Escalada App - One-Page Summary

What It Is

Escalada is a real-time climbing competition management app with a React frontend, a FastAPI backend, and a shared core logic package. It manages category boxes, live scoring, timers, and public-facing live/ranking views.

Who It's For

- Primary persona: competition operators (admin + judges) running live climbing events, with spectators using public read-only views.

What It Does

- Runs multi-box competition control with route init, timer control, progress updates, and score submission.
- Provides dedicated interfaces: Control Panel (admin), Judge Page (per box), Contest Page (display), and public hub/live/rankings pages.
- Streams real-time updates over authenticated per-box WebSockets and a public WebSocket feed.
- Persists box state in JSON files and records append-only audit events (NDJSON).
- Supports auth roles (admin, judge, viewer, spectator) with JWT and per-box access control.
- Includes admin backup/restore plus CSV/official export endpoints for operations.

How It Works (Repo-Evidenced Architecture)

- UI (`escalada-ui`): React + Vite app with routes for admin, judge, contest display, rankings, and public pages (`src/App.tsx`).
- API (`escalada-api`): FastAPI app wires routers for live commands, auth, backup, audit, public data, health, ops (`escalada/main.py`).
- Core (`escalada-core`): pure command/state engine (`apply_command`) validates and transitions contest state; API orchestrates persistence/broadcast.
- Persistence: JSON storage (`data/boxes/*.json`, `data/events.ndjson`) with per-box locks and atomic writes (`escalada/storage/json_store.py`).
- Data flow: UI action -> `POST /api/cmd` -> core state transition -> save/audit -> broadcast via `/api/ws/{box_id}` and public WS -> UI snapshot/echo update.

How To Run (Minimal)

- Backend: `cd escalada-api && poetry install && poetry run pip install -e ../escalada-core`
- Start API: `poetry run uvicorn escalada.main:app --host 0.0.0.0 --port 8000 --workers 1`
- Frontend: `cd ../escalada-ui && npm install && npm run dev`
- Open UI: `http://localhost:5173` (API expected at `http://localhost:8000`).
- Docker/Postgres deployment path: Not found in repo (backend README states JSON-storage mode only).