

DATABASE / AUTH / APIS / REALTIME

SUPABASE

THE POSTGRES DEVELOPMENT PLATFORM



A real Postgres database surrounded by integrated tools that help an application store data, manage accounts, expose APIs, secure rows, handle files and react to live events.

FOUNDATION	MANAGEMENT	IDENTITY	APP ACCESS
POSTGRES	STUDIO + SQL	AUTH + RLS	GENERATED APIS

01 / 60-SECOND EXPLANATION

A BACKEND WITHOUT HIDING THE DATABASE.

Your frontend contains screens and interactions. Supabase provides the backend systems those screens need: structured data, accounts, permissions, files, server-side code and live events.

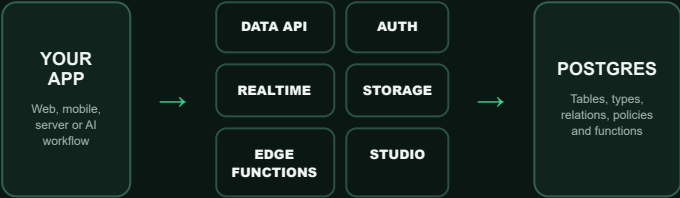
The core is **REAL POSTGRES.**

You can use the visual dashboard, normal SQL, client libraries, REST, GraphQL, an ORM or a direct database connection.

NOT MAGIC: Supabase removes repetitive setup. You still own the schema, permissions, indexes, backups strategy, costs and production security.

02 / SYSTEM ARCHITECTURE

ONE DATABASE. MULTIPLE WAYS IN.



Auth issues a user session. The client sends it with a request. Postgres roles, grants and Row Level Security decide which records that request may touch.

03 / THE REAL FIVE-MINUTE DEMO

CREATE AN ACCOUNT → CONNECT ITS DATA → SECURE IT → USE IT

STEP 01 ACCOUNT Create one fake user. Auth assigns a unique UUID and session identity.	STEP 02 PROFILE A trigger automatically creates the user's matching public profile row.	STEP 03 EDIT The signed-in user changes their display name through a one-file app.	STEP 04 ADMIN + LIVE SQL changes the account plan; Realtime pushes it back to the app.
---	--	---	---

A TINY, TYPED TABLE

```
id          uuid          references auth.users(id)
display_name text          user-editable
plan        text          admin-controlled
updated_at  timestampz default now()
```

04 / WHAT THIS STACK CAN POWER

SAAS PRODUCTS Accounts, teams, subscriptions, permissions and business records.	MARKETPLACES Profiles, listings, orders, files and role-based access.	LIVE APPS Chat, dashboards, presence, notifications and collaborative state.	AI SYSTEMS Users, jobs, outputs, vector data and server-side workflows.
--	--	---	--

FROM ACCOUNT TO AUTHORIZED ROW

The most important Supabase concept is not the Table Editor. It is the path from an authenticated user to data that Postgres is willing to return.

01

USER SIGNS IN

Supabase Auth verifies the configured login method.

02

SESSION JWT

The client receives a signed token containing identity and role claims.

03

QUERY ARRIVES

The Data API maps the request to a Postgres role.

04

RLS FILTERS

Policies allow only rows the current user may access.

DATA ACCESS / JAVASCRIPT

THE TABLE BECOMES AN API.

Supabase reflects supported database objects into a Data API. Client libraries provide a clean interface while Postgres remains the source of truth.

```
const { data, error } = await supabase
  .from('profiles')
  .select('display_name, plan, updated_at')
  .eq('id', user.id)
  .single()
```

CRUD is fast to start, but authorization is still enforced through grants and policies. The query is easy; deciding who may run it is the serious part.

SECURITY BOUNDARY

KEYS DO DIFFERENT JOBS.

PUBLISHABLE KEY

Designed for client use when permissions and RLS are correctly configured. It identifies the project; it does not grant unlimited access.

SERVICE-ROLE KEY

Privileged and able to bypass RLS. Keep it on trusted servers. Never expose it in browser code, screenshots or video.

WHERE SUPABASE SHINES

GOOD FIT WHEN...

- You want relational Postgres data rather than a proprietary document store.
- You need database, Auth, Storage and live features to work together.
- You want to move quickly but keep SQL, migrations and standard tooling.
- Your frontend benefits from generated APIs and user-aware database rules.

WHAT THE PLATFORM DOES NOT REMOVE

YOU STILL OWN...

- Schema design, relationships, constraints and useful indexes.
- RLS policies, grants, secret handling and application authorization.
- Capacity planning, egress, observability and recovery requirements.
- Testing migrations and separating staging from production.

EVERGREEN PRODUCTION CHECKLIST

- ✓ Keep production and staging separate.
- ✓ Grant only the operations each role needs.
- ✓ Test policies as signed-out and signed-in users.
- ✓ Enable RLS on exposed tables.
- ✓ Never ship privileged secrets to the client.
- ✓ Review current limits, backups and pricing.

LIVE SOURCES

[Architecture](#) [Tables](#) [Data API](#) [Auth](#) [RLS](#) [Realtime](#) [Pricing](#)

CLIENT LIBRARY

Fast frontend access. Pair the publishable key with carefully tested grants and RLS.

YOUR SERVER

Add trusted business logic, third-party secrets and privileged operations away from the browser.

DIRECT POSTGRES

Use connection strings, poolers, SQL tools and ORMs from trusted infrastructure.

BUILD WITH REAL DATA. TRADE WITH A PLAN.

Visit the AlphaStack Bybit referral link to support the channel.

[TRADE ON BYBIT](#)