1) RBAC matice (role × akce)

Akce / Modul	Admin	Operátor	Manipulant
Přihlášení / logout	✓	✓	✓
Úložiště – CRUD	✓		
Trasy – CRUD (vč. hromadné aktualizace)	✓		
Pravidla urgentnosti – CRUD	✓		
Layout 12×12 – CRUD	✓		
Uživatelé – CRUD, reset hesla, (de)aktivace, homeStorage	✓		
Objednávka (Order) – založit (POST /orders)	✓	✓	
Fronta úkolů – listovat	✓		✓
Vzít úkol (new→in_progress)	✓		✓
Dokončit úkol (in_progress→done)	✓		✓
Zrušit úkol (→canceled)	✓	✓	✓*
Audit log – čtení	✓		

* Manipulant smí zrušit **jen** úkol ve stavu new (potvrzení + záznam do auditu). To odpovídá tvému popisu a snižuje chaos na hale.

Proč takto:

- Udržuje to jednoduché mentální modely: Admin = pravidla a data; Operátor = zadává; Manipulant = vykonává.
- Zrušení dáváme všem třem (s limitem pro Manipulanta), protože v reálu vznikají omyly u všech rolí a potřebujeme rychle „odblokovat“ frontu.

---

2) Realtime komunikace

Volba pro v1: SSE (Server-Sent Events) + Redis Pub/Sub.

- **Proč SSE:** Jednosměrný push (server→klient) nám stačí: Operátor i Manipulant jen *poslouchají* změny; zápisy dělají dál přes REST. SSE je firewall-friendly, levnější na údržbu než WebSocket a má skvělé auto-reconnect chování.

- **Redis Pub/Sub:** Umožní škálovat BE na více instancí; každá pushne do Redis kanálu a všechny SSE streamy dostanou stejné eventy.

Kanalizace událostí (návrh):

- GET /events?role=operator&home=A01 → posílá:
    - metrics.updated (payload: { from, count, hasUrgent, ageMinutes }) – vzniká po změně orders nebo každých 15 s jako „keep-fresh“.
    - order.status_changed (jen relevantní pro home=A01).
- GET /events?role=worker&userId=… → posílá:
    - order.created (filtr dle haly/TO mini-filtru),
    - order.status_changed (pro jeho „Moji misi“).

Změny v BE:

- Eventy se emitují **v servisní vrstvě** po úspěšné transakci (např. take → publish order.status_changed).
- Heartbeat: každých 25 s ping pro udržení spojení.
- V FE jednoduchý EventSource wrapper; když spadne, zobrazit nenásilný banner „Ztracené spojení – připojuji…“.

Proč ne Socket.IO v1: Potřebujeme hlavně server→klient push, bez presence/RTC a bez nutnosti custom acků. SSE je jednodušší a levnější. Socket.IO si necháme pro v2, pokud přidáme „živé navádění“ apod.

---

3) CI/CD & DevOps

Repo: monorepo

/apps/backend (Node+Express+Sequelize)

/apps/frontend (Vue 3 + Vite + Pinia)

/infra (docker-compose, migrations, seed, pgadmin)

/.github/workflows (CI)

Docker Compose (lokál + staging):

- postgres:16 (persistentní volume, POSTGRES_INITDB_ARGS=--data-checksums)
- pgadmin (volitelně)
- backend (port 3000, healthcheck /health, čeká na DB)

- frontend (build → Nginx static; lokálně Vite dev server)
- Make/NPX skripty: dev, migrate, seed, lint, test, build

CI (GitHub Actions – návrh jobů):

- **Checks:** Node LTS matrix, lint (ESLint/Prettier), typecheck (TS), test (Vitest/Jest + Supertest pro API).
- **Build & Artifacts:** FE build (Vite), BE docker image (tag z SHA).
- **DB migrations:** samostatný krok v deploy pipeline; idempotentní.
- **Security:** npm audit --audit-level=high fail build, SAST (např. CodeQL) jako nightly.

Proč takto:

- Minimum magie, maximum transparentnosti.
- Idempotentní migrace + seed = *kdykoliv* zreprodukovatelný stav.
- Rychlý onboarding dalších vývojářů/instancí.

---

4) Error handling & offline scénáře

Backend (jednotný kontrakt chyb):

{ "error": { "code": "BAD_REQUEST", "message": "from & to required" } }

Statusy: 400/401/403/404/409/422/500 (422 přidáváme pro validační chyby formulářů). To navazuje na tvé API zásady.

Frontend:

- Axios interceptor → mapuje kódy na srozumitelné toasty/dialogy (nepadají celé stránky).
- 401/403 → okamžitý logout (už máš).
- **Retry politika:**
    - GET (idempotentní) – exponenciální backoff (max 3 pokusy).
    - POST/PUT/DELETE – **bez retry**, pokud nemáme idempotency token (v1 nebudeme mít).
- **Empty/Loading/Error stavy** u všech klíčových komponent (mřížka, trasy, fronta).

Offline (tablet Manipulanta):

- **v1 pragmaticky:**

    - Při offline režimu: „read-only“ poslední **cache snapshot** fronty (IndexedDB) + banner „Jste offline; akce dočasně nedostupné“.
    - Zákaz take/done v offline (vyžaduje atomickou změnu na BE).
    - SSE auto-reconnect, po návratu online auto-refresh dat.
- **v2 (volitelné):** background sync + idempotency keys pro „take/done“.

Proč takto:

- Minimalizujeme riziko dvojího převzetí/doručení bez složitých kompenzací.
- Uživatel *vidí* frontu i bez netu, ale nedělá nevratné akce.

---

5) Auditní logy

Schéma (tabulka audit_logs):

- id (PK)
- ts (timestamp)
- actorId (FK → users.id, NULL pro systém)
- action (ENUM): USER_ACTIVATED, USER_DEACTIVATED, USER_RESET_PASSWORD, ORDER_CREATED, ORDER_TAKEN, ORDER_DONE, ORDER_CANCELED, ROUTES_BULK_UPDATE, PRIORITY_RULE_UPSERT, LAYOUT_SAVED, …
- entityType (ENUM): USER, ORDER, ROUTE, PRIORITY_RULE, LAYOUT
- entityId (text / bigint)
- meta (JSONB): např. {reason, from, to, prev, next, ip, ua}
- Indexy: (entityType, entityId), (ts), (action)

Kdy logovat (minimální set):

- Změny bezpečnostního charakteru: (de)aktivace uživatele, reset hesla, role.
- Kritické business akce: lifecycle objednávky (create/take/done/cancel) – *včetně toho, **kdo** zrušil a **proč***.
- Hromadné úpravy tras; uložení layoutu.

Zobrazení (v2): jednoduchá stránka v Admin → Audit (filtry + detail meta).

Proč takto:

- Plní požadavek na dohledatelnost a odpovědnost (kdo/ kdy/ co/ proč).

- JSONB meta dává volnost bez migrací při rozšiřování.

---

6) NFR (nefunkční požadavky)

Výkon & latence (cíle v1):

- POST /orders p95 < **200 ms** (DC-lokální, bez heavy logiky).
- „Time-to-visible“ nové objednávky u Manipulanta < **1 s** (SSE).
- Metriky slotů refresh **≤ 15 s** (poll + event-driven publish).
- Současně: 20 Operátorů, 30 Manipulantů, 2 Admini – bez degradace.

Dostupnost & zálohy:

- SLA cíl v1: **99.5 %** (on-prem/staging realistické).
- **Zálohy DB:** denní full + WAL/PITR; obnova testována 1× týdně.
- **Retence dat:** orders + audit min. **2 roky** (konfigurovatelně).

Bezpečnost:

- Hesla: bcrypt 10–12 (už máš); politika min. délky, zamknutí účtu po X pokusech (rate-limit login – už máš).
- JWT access **12 h** (dle dokumentu); refresh token *není nutný* v interní hale (nižší komplexita).
- CORS jen FE origin (máš); Helmet + morgan (máš); request-ID do logů (přidáme).
- Minimální osobní data: username, audit bez citlivostí (GDPR-friendly).

Observabilita:

- Strukturované logy (JSON) + **requestId** korelace.
- Základní metriky: počet objednávek / min, latence endpointů, SSE klienti.
- Healthcheck /health (máš) + /ready (DB ping).

Proč takto:

- Splníme reálné nároky haly bez over-engineerování.
- Jasná čísla = jasné akceptační kritérium.

Další ujasněné body:

1) Doménové konstanty & konfigurace

- **Prahy pro okraje slotů (UI):** amber=5 min, red=15 min (konfigurovatelné v DB tabulce app_settings). Navazuje na definici stavů slotů a metriku ageMinutes pro Operátora.
- **Interval výpočtu metrik:** backend publikuje SSE událost metrics.updated max. každých **15 s**; FE Operátora přepočítá UI každých **30 s** jako „levný“ fallback.
- **Kapacita slotu:** vždy 1 (půl-paleta).
- **Stavy objednávky:** new → in_progress → done (+ canceled jako terminální).

2) Identifikátory & validace vstupů

- **Kód úložiště (Storage.code):** regex ^[A-Z]\d{2}(-[A-Z])?$ (příklady: A01, E15, A01-A).
- **Zákaz duplicit storageCode v layoutu 12×12** – už máš jako pravidlo; FE i BE budou validovat.
- **Trasy:** unikát (from,to); BE vrací 409 při duplicitě.
- **Objednávka POST /orders:** vyžaduje from, to ∈ definované trasy; urgency ∈ {STANDARD,URGENT}.

3) RBAC (zpřesnění detailů)

- **Zrušit úkol**:
  - Admin/Operátor: new | in_progress → canceled (nutné uvést reason).
  - Manipulant: pouze new → canceled (povinné potvrzení v UI).
- **Audit činností**: viz body 8 (schema audit_logs).

4) Realtime vrstva (SSE)

- **Kanály:**
  - GET /events?role=operator&home=A01 → metrics.updated, order.status_changed.
  - GET /events?role=worker&userId=… → order.created, order.status_changed.
- **Transport:** SSE (auto-reconnect), backend s Redis Pub/Sub (škálování).
- **Důvod:** stačí jednosměrné „push“; WS si necháme pro v2 (přítomnost, navádění).

5) Chybové stavy & offline

- **Kontrakt chyb (BE):**
- { "error": { "code": "BAD_REQUEST", "message": "from & to required" } }

Statusy: 400/401/403/404/409/422/500.

- **FE retry politika:**
  - GET → exponenciální backoff (3 pokusy).
  - POST/PUT/DELETE → bez retry (kvůli idempotenci).
- **Offline (tablet Manipulanta):** read-only cache fronty (IndexedDB) + zákaz take/done offline; návrat online = auto-refresh.

6) API kontrakty – doplnění

- **Idempotence POST /orders (volitelné):** podpora hlavičky Idempotency-Key (UUID); pokud shodný klíč v posledních 2 min, vrátit původní 201. (Zabrání dvojkliku Operátora.)
- **/routes/bulk/:fromCode**: atomická transakce delete+insert (už specifikováno).
- **/orders/metrics?status=new**: přesné názvy polí: { from, count, hasUrgent, oldestCreatedAt, ageMinutes }.

7) UX & přístupnost (A11y)

- **Prázdné stavy:**
  - Operátor – „Pro tento ODKUD nejsou definovány trasy. Oprav v Admin → Trasy.“
  - Layout chybí – žlutý infobox s návodem (Admin → Layouty).
- **Error stavy:** non-blocking toasty; nikdy „white screen“.
- **Klávesnice:** Enter = odeslat objednávku v potvrzovacím overlayi; Esc = zavřít overlay.
- **Kontrast a velikost:** velikost textů min. 14px; na tabletu tlačítka min. 44×44 px.

8) Auditní logy (DB)

- **Tabulka audit_logs**:
  id, ts, actorId, action ENUM, entityType ENUM, entityId, meta JSONB
  Indexy: (entityType, entityId), (ts).

- **Logovat minimálně:** ORDER_CREATED/TAKEN/DONE/CANCELED, USER_(DE)ACTIVATED, USER_RESET_PASSWORD, ROUTES_BULK_UPDATE, PRIORITY_RULE_UPSERT, LAYOUT_SAVED.
- **UI (v2):** filtrování podle akce, entity, data.

9) Bezpečnost

- **JWT:** HS256, exp **12 h** (bez refreshu v v1).
- **Login rate-limit:** 5/min/IP.
- **Hesla:** bcrypt 10–12; min. 10 znaků, složitost bez speciálního znaku povinného (přísnost držme pragmatickou).
- **CORS:** jen FE origin.
- **Auto-logout neaktivita FE:** 12 h po vydání tokenu (navazuje na expiraci tokenu).
- **Request ID:** každý request má X-Request-ID; korelace v logách.

10) Observabilita & metriky

- **Logy:** JSON, pole requestId, actor, route, latency_ms, status.
- **Metriky:** počet objednávek/min, p95 latence endpointů, počet SSE klientů, chybovost.
- **Health/Ready:** /health (živost), /ready (DB ping).

11) NFR cíle (měřitelné)

- **Výkon:** POST /orders p95 < **200 ms**; „time-to-visible“ u Manipulanta < **1 s** (SSE).
- **Zátěž v1:** 20 Operátorů / 30 Manipulantů současně bez degradace.
- **Dostupnost:** cíl **99.5 %** (on-prem realisticky).
- **Retence:** orders + audit_logs min. **24 měsíců**.

12) CI/CD & prostředí

- **Monorepo**: /apps/backend, /apps/frontend, /infra, /.github/workflows.
- **Compose (lokál/staging):** Postgres 16, pgAdmin (volit.), BE, FE (Nginx).
- **Skripty:** dev, migrate, seed, lint, test, build.
- **CI kroky:** lint → typecheck → test (Jest/Vitest + Supertest) → build artefaktů → image push → migrace při nasazení.
- **.env.example:** DATABASE_URL, JWT_SECRET, CORS_ORIGIN, VITE_API_URL.

13) Seed & bootstrap

- **Admin účet:** admin / admin (vynucená změna při 1. přihlášení).
- **Ukázková data:** storages: A01, G22, …, routes: A01→G22, priority_rules (STANDARD), 3–5 orders:new pro demo Operátora.
- **Layout:** 12×12 se 4–6 aktivními buňkami (A-zóna).

14) Podporované platformy

- **Prohlížeče:** Chrome ≥ 115, Edge ≥ 115; interní kiosky – Chrome.
- **Tablet manipulanta:** Android ≥ 10, 10", minimálně 2 GB RAM; viewport 1280×800; PWA „Add to Home Screen“.

15) Lokalizace & formáty

- **Jazyk v1:** CZ (i18n připraveno).
- **Časová zóna:** Europe/Prague; formát HH:mm, datum DD.MM.YYYY.
- **Relativní časy v UI:** „{ageMinutes} min“ (zaokrouhleno dolů).

16) Skenery / QR (rezerva do v1.1)

- **Formát QR:** ORDER:<id> nebo STORAGE:<code>; BE endpointy připravené, UI zatím bez povinnosti.

17) Edge-cases & ochrany

- **Flood ochrana Operátora:** limit 5 objednávek / 10 s na uživatele; při překročení toast „Zkuste odeslat za chvíli“.
- **Konzistence take:** atomický UPDATE WHERE id=? AND status='new' (už specifikováno).
- **Deaktivace uživatele:** transakčně vrací rozpracované úkoly do new (už specifikováno).