APPLICANT TRACKING SYSTEM

TA-ATS + auth_mcp

A microservices ATS with a central authentication & MFA service

Architecture · Login flow · Setup & run

OVERVIEW

What is TA-ATS?

1

Microservices

4 independent services + 2 databases — each scales and deploys on its own.

2

Central auth

auth_mcp verifies passwords and runs MFA for any project that plugs in.

3

Role-based access

Groups + permissions decide what each user can see and do.

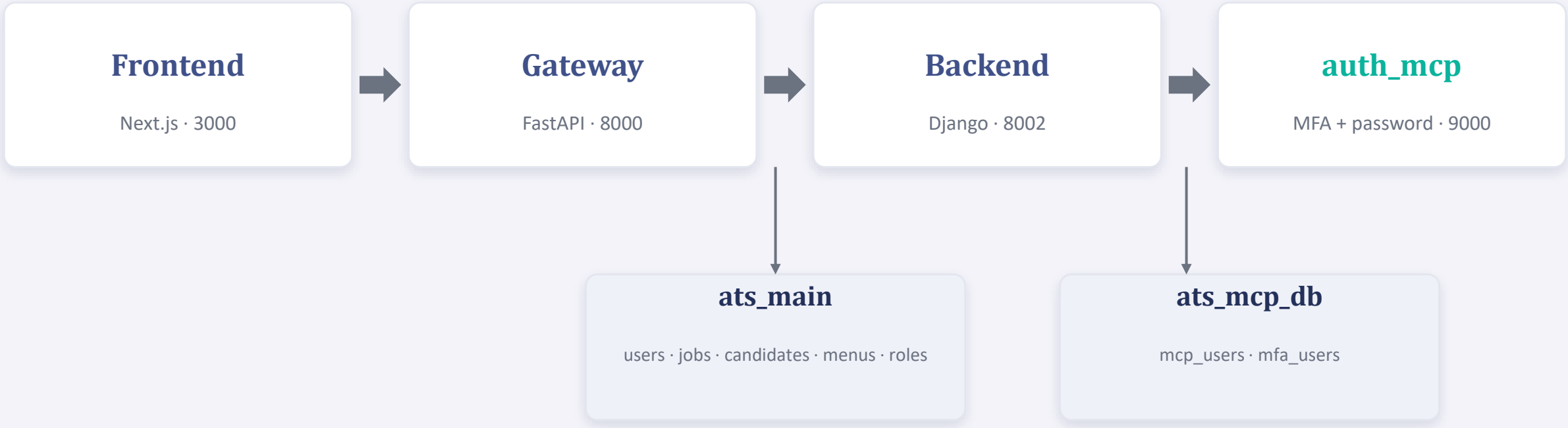
4

Dynamic UI

Sidebar menus are stored in the DB and filtered per user's permissions.

ARCHITECTURE

The moving parts



Backend owns ats_main. auth_mcp owns ats_mcp_db. They talk only through MCP tool calls — never each other's DB.

THE AUTH SERVICE

Why a separate auth_mcp?

Reusable	One auth + MFA service any project can plug into — no duplicated login code.
Never stores passwords	It keeps only hashes in mcp_users and verifies them; the app checks nothing itself.
Owns the MFA policy	Admin picks one method (Email OTP / Authenticator); the MCP enforces it everywhere.
Self-healing	If a user is valid locally but missing in the MCP, the backend syncs the hash automatically.

HOW IT WORKS

Login flow, step by step

1. User submits email + password on the Next.js login screen.
2. Request travels Frontend → Gateway (8000) → Backend (8002).
3. Backend asks auth_mcp to verify the password (mcp_users in ats_mcp_db).
4. If password missing but valid locally → self-heal: sync hash to the MCP.
5. If MFA is on, auth_mcp emails / checks the OTP (mfa_users).
6. Backend issues JWT tokens → dashboard loads → sidebar filtered by permissions.

DATA

Two databases, clear roles

ats_main

The app's master data

- Users & accounts (source of truth)
- Jobs, clients, candidates
- Menus, groups & permissions
- Audit logs, password policy

ats_mcp_db

The MCP's private store

- mcp_users — password-hash mirror
- mfa_users — MFA enrolment & OTP
- Rebuilt / re-synced from ats_main
- Backend never queries it directly

CAPABILITIES

Key features

Secure login + MFA

Email OTP or Authenticator, enforced centrally.

Roles & permissions

Groups, per-user grants, admin-managed via UI.

Dynamic sidebar

DB-driven menus, filtered by permission.

Forgot / reset password

3-step email OTP flow.

Password policy

Expiry, history, strength rules.

Professional UI

Velzon theme — light, Poppins, indigo.

SETUP — PART 1

Prerequisites & databases

1 · Install once

- Python 3.12+
- Node.js 20+
- PostgreSQL 16 / 18

Folder layout

```
python\
   auth_mcp\
   TA-ATS-interns\   (backend, gateway, frontend)
```

2 · Create the databases

```
psql -U postgres -c "CREATE DATABASE ats_main;"
psql -U postgres -c "CREATE DATABASE ats_mcp_db;"
```

Tables are created automatically later by migrations — you only create the two empty databases here.

SETUP — PART 2

Install & run each service

MCP · auth_mcp\ · 9000

```
python -m venv venv
.\venv\Scripts\activate
pip install -r requirements.txt
python server.py
```

Backend · backend\ · 8002

```
pip install -r requirements\base.txt
python manage.py migrate
python manage.py createsuperuser
python manage.py runserver 8002
```

Gateway · gateway\ · 8000

```
pip install -r requirements.txt
uvicorn app.main:app --reload --port 8000
```

Frontend · frontend\ · 3000

```
npm install
npm run dev
```

Tip: run migrate on a fresh clone AND after pulling — skipping it causes 500s. Or just double-click run.bat.

START ORDER — EVERY TIME

Run it

1. PostgreSQL
2. MCP (9000)
3. Backend (8002)
4. Gateway (8000)
5. Frontend (3000)

Open the app

App	http://localhost:3000
API docs	http://localhost:8002/api/docs/
Health	http://localhost:8000/health
Login	the superuser you created