IDENTITY AND ACCESS

ri-auth

A production sign-in and account service with its own repository, its own database and a client package, built once on purpose so the next product does not have to build it again.

98	8	3	6
Tests over auth, recovery and deletion	Tables in the schema	Sign-in methods on one account	Two-factor endpoints

THE PROBLEM

Every product needs accounts, and building accounts into a product is how you end up building them again for the next one. The second time costs as much as the first, and by then there are two password reset flows, two deletion policies and two sets of mistakes.

QuietWatcher needed identity. Rather than put it in the app, we built it as a service with its own repository, its own database, its own OpenAPI contract and a client library the app consumes like any other dependency.

The scope was held on purpose. ri-auth does accounts, credentials, second factors, recovery, policy acceptance and deletion. Billing, organizations and entitlements live in the product that needs them, because those are the parts that differ from product to product and an identity service that carries them stops being reusable.

WHAT IT DOES

- **Three ways in, one account.** Email and password, Apple or Google. Identities are a separate table from users, which is what makes one person with a password, an Apple identity and a Google identity a single account rather than three.
- **Policy versions bound at signup.** Signup will not complete unless the client echoes back the identifiers of the policy versions it just fetched, so nobody is silently bound to a stale terms or privacy version.
- **Passwords hashed with Argon2**, sessions issuing access and refresh tokens.
- **Two-factor, time-based, in six calls:** begin enrolment, confirm, verify at login, disable, regenerate backup codes, report status. Backup codes are stored separately from the secret.
- **Recovery for the paths people actually hit.** Password reset request and confirm, email verification request and confirm, and Apple's private relay flow on its own callback.

THE TWO ENDPOINTS THAT SHOW WHAT KIND OF SERVICE THIS IS

- **An honest methods endpoint.** A client can ask which sign-in routes an account actually has, so the login screen does not offer a password to an account that only ever used Apple. That is a small call that removes a whole category of support ticket.
- **Deletion as a first-class endpoint, not a support ticket.** A soft delete with defined retention: sensitive material removed at once, the rest held for a fixed window and then hard-deleted, and the address free to register again afterwards. That combination is what an app store reviewer asks for and what a privacy policy has to be able to describe.

HOW IT IS BUILT

FastAPI, SQLAlchemy 2 against async Postgres, Alembic migrations confined to its own schema, Argon2, a standard TOTP library and JOSE for tokens. Three packages in one repository: the API, the Dart client and the shared schema.

FastAPI | Python | SQLAlchemy 2 | PostgreSQL | Alembic | Argon2 | TOTP | JOSE | Dart client

STATE TODAY

In production, serving QuietWatcher's authentication, with 98 tests covering the auth flow, two-factor, recovery, OAuth verification, deletion and retention.

A Makefile means a new developer runs four lines and has a working environment, which is the difference between a service another team can adopt and one only its author can run.

WHY THIS IS WORTH WEEKS TO A BUYER

Identity is tedious to build, dangerous to get wrong and expensive to retrofit. The parts most teams skip, policy versioning at signup, backup codes, a deletion path with a written retention policy behind it, are already built and already tested.

WHAT WE CAN START ON

A login and account layer for a product that does not have one yet. An identity service pulled out of an application that has outgrown it. Or a review of the account, recovery and deletion paths you already ship.

512-900-4871 · div42.com · roblesindustries.com