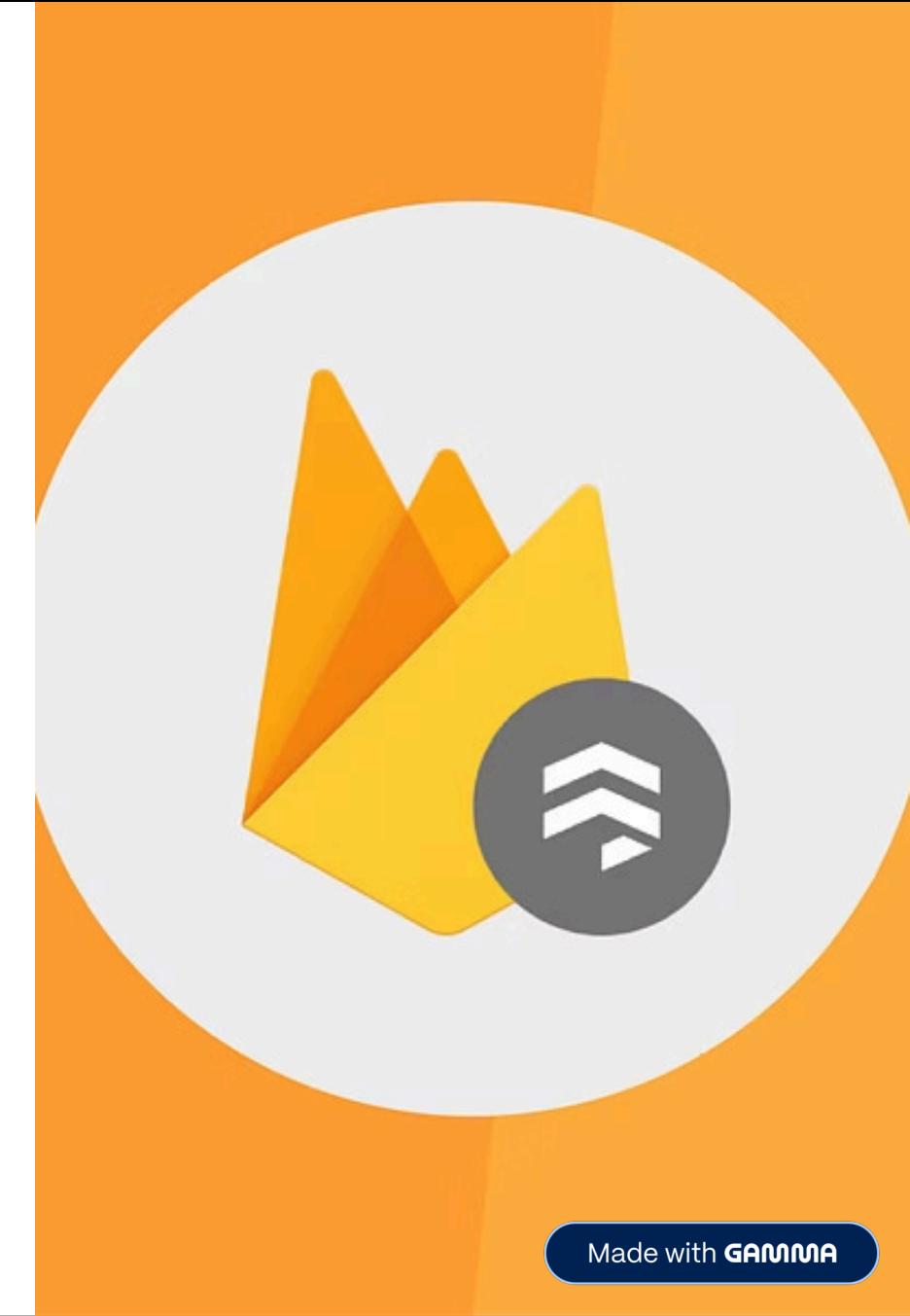
Suporte Off-line

Permite que aplicativos funcionem perfeitamente mesmo sem conexão, sincronizando as mudanças quando online.



Integração Total

Compatibilidade nativa com Firebase, Google Cloud e Cloud Functions para um ecossistema completo.

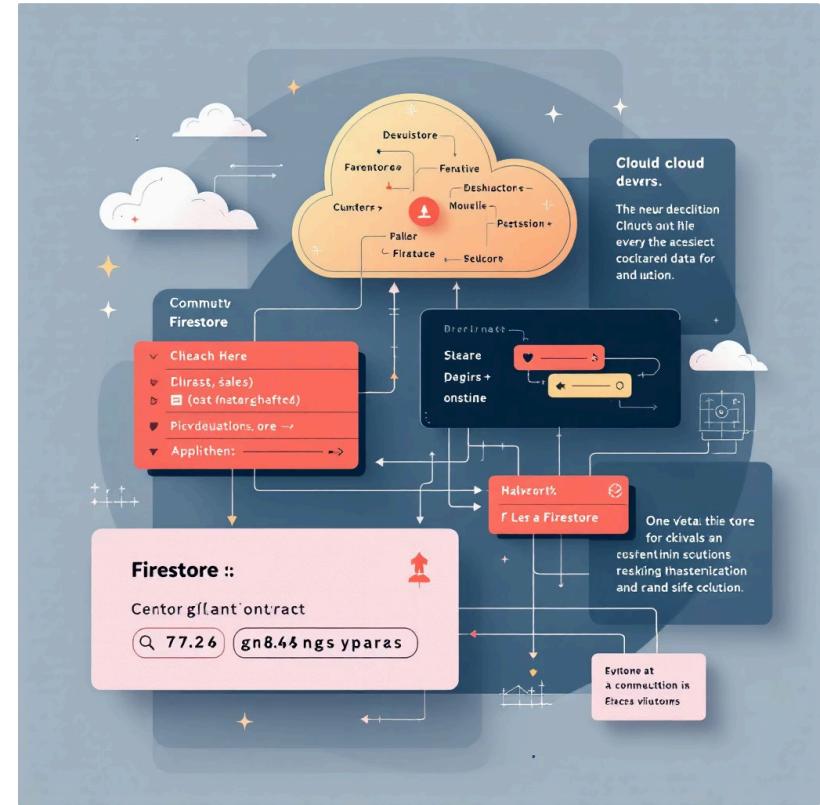


Modelo de Dados Flexível

O Cloud Firestore utiliza um modelo de dados NoSQL baseado em documentos, o que oferece uma flexibilidade incomparável para estruturar suas informações.

- **Documentos:** Unidades de dados que contêm campos mapeados para valores. Podem incluir objetos aninhados complexos e subcoleções.
- **Coleções:** Contêineres de documentos que ajudam a organizar os dados e facilitar consultas eficientes.
- **Subcoleções:** Permitem criar estruturas de dados hierárquicas e complexas que podem escalar à medida que seu banco de dados cresce.

Este modelo suporta diversos [tipos de dados](#), desde strings e números simples até objetos JSON.



Consultas Expressivas e Poderosas

No Cloud Firestore, você pode ir além das simples recuperações de dados, utilizando consultas para extrair informações específicas e relevantes de forma eficiente.

Filtros em Cadeia

Combine múltiplos filtros para refinar seus resultados, buscando documentos que correspondam a critérios complexos.

Filtragem e Classificação

Adicione cláusulas de classificação e limites para ordenar e paginar seus dados, garantindo uma experiência de usuário otimizada.

Consultas Superficiais

Recupere apenas os dados necessários no nível do documento, sem a sobrecarga de buscar coleções inteiras ou subcoleções aninhadas.



Atualizações em Tempo Real e Suporte Off-line



Sincronização Instantânea

Assim como o Realtime Database, o Cloud Firestore usa sincronização para manter os dados atualizados em qualquer dispositivo conectado, enviando apenas as alterações.



Experiência Off-line Robusta

O Cloud Firestore armazena em cache os dados ativos, permitindo que seu aplicativo funcione sem problemas, mesmo em ambientes sem conectividade. As mudanças locais são sincronizadas automaticamente quando a conexão é restabelecida.

Projetado para Escala Global

O Cloud Firestore herda a robustez e a confiabilidade da infraestrutura do Google Cloud, sendo projetado para suportar as maiores e mais exigentes cargas de trabalho do mundo.

01

Replicação Automática

Dados replicados automaticamente em várias regiões para alta disponibilidade e durabilidade.

02

Consistência Forte

Garantias de consistência que asseguram a integridade dos seus dados em todas as operações.

03

Operações Atômicas

Suporte a transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade) e operações em lote atômicas.

04

Infraestrutura Google

Beneficie-se da mesma infraestrutura que impulsiona os serviços do Google, garantindo desempenho e escalabilidade.



Compatibilidade com MongoDB

Para empresas que já utilizam MongoDB, o Cloud Firestore oferece uma transição suave e integração eficaz.

Na [**edição Enterprise**](#) do Cloud Firestore, é possível utilizar uma API compatível com MongoDB. Isso permite que você aproveite seu código, drivers, ferramentas e o vasto ecossistema de código aberto do MongoDB, simplificando a migração e o gerenciamento de dados.



Como o Cloud Firestore Funciona?

O Cloud Firestore é um banco de dados NoSQL hospedado na nuvem, acessível diretamente por SDKs nativos em diversas plataformas.



SDKs Nativos

Acesso direto para aplicativos Apple (iOS/macOS), Android e Web com SDKs otimizados.

Multi-Linguagem

Disponível para Node.js, Java, Python, Unity, C++, Go e APIs REST/RPC para desenvolvimento de servidor.

Modelo de Documentos

Organiza dados em documentos e coleções, permitindo estruturas hierárquicas flexíveis.

A consulta expressiva e o suporte a listeners em tempo real garantem que seus dados estejam sempre atualizados e facilmente acessíveis, independentemente da complexidade da sua aplicação.



Segurança e Acesso

A segurança dos seus dados é uma prioridade máxima no Cloud Firestore, com opções robustas para controle de acesso.



Firebase Authentication

Integração perfeita com o Firebase Authentication para gerenciar a identidade dos usuários em aplicativos móveis e web.



Cloud Firestore Security Rules

Defina regras de segurança granulares para Android, plataformas da Apple e JavaScript para controlar quem pode ler e escrever seus dados.



Identity and Access Management (IAM)

Para ambientes de servidor, utilize o IAM do Google Cloud para gerenciar permissões de acesso de forma centralizada e segura.

Caminho para Implementação

Começar a usar o Cloud Firestore é um processo direto, projetado para acelerar o desenvolvimento da sua aplicação.

1

Integre os SDKs

Adicione rapidamente os clientes do Cloud Firestore via Gradle, CocoaPods ou script de inclusão.

2

Proteja seus Dados

Use as Regras de Segurança do Cloud Firestore ou o IAM para proteger suas informações de acordo com o ambiente.

3

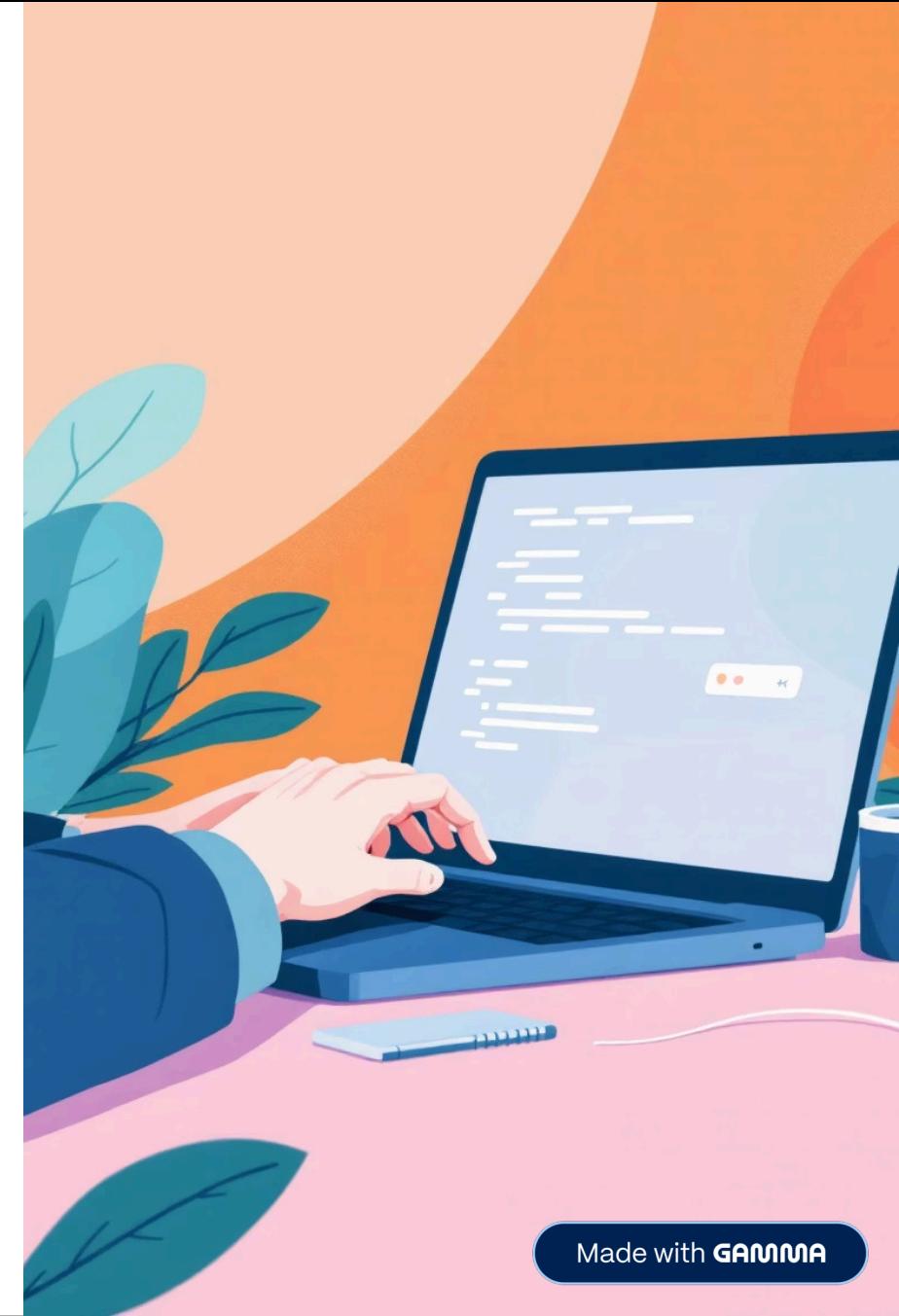
Adicione Dados

Crie documentos e coleções para começar a popular seu banco de dados com as informações da sua aplicação.

4

Receba Dados

Utilize consultas eficientes ou listeners em tempo real para recuperar e manter os dados atualizados em seus aplicativos.



Hora da Pratíca



Instalação das Dependências

```
8  
9   dependencies:  
10    flutter:  
11      sdk: flutter  
12    intl:  
13    firebase_core: ^2.24.2  
14    cloud_firestore: ^4.13.3
```

npm install -g firebase-tools

dart pub global activate flutterfire_cli

\$env:PATH += "\$HOME\AppData\Local\Pub\Cache\bin"

firebase login

flutterfire configure

firebase.google.com/products/firestore?hl=pt-br

Firebase Criar Mais Pesquisa / Português ... Blog Studio Ir para o console N

Cloud Firestore

Store and sync app data at global scale

Build robust mobile, web, and server applications at global scale. Effortlessly store, sync, and query your data in real time across devices, even offline, with this powerful NoSQL document database.

[Começar](#) [Ver documentação →](#)



[Criar](#)[Execução](#)[Mais](#)[Pesquisa](#)[Português](#) ...[Blog](#)[Ir para o console](#)[Fazer login](#)

Prototype, build & run modern, AI-powered experiences users love with Firebase, a platform designed to support you throughout your app development lifecycle.

Backed by Google and trusted by millions of businesses around the world.

[Get started in console](#)[Try Firebase Studio →](#)

CRIAÇÃO

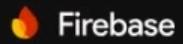
Lance seus produtos rapidamente e com segurança usando produtos que podem ser escalonados globalmente

Crie experiências com tecnologia de IA nos seus apps e acelere o desenvolvimento com infraestrutura gerenciada, com tecnologia do Google Cloud, para se concentrar no que é mais importante.

EXECUÇÃO

Execute seu app com confiança e ofereça a melhor experiência aos seus usuários

Lance, monitore e itere com ferramentas que ajudam você a otimizar a qualidade e a experiência do seu app.



Olá, Erick

É ótimo ter você no Firebase!

Vamos começar



Comece configurando um
Projeto do Firebase

Integre produtos do Firebase para turbinar seu app

Testar um app de exemplo



Criar um app do Flutter com tecnologia de IA

Implante um app de exemplo que mostre como a API Gemini Live, os comandos multimodais e a criação de imagens com o Nano Banana funcionam no Flutter.



Testar um app agêntico de barista

Implante um app de exemplo que use o Firestore, o Firebase Authentication e a chamada de função no Firebase AI Logic. Depois disso, estude o código no Firebase Studio.



Confira nosso projeto de demonstração (somente leitura)

Ver

Idioma — português (Brasil) ▾

Suporte — Termos — Política de Privacidade



Made with **GAMMA**

x Criar um projeto

Vamos começar nomeando o projeto[?]

Nome do projeto

notas-app

notas-app-8df11

Aceito os [Termos do Firebase](#)

Participo do [Programa para desenvolvedores do Google](#) para aperfeiçoar sua jornada com assistência de IA, recursos de aprendizado, selos de perfil e muito mais.

Já tem um projeto do Google Cloud?

[Adicionar o Firebase ao projeto do Google Cloud](#)

Continuar



Firebase

notas-app ▾

Visão geral do projeto

Categorias dos produtos

Criação

- App Check
- App Hosting
- Authentication
- Data Connect
- Extensions
- Firestore Database
- Functions
- Hosting
- Machine Learning
- Realtime Database
- Storage

Executar

Analytics

AI

Spark Sem custos (US\$ 0/mês) Fazer upgrade NOVO

Teste um app agêntico de barista com a tecnologia do Firebase AI Logic! Testar app de exemplo X

notas-app Plano Spark

+ Adicionar app

Olá, Erick

Este é seu projeto do Firebase!

O Gemini pode dar sugestões de serviços e soluções do Firebase para seu app

Próximas etapas com o Gemini

Ver documentos ↗

Fale sobre seu aplicativo
Descreva seu app, e o Gemini vai sugerir produtos para ajudar você a começar

Criar um back-end
O Firebase oferece muitos serviços de back-end, incluindo opções de bancos de dados SQL e NoSQL. Saiba quais serviços podem ser ideais para seu app.

Adicionar análises e monitoramento
Saiba mais sobre os produtos de monitoramento do Google Analytics e do Firebase

Adicione IA ao seu app
Saiba como integrar a IA ao seu app

Gerar receita com seu app

Hospedar um aplicativo da web

 **Firebase**

notas-app ▾

Visão geral do projeto 

Atalhos de projetos

Firestore Database 

Categorias dos produtos

Criação 

Executar 

Analytics 

AI 

Ferramentas de desenvolvimento relacionadas

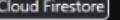
 **Firebase Studio** 

Spark Sem custos (US\$ 0/mês) Fazer upgrade 

 Cloud Firestore

Atualizações em tempo real, consultas eficientes, escalonamento automático e compatibilidade com o MongoDB

[Criar banco de dados](#)  [Pedir ao Gemini](#)

Saiba mais 

 [Por onde começar?](#)
Ver a documentação

 [Qual será o custo do produto "Cloud Firestore"?](#)
Consultar preços

 Introducing Cloud Firestore
Watch later Share

 Criar um banco de dados

1 Selecionar a edição

Edição Standard

Mecanismo de consulta simples com indexação automática. Para documentos de até 1 MiB.

Edição Enterprise

Mecanismo de consulta avançado compatível com o MongoDB. Para documentos de até 4 MiB. Apenas para drivers e ferramentas do MongoDB.

Não sabe qual é a melhor edição para você? [Compare as edições](#)

Avançar

2 ID e local do banco de dados

3 Configurar

```
lib > main.dart > ...
1 import 'package:flutter/material.dart';
2 import 'package:firebase_core/firebase_core.dart';
3 import 'firebase_options.dart'; // NOVO IMPORT!
4 import 'views/app.dart';
5
6 Run | Debug | Profile
7 void main() async {
8     WidgetsFlutterBinding.ensureInitialized();
9
10    // Inicialização correta para todas as plataformas (incluindo Web)
11    await Firebase.initializeApp(
12        options: DefaultFirebaseOptions.currentPlatform,
13    );
14
15    runApp(const NotasApp());
}
```

```
lib > viewmodels > NotaViewModel.dart
1 // viewmodels/notaviewmodel.dart
2 import 'package:flutter/material.dart';
3 import 'package:cloud_firestore/cloud_firestore.dart'; // 🚨 NOVO: Import do Firestore
4 import 'package:provider/provider.dart';
5 import '../models/nota.dart';
6 // Remova: os imports de http, dart:convert e uuid (não são mais necessários)
7
8 class NotaViewModel extends ChangeNotifier {
9   // Referência ao Firestore
10  final FirebaseFirestore _db = FirebaseFirestore.instance;
11
12  // Nome da coleção no seu banco de dados
13  final String _collectionName = 'notas';
14  List<Nota> _notas = [];
15
16  List<Nota> get notas => List.unmodifiable(_notas);
17
18  // O construtor carrega os dados ao iniciar
19  NotaViewModel() {
20    read();
21  }
22
23  // --- CREATE (Criar) ---
24  void create(String title) async {
25    // 1. Cria um objeto Nota (o ID temporário será substituído)
26    Nota novaNota = Nota("nota", title, DateTime.now());
27
28    // 2. Adiciona ao Firestore e obtém a referência do novo documento (docRef)
29    await _db.collection(_collectionName).add(novaNota.toJson()).then((docRef) {
30      // 3. Atualiza o documento no Firestore com o ID gerado (docRef.id)
31      docRef.update({'id': docRef.id});
32    });
33
34    // 4. Recarrega a lista para atualizar a UI
35    await read();
36    notifyListeners();
37}
```

```
38
39 // --- DELETE (Deletar) ---
40 void delete(String id) async {
41     // 1. Atualiza a lista local primeiro (feedback rápido)
42     _notas.removeWhere((e) => e.id == id);
43     notifyListeners();
44
45     // 2. Deleta o documento no Firestore pelo ID
46     await _db.collection(_collectionName).doc(id).delete().catchError((e) {
47         print("Erro ao deletar no Firestore: $e");
48     });
49 }
50 }
```

```
50
51 // --- UPDATE (Atualizar) ---
52 void update(Nota nota) async {
53     // 1. Atualiza a lista local (feedback rápido)
54     var index = _notas.indexWhere((e) => e.id == nota.id);
55     if (index != -1) {
56         _notas[index] = nota;
57     }
58     notifyListeners();
59
60     // 2. Atualiza o documento no Firestore.
61     await _db
62         .collection(_collectionName)
63         .doc(nota.id) // Usa o ID da nota para localizar
64         .update(nota.toJson()) // Atualiza com os novos dados
65         .catchError((e) {
66             print("Erro ao atualizar no Firestore: $e");
67         });
68 }
69
70 // --- READ (Ler) ---
```

```
9 // --- READ (Ler) ---
1 Future<void> read() async {
2   try {
3     Delegar ao agente
4     // 1. Busca todos os documentos na coleção
5     QuerySnapshot snapshot = await _db.collection(_collectionName).get();
6
7     // 2. Mapeia os documentos do Firestore para objetos Nota do Dart
8     _notas =
9       snapshot.docs.map((doc) {
10       Map<String, dynamic> data = doc.data()! as Map<String, dynamic>;
11
12       // O ID do documento é o ID da nota
13       data['id'] = doc.id;
14
15       // Converte o Timestamp do Firestore para o formato DateTime esperado pelo Dart
16       if (data['date'] is Timestamp) {
17         data['date'] =
18           (data['date'] as Timestamp).toDate().toIso8601String();
19       }
20
21       return Nota.fromJson(data);
22     }).toList();
23   } catch (e) {
24     print("Erro ao ler dados do Firestore: $e");
25     _notas = [];
26   } finally {
27     notifyListeners(); // Atualiza a UI
28   }
29 }
```

CRUD

