# Architecture Overview

This document captures the current end-to-end architecture of the Telco ADB Automation stack (Electron + React + FastAPI + ADB). It should be treated as the canonical reference when aligning design decisions, onboarding new contributors, or planning large initiatives (scheduler, live logs, multi-session locker).

---

## 1. System Context

```
Electron Shell (UI)
• React renderer
• Preload bridge

           IPC


FastAPI Backend
• Modules API
• Workflows engine
• REST + WebSockets

             ADB commands / Scripts


Android Devices (USB)
• ADB server
• Telco modules scripts
```

Supporting services (to be plugged in):

- Notification channels (mail/Slack/webhooks) for alerting.
- Scheduler service (cron-like worker) to trigger workflows automatically.
- Storage backend (today local file caches; roadmap includes SQLite/Postgres for shared deployments).

---

## 2. Component View

| Layer | Component | Description |
|---|---|---|
| Electron | `main.ts`, `preload.ts` | Boots the backend (PyInstaller bundle in prod, `simple-server.py` in dev), exposes secure `window.electronAPI` (backend URL, app version, telemetry). |
| Frontend | React 18 app (MUI + Chakra theme) | Pages: Dashboard, Device Manager, Workflows, Modules/TestModules, Reports. Uses lazy routes and shared hooks (`useDevices`, `useWorkflows`). |
| Backend | FastAPI (`simple-server.py`) | REST endpoints for devices, modules, workflows; WebSocket service for live device updates; orchestrates ADB via `modules/telco_modules.py`. |
| Module runtime | Telco modules | Implements ADB routines (voice call test, ping, airplane mode, etc.), used by both standalone module runs and workflows. |
| Scheduling (future) | Scheduler worker | Planned service that reads cron-like definitions, enqueues workflow executions, persists run history. |
| Telemetry/Storage | Local storage + JSON caches | Device metadata, workflow drafts, activity logs live in browser localStorage today. Registry/activity utilities added for Device Manager history. |

Communication patterns:

- React  FastAPI via REST (`fetch`), with WebSockets (`websocket.ts`) for device updates.
- FastAPI  ADB devices via `subprocess` calls within module executors.
- Electron main process monitors backend stdout/stderr to surface failures.

## 3. Data Flow: Workflow Execution

1. User selects devices on Dashboard.
2. Workflows page posts to `/api/workflows/run` with workflow ID + device list.
3. Frontend sets up `AbortController`, run state, and loops through modules.
4. For each module, frontend calls `/api/modules/{module_id}/execute` (per device). Backend executes corresponding Telco module synchronously. When all devices resolve, next module starts.
5. Device Manager logs per-device activity (local storage) to populate timelines and analytics.
6. Stop/Pause signals abort the controller and set cancellation flags so the loop exits cleanly.

---

## 4. Key Cross-Cutting Concerns

### 4.1 Telemetry & Logging

- Electron logs backend stdout/stderr.
- Frontend telemetry client sends `dashboard_viewed`, `search_used`, etc. via `navigator.sendBeacon` or Electron IPC.
- Device activity logs (workflows/modules) recorded client-side (`deviceActivity.ts`), enabling Device Manager timelines.

### 4.2 Error Handling

- Backend returns structured JSON (`success`, `error`, `message`). Frontend surfaces via snackbars.
- Workflow runner catches aborts vs. real failures to differentiate "Cancelled" vs "Failed".
- Device cards degrade gracefully (historical registry entries are marked and actions disabled).

### 4.3 Security

- Preload layer isolates renderer from Node APIs.
- CSP warning noted in dev (Electron security checklist to be applied before packaging).
- Future work: authentication/authorization, secret management for notifications.

---

## 5. Roadmap Hooks

| Feature | Architectural Touchpoints |
|---|---|
| Scheduler service | Backend worker (Celery/APScheduler) + DB tables (`scheduled_runs`, `run_history`), new API endpoints + UI calendar. |
| Live ADB logs | Dedicated WebSocket channel streaming per-device logs, renderer log console, backend process that tails `adb logcat`. |
| Alerts/Notifications | Observer service subscribing to workflow outcomes/device status, sending Slack/mail via configured channels. |
| API/CLI exposure | Hardened REST endpoints, auth (tokens), CLI scaffolding to call workflows/modules remotely. |
| Multi-session locker | Backend state for device reservations, frontend locker UI + chat thread per device, WebSocket to broadcast ownership changes. |

---

## 6. Version Tracking

Every release must update `/CHANGELOG.md` (maintain semantic versioning) and reference the docs:

- Architecture changes → update this document.
- Backend/frontend guides → note newly introduced modules/APIs/components.
- Scheduler/log streaming/multi-session features → add diagrams and data flow updates once implemented.

Use the release checklist:

1. Run `npm run build` and backend smoke tests.
2. Update docs (architecture + guides).
3. Tag release and publish artifacts (Electron bundle, backend package).

---

*Last updated: 2025-12-12*