**Admin Console: big-picture philosophy**

Your admin console should **not** be:

- a dumping ground for every internal tool
- a mirror of the database
- a "developer playground"

It **should** be:

a **mission control** for the app
focused on **operations, confidence, and intervention**

Think: *"What does an on-call human need to see or do at 02:00?"*

---

**High-level layout (mental model)**

**Left rail (primary navigation)**

Minimal, boring, predictable:

1. **Overview**
2. **Operations**
3. **Users**
4. **Flights**
5. **Messaging**
6. **Logs**
7. **Config**
8. **Admin**

No tiles jungle. No scrolling wall of buttons.

---

**1️⃣ Overview (landing page)**

This replaces the current "everything everywhere" feeling.

**What it shows (read-only, calm)**

- **System health**
  - API up/down

        - Cron last run (18:00 / 22:00)
        - Last error timestamp
    - **Today / Tomorrow**
        - Flights processed
        - Listings pending / sent / booked
    - **Users**
        - Total users
        - New signups (24h / 7d)
    - **Alerts**
        - “Booking backlog”
        - “Airport feed delayed”
        - “Push failures”

No actions here. Just **situational awareness**.

---

### 2️⃣ Operations (what legacy did well — but scattered)

This is where your existing “Daily Departures”, “Late Listings”, etc. belong.

**Subsections**

- **Daily departures**
- **Listing processing**
- **Airport bookings**
- **Cancellations / disruptions**
- **Manual overrides** (rare, guarded)

Each screen should answer:

- *What’s happening?*
- *Is it normal?*
- *Do I need to intervene?*

---

### 3️⃣ Users (this is where legacy gets dangerous)

**User search (powerful but safe)**

Search by:

- PSN (exact)

- Name (partial)
- Email
- Airport
- Airline
- Status (active / inactive)

**User profile view (single pane)**

- Identity (read-only)
- Devices registered
- Notification opt-in status
- Listing history (summary)
- Recent errors affecting this user

**Guardrails**

- **No raw password resets**
- **No silent data edits**
- All actions logged

---

### 4️⃣ Flights (your real domain)

**Flight-centric view (not user-centric)**

- Flight instance → see:
  - all listed users
  - booking status per user
  - airport responses
  - security numbers
- Timeline view:
  - listed → sent → booked → confirmed

This replaces having to mentally reconstruct things from logs.

---

### 5️⃣ Messaging (yes — absolutely)

You're spot-on here.

**Capabilities (very powerful, must be controlled)**

**Send to:**

- Single user (PSN)
- Multiple PSNs
- Dynamic groups:
  - airport
  - airline
  - flight number
  - listing status
  - date range
  - device platform

**Message types:**

- Push notification
- Silent refresh trigger
- Info banner
- Emergency alert (rare)

**Safety features (non-negotiable):**

- Preview before send
- Dry-run ("will reach 124 users")
- Rate limiting
- Full audit trail

> Your legacy console already hints at this — the new one just needs **structure and brakes**.

———

### 6️⃣ Logs (yes, but not a wall of text)

This is where your new logger fits beautifully.

**Log viewer, not raw files**

- Filter by:
  - source (server / cron / react)
  - level (INFO / WARN / ERROR)
  - event name
  - date/time
- Quick actions:
  - "Show related logs"

- “Jump to flight”
- “Jump to user”

**Important rule**

Admins **view logs**.
They do **not** tail files or see stack dumps by default.

---

### 7️⃣ Config (small, powerful, dangerous)

This is where:

- guest promo enable/disable lives
- feature flags live
- airport toggles live

Design rule:

If changing it can break the app, it must:

- be labelled clearly
- require confirmation
- be logged

---

### 8️⃣ Admin / Security

- Admin users
- Roles:
  - read-only
  - ops
  - super-admin
- Login history
- IP restrictions
- Session revocation

Your screenshots show you already care about security — formalise it.

---

**About logs (your specific question)**

> “Would all logs be viewable from there?”

**Yes — but curated.**

- Raw logs stay on disk
- Admin console provides:
  - filtered views
  - summaries
  - links to context

Think **“observability”**, not “debug console”.

---

**About push messages (your instinct is correct)**

> “Could admins send push messages to individual users and groups?”

Yes — and this is one of your **strongest operational advantages**.

But:

- make it intentional
- make it auditable
- make it boring to use incorrectly

Your legacy console already proves the value — the new one just needs **structure and restraint**.

---

**Final thought (important)**

You do **not** need to build all of this at once.

The right order is:

1. Overview
2. Logs
3. Messaging
4. Users
5. Operations
6. Config

Each one can be added **without refactoring the others** if the foundation is right.

And the foundation you're laying now (logging, clean APIs, clear domains) is exactly what makes this admin console *possible* later.

If you want next, we can:

- sketch a **navigation tree**
- define **roles & permissions**
- or design **just one screen** (e.g. Messaging) in