

1 Branch 0 Tags Go to file About Code

Fix README typo (#3) ✓

3329347 · 23 minutes ago

- .github 3 hours ago
client 3 hours ago
server 3 hours ago
.gitignore 3 hours ago
LICENS... 3 hours ago
READM... Fix README t... 23 minutes ago

The realtime client-side database

instantdb.com

- Readme
Apache-2.0 license
Activity
Custom properties
242 stars
4 watching
5 forks
Report repository

README Apache-2.0 license

releases published



chat 83 online Stars 231

Get Started · Examples · Try the Demo · Docs · Discord

Packages

No packages published

Contributors 6



Languages

- Clojure 52.7%
TypeScript 33.7%
JavaScript 12.5%
CSS 0.4%
PLpgSQL 0.2%
Makefile 0.2%
Other 0.3%

Instant is a client-side database that makes it easy to build real-time and collaborative apps like Notion or Figma.

You write relational queries in the shape of the data you want and Instant handles all the data fetching, permission checking, and offline caching. When you change data, optimistic updates and rollbacks are handled for you as well. Plus, every query is multiplayer by default.

We also support ephemeral updates, like cursors, or who's online. Currently we have SDKs for Javascript, React, and React Native.

How does it look? Here's a barebones chat app in about 10 lines:

```
// 📄 📄 📄 Real-time Chat
// -----
// * Updates instantly
// * Multiplayer
// * Works offline
function Chat() {
  // 1. Read
  const { isLoading, error, data } = useQuery({
    messages: {},
  });

  // 2. Write
  const addMessage = (message) => {
    transact(tx.messages[id()].update(message));
  }

  // 3. Render!
  return <UI data={data} onAdd={addMessage} />
}
```



Want to see for yourself? [try a demo in your browser.](#)

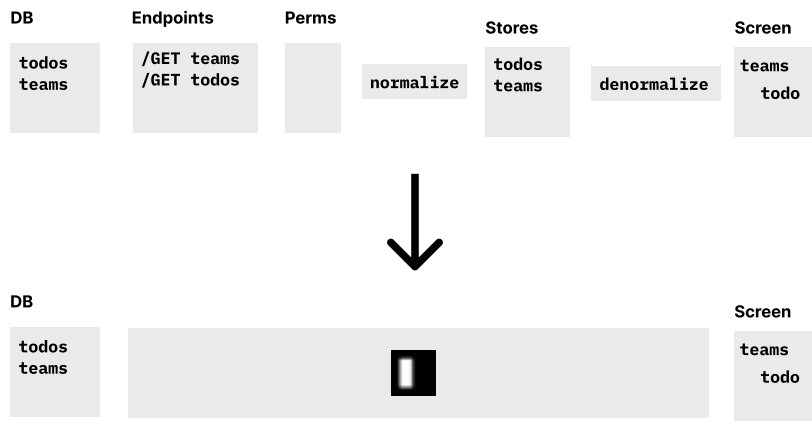
Motivation

Writing modern apps are full of schleps. Most of the time you start with the server: stand up databases, caches, ORMs, and endpoints. Then you write client-side code: stores, selectors, mutators. Finally you paint a screen. If you add multiplayer you need to think about stateful servers, and if you support offline mode, you need to think about IndexedDB and transaction queues.

To make things worse, whenever you add a new feature, you go through the same song and dance over and over again: add models, write endpoints, stores, selectors, and finally the UI.

Could it be better?

In 2021, **we realized that most of the schleps we face as UI engineers are actually database problems in disguise.** (We got into greater detail [in this essay](#))

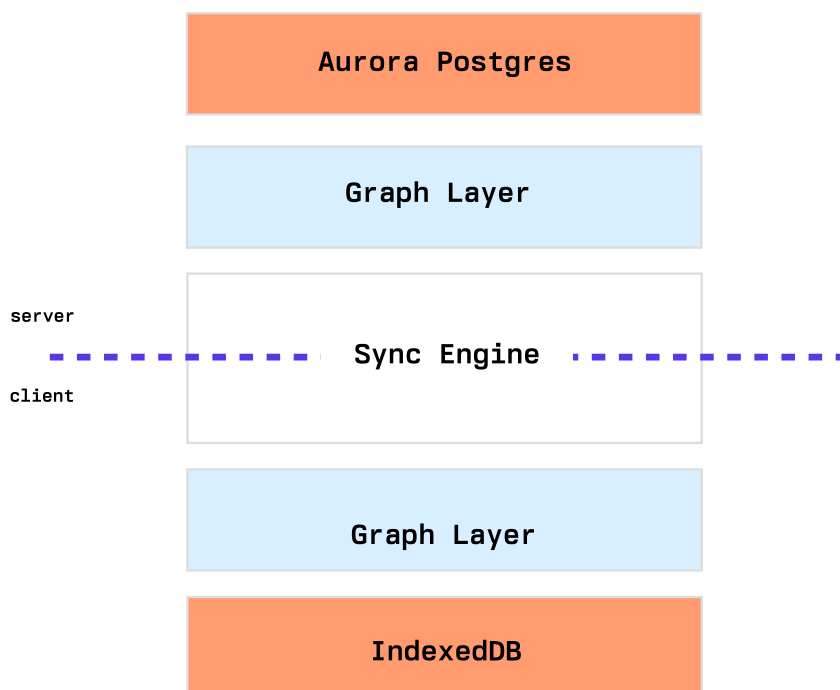


If you had a database on the client, you wouldn't need to think about stores, selectors, endpoints, or local caches: just write queries. If these queries were multiplayer by default, you wouldn't have to worry about stateful servers. And if your database supported rollback, you'd get optimistic updates for free.

So we built Instant. Instant gives you a database you can use in the client, so you can focus on what's important: building a great UX for your users, and doing it quickly.


Architectural Overview

Here's how Instant works at a high level:



Under the hood, we store all user data as triples in one big Postgres database. A multi-tenant setup lets us offer a free tier that never pauses.

A sync server written in Clojure talks to Postgres. We wrote a query engine that understands datalog and [InstaQL](#), a relational language that looks a lot like GraphQL:

```
// give me all users, their posts and comments   
{ users: { posts: { comments: {} } } }
```

Taking inspiration from [Asana's WorldStore](#) and [Figma's LiveGraph](#), we tail postgres' WAL to detect novelty and invalidate relevant queries.

For the frontend, we wrote a client-side triple store. The SDK handles persisting a cache of recent queries to IndexedDB on web, and AsyncStorage in React Native.

All data goes through a permission system powered by Google's [CEL library](#).

Getting Started

The easiest way to get started with Instant is by signing up on [instantdb.com](#). [You can create a functional app in 5 minute or less.](#)

If you have any questions, you can jump in on our [discord](#).

Contributing

You can start by joining our [discord](#) and introducing yourself. Even if you don't contribute code, we always love feedback.

If you want to make changes, start by reading the [client](#) and [server](#) READMEs. There you'll find instructions to start Instant locally.