

GDG Quito · BWA! Latam

Firestore vs Supabase

Documents or tables: picking your backend in *2026*.



Hi, I'm

Sebastian Alvarez

Sr Full-Stack Engineer
· GDG Quito Organizer

I build backends, cloud infrastructure and web products. Firebase is behind almost all my personal projects, and Supabase has earned a place in the ones that call for Postgres.

● GDG Quito

● Quito, Ecuador

@listerineh

Agenda

- **01** What is a BaaS?
- **02** Firebase
- **03** Supabase
- **04** Head to head
- **05** Which one?

01

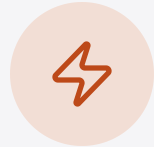
What is a BaaS?

Backend-as-a-Service: the backend you don't have to build.

Backend-as-a-Service

- Database, authentication, files and functions **ready to use**
- An ecosystem to build, ship and run apps
- Not just a database
- Your team writes product, not infrastructure

The promise



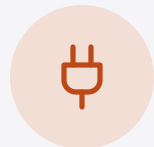
Speed

Ship features in days, not weeks.



Scale

Traffic grows and you don't run servers.



Integration

Auth, data, files and functions that talk to each other.



Security

Access rules live next to the data.

The real difference is the **data model**.

Firestore gives you documents; Supabase gives you Postgres. Everything else follows from that.

02

Firebase

Google's platform for apps.

Firebase in 2026

- Part of Google since 2014, integrated with Google Cloud
- **Firestore**: a document database with real-time and offline mode
- **Authentication, Cloud Storage, Cloud Functions and Hosting**
- **Firestore Enterprise**: pipelines with aggregations (GA in April 2026)
- **SQL Connect** (formerly Data Connect): managed PostgreSQL on Cloud SQL

Firestore: documents and collections

JAVASCRIPT

```
users/{userId} {  
  name: "Sebastian",  
  settings: { theme: "dark", notifications: true }  
}  
  
users/{userId}/posts/{postId} {  
  title: "My first post", tags: ["tech", "firebase"], likes: 42  
}
```

No fixed schema: data is nested and duplicated to avoid joins.

Real-time with Firestore

JAVASCRIPT

```
const q = query(
  collection(db, "rooms", roomId, "messages"),
  orderBy("timestamp", "desc"), limit(50),
);

onSnapshot(q, (snap) => {
  setMessages(snap.docs.map((d) => ({ id: d.id, ...d.data() })));
});
```

A real-time chat: it works offline and syncs on its own when the connection comes back.

Pipelines in Firestore Enterprise

JAVASCRIPT

```
const top = await db.pipeline()  
  .collection("recipes")  
  .unnest(field("tags").as("tag"))  
  .aggregate({ accumulators: [countAll().as("n")], groups: ["tag"] })  
  .sort(field("n").descending())  
  .limit(10)  
  .execute();
```

The 10 most used tags, computed on the server: complex queries used to be Firestore's weak spot.

Authentication and rules

JAVASCRIPT

```
const { user } = await signInWithPopup(auth, new GoogleAuthProvider());

// firestore.rules
match /users/{userId} {
  allow read, write: if request.auth.uid == userId;
}
```

Email, Google, Apple, GitHub, phone, anonymous... and security rules use the same session.

03

Supabase

Postgres with everything an app needs around it.

Supabase in 2026

- Open source and built on **PostgreSQL**
- Auth, Storage, **Edge Functions** (Deno) and Realtime around Postgres
- REST and GraphQL APIs generated from your schema
- Extensions: **pgvector** for AI, PostGIS, pg_cron
- Can be self-hosted

Postgres: tables and relations

SQL

```
create table posts (  
  id uuid primary key default gen_random_uuid(),  
  user_id uuid references users (id) on delete cascade,  
  title text not null,  
  tags text[],  
  created_at timestamptz default now()  
);
```

A defined schema, foreign keys, joins and ACID transactions.

Row Level Security

SQL

```
alter table posts enable row level security;  
  
create policy "everyone edits their own"  
  on posts for update  
  using (auth.uid() = user_id);
```

Authorization lives in the database and applies to the API, Realtime and Storage.

Real-time with Supabase

JAVASCRIPT

```
supabase
  .channel("posts")
  .on("postgres_changes",
    { event: "*", schema: "public", table: "posts" },
    (payload) => console.log(payload))
  .subscribe();
```

Postgres changes, Broadcast and Presence. No offline cache.

Supabase in numbers

13M+

developers

\$10.5B

valuation

70%

of new databases,
created by AI

04

Head to head

Feature by feature.

Documents vs tables

Firestore

- No schema: change it whenever you like
- Nested, duplicated data
- Native offline mode
- Complex queries cost more

Postgres

- Schema and foreign keys
- Joins and ACID transactions
- Full SQL and extensions
- No offline mode

A join is one query

SQL

```
select o.id, u.name, p.name as product, oi.quantity * oi.price as total
from orders o
join users u on u.id = o.user_id
join order_items oi on oi.order_id = o.id
join products p on p.id = oi.product_id
where o.status = 'completed';
```

In Firestore it's several reads and stitching the data together on the client.

Feature by feature

- **Auth:** a tie. Firebase has more polished SDKs; Supabase's RLS is more expressive
- **Storage:** both solid; Supabase applies RLS to files too
- **Functions:** Firebase wins on ecosystem; Supabase on edge latency
- **Real-time:** Firebase, thanks to offline sync
- **Queries and reports:** Supabase, with full SQL
- **AI:** Supabase with pgvector; Firestore has vector search too

Pricing

Firestore

- **Spark**, free: 50,000 reads and 20,000 writes a day
- **Blaze**: pay per operation, ~\$0.03 per 100,000 reads
- Careful: a listener on a busy collection can surprise you on the bill

Supabase

- **Free**: 500 MB database, 50,000 monthly active users
- **Pro** from \$25 a month: 8 GB database and 100 GB of files
- You pay for resources, not operations: easier to predict

And in 2026, **Firestore** **speaks Postgres too.**

SQL Connect: Cloud SQL for PostgreSQL with real-time, offline caching and native SQL.

05

Which one?

It depends on your app, not on hype.

When I pick each one

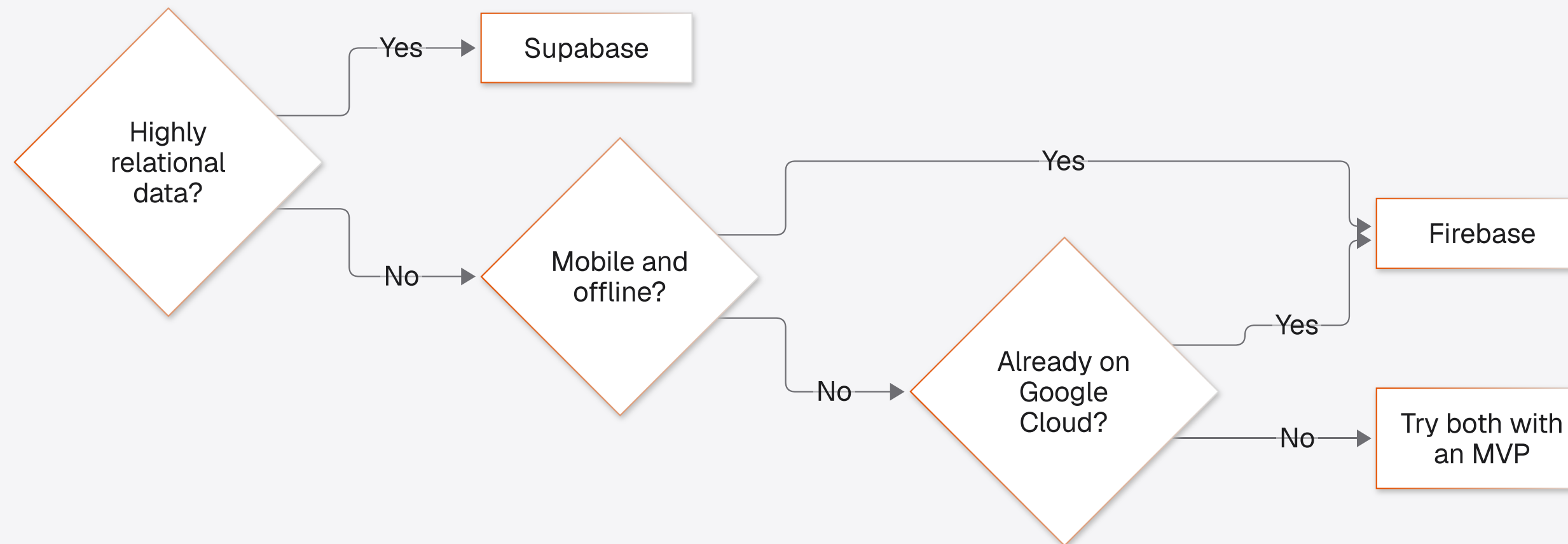
Firestore

- Mobile apps that must work offline
- MVPs without designing a schema
- Projects already on Google Cloud
- Real-time collaboration

Supabase

- Relational data: e-commerce, SaaS, CRM
- Reports and analytics with SQL
- AI with pgvector
- Open source and self-hosting

One way to decide



And nothing stops you from combining them, each doing what it does best.

Neither is **better** in general.

It depends on your use case, what your team knows and your architecture.

More questions? Let's connect

The full comparison is on the blog:
listerineh.dev/blog/firebase-vs-supabase.

linkedin.com/in/listerineh

GDG Quito

listerineh.dev