Architektura (high-level)

- **Frontend:** Vue 3 (Composition API) + Vite + Vue Router + Pinia (stav) + Axios (HTTP).
  Cíl: rychlá SPA, čisté moduly pro Operátora, Admina, Manipulanta

- **Backend:** Node.js + Express + Sequelize (ORM) + JWT (auth) + zod/express-validator (validace) + bcrypt (hesla).
  Cíl: jednoduchý REST, čitelné middleware, transakce při zápisech.

- **DB:** PostgreSQL.
  Cíl: pevná integrita (unique/foreign klíče), indexy na kritických polích, MIGRACE.

Frontend (Vue 3 + Vite + Axios)

Struktura projektu

```
src/
  main.ts
  app.vue
  router/
    index.ts                // Vue Router (guards: auth, role)
  store/
    auth.ts                 // Pinia: token, user, role,
homeStorageCode
    ui.ts                   // drobné UI stavy (toasty apod.)
  api/
    http.ts                 // Axios instance + interceptory
    storages.ts
    routes.ts
    priority.ts
    users.ts
    orders.ts
  pages/
    Auth/Login.vue
    Operator/View.vue       // A/B flow + spodní vizuál
    Admin/
```

```
      Storages.vue          // CRUD
      Routes.vue            // z pohledu FROM + hromadné akce
      PriorityRules.vue     // CRUD
      LayoutEditor.vue      // 12×12, validace, uložit do BE
      Users.vue             // CRUD + homeStorage
    Worker/                 // Manipulant – fronta (v1)
  components/
    layout/AppShell.vue
    ui/
      Button.vue, Chip.vue, Toast.vue, Table.vue, Badge.vue
  assets/
    logo.svg                // „MANIPULÁTOR“ + půl-paleta + modré
šipky
```

Router & Guardy

- **/login** (public)
- **/operator** (role: `operator|admin`)
- **/admin/*** (role: `admin`)
- Guard: pokud není `auth.token` → redirect `/login`. Při 401/403 z API → logout.

Pinia store – `auth`

- `token`, `user:{username,role,homeStorageCode?}`, `login()`, `logout()`.
- Po loginu uložit token do `localStorage`, nastavit Axios Authorization.

Axios (`api/http.ts`)

- BaseURL z `.env` (`VITE_API_URL`).
- Request interceptor: `Authorization: Bearer <token>` (pokud existuje).
- Response interceptor: 401/403 → `auth.logout()`.

Operátor – UX & data

- Čte **layout** z `BE` (12×12) a validuje kódy proti `/storages`.
- **Stav A**: výběr ODKUD (zelený slot) → fetch `/routes?from=…`
- **Stav B**: výběr KAM (oranžová tlačítka) → default urgence z `/priority-rules`.
- **Odeslat**: `POST /orders`.
- **Stavy slotů z BE**: `GET /orders/metrics?status=new` každých 30 s:
    - pro `from`: `count`, `hasUrgent`, `ageMinutes` → šedý závoj, ⏱, ⚡, barevný okraj (prahy 5/15 min).
- Spodní vizuál v1: žlutá čára / pruh VZV / bílá linka / chodník (jen vrstvy).

Admin – moduly

- **Úložiště**: `/storages` CRUD.
- **Trasy**: `/routes` – editor z pohledu FROM (✓ existuje / ☐ k založení, hromadné akce + Undo toast).
- **Urgence**: `/priority-rules` CRUD (`scope='route'`, `from`, `to`, `defaultUrgency`, `enabled`).
- **Layout 12×12**: čte z BE
    - grid[i][j] = `{active, storageCode?, label?}` (label fallback = 1. písmeno kódu),
    - validace: storage existuje, **žádné duplicity storageCode**.
- **Uživatelé**: `/users` CRUD, aktivace, reset hesla, `homeStorageCode`.

Backend (Express + Sequelize)

Middleware

- `helmet`, `cors`, `morgan` (log), `express.json()`.
- `authJwt` – ověř token (Bearer), připojí `req.user`.
- `requireRole(...roles)` – RBAC.
- `errorHandler` – jednotný JSON `{ error: { code, message } }`.

Modely (Sequelize)

User

- `id PK`, `username (uniq)`, `passwordHash`, `role ENUM('admin','operator','worker')`, `active bool default true`, `homeStorageCode FK→Storage.code NULL`.
- Index: `username unique`.

Storage

- `id PK`, `code (uniq)`, `name`.
- Index: `code unique`.

Route

- `id PK`, `fromCode FK→Storage.code`, `toCode FK→Storage.code`, `active bool default true`.
- Unique: `(fromCode,toCode)`.
- Indexy: `fromCode`, `toCode`.

PriorityRule

- `id PK`, `scope ENUM('route')`, `fromCode FK`, `toCode FK`, `defaultUrgency ENUM('STANDARD','URGENT')`, `enabled bool`.
- Unique: `(scope,fromCode,toCode)`.

Order

- `id PK`, `fromCode FK`, `toCode FK`, `urgency ENUM('STANDARD','URGENT')`,
- `note TEXT NULL`,
- `status ENUM('new','in_progress','done','canceled') default 'new'`,
- `assigneeId FK→User.id NULL`,
- `createdAt`, `updatedAt`, `takenAt NULL`, `doneAt NULL`, `canceledAt NULL`.
- Indexy: `status`, `fromCode`, `(fromCode,status)` (pro metriky), `(urgency,status)`.

Pozn.: Layout v1 držíme BE: `Layout(id PK, name)`, `LayoutCell(layoutId, row, col, active, storageCode, label)`.

Asociace (logické)

- `Storage.hasMany(Route, { as:'asFrom', foreignKey:'fromCode' })`
- `Storage.hasMany(Route, { as:'asTo', foreignKey:'toCode' })`
- `User.hasMany(Order, { foreignKey:'assigneeId' })`

Bezpečnost

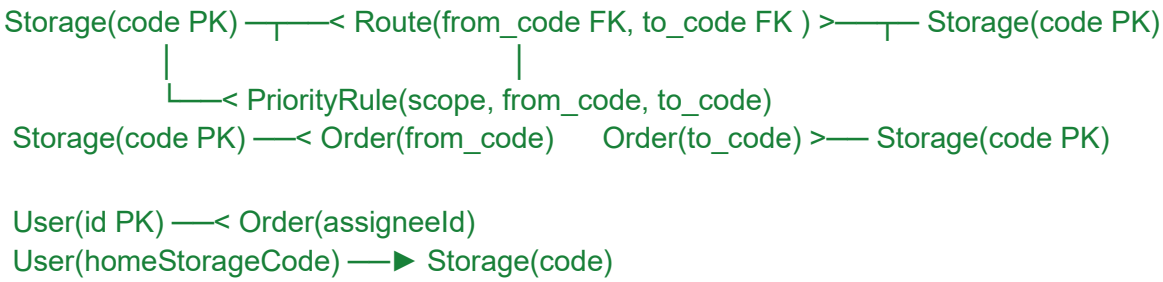
- Hesla: `bcrypt` (salt 10–12).
- JWT: HS256, expirace 12 h.
- Rate-limit na `/login` (např. 5/min/IP).
- Sanitizace a validace vstupů (`zod`/`express-validator`).
- CORS pouze z FE originu.

Databáze (PostgreSQL)

DDL (zkráceně)

- `storages(code UNIQUE NOT NULL)`
- `routes(from_code, to_code, UNIQUE(from_code,to_code))`
- `priority_rules(scope, from_code, to_code, UNIQUE(scope,from_code,to_code))`
- `users(username UNIQUE, role, active, home_storage_code NULL REFERENCES storages(code))`
- `orders(from_code REFERENCES storages(code), to_code REFERENCES storages(code), status, urgency, assignee_id NULL REFERENCES users(id), created_at, taken_at, done_at, canceled_at)`

ER diagram (ASCII)



Indexy a výkonnost

- `routes(from_code)` – rychlé `GET /routes?from=…`
- `orders(from_code,status)` – metriky pro sloty
- `orders(status)` – fronty manipulanta
- `priority_rules(from_code,to_code,enabled)` – rychlý default urgence

API_specifikace (REST)

Auth

- `POST /login`
  Req: `{ username, password }`
  Res: `{ token, user:{id,username,role,homeStorageCode} }`
  Err: 401 wrong creds

Úložiště

- `GET /storages` → `[{id,code,name}]`
- `POST /storages` → create (409 dup code)
- `PUT /storages/:id`
- `DELETE /storages/:id` (409 pokud referencováno trasami/orders)

Trasy

- `GET /routes?from=A01`
  → `[{ id, from:'A01', to:'G22', name:'Lakovna', status:'exists' }]`
- `POST /routes { from, to }` (idempotentní – 409 dup)
- `DELETE /routes/:id`
- `PUT /routes/bulk/:fromCode (pro hromadnou aktualizaci/revert tras) → jedna databázová transakce, která nejprve **odstraní všechny trasy** pro dané fromCode a poté **vloží nové** z přijatého seznamu toCodes.`

Pravidla urgentnosti

- `GET /priority-rules`

- POST /priority-rules { scope:'route', from, to, defaultUrgency, enabled }
- PATCH /priority-rules/:id
- DELETE /priority-rules/:id

Uživatelé

- GET /users
- POST /users { username,password,role,homeStorageCode? }
- PATCH /users/:id (např. { active:false } _(při nastavení active:false backend **zároveň v transakci uvolní všechny úkoly**, které měl uživatel ve stavu 'in_progress', převede je na 'new' a vynuluje assigneeId)_, { homeStorageCode:'A01' })
- POST /users/:id/reset-password { password }
- DELETE /users/:id

Objednávky (v1 – včetně Manipulanta)

- POST /orders { from,to,urgency,note? }
  → 201 { id, from, to, urgency, note, status:'new', createdAt }
- **Metriky slotů pro Operátora:**
  GET /orders/metrics?status=new
  → [{ from, count, hasUrgent, oldestCreatedAt, ageMinutes }]
- Fronta manipulanta (v1):
  - **Podseznam: Fronta manipulanta (v1):**
    - **GET /orders?status=new** → seznam čekajících úkolů (filtrování dle haly/mini-filtru „TO“ na FE).
    - **POST /orders/:id/take** → 200 (atomicky new → in_progress; optimistický zámek).
    - **POST /orders/:id/done** → 200 (in_progress → done).
    - **POST /orders/:id/cancel** → 200 (new → canceled; **RBAC:** Admin/Operátor mohou rušit new|in_progress s povinným

důvodem, **Manipulant** jen new).

Formát chyb

```
Status: 400/401/403/404/409/500
{
  "error": { "code": "BAD_REQUEST", "message": "from & to required"
}
}
```

Datové_modely (TypeScript/JSON kontrakty)

```
// Storage
type Storage = { id: number; code: string; name: string };

// Route row
type Route = { id: number; from: string; to: string; name: string;
status: "exists" | "not" };

// Priority rule
type PriorityRule = {
  id: number;
  scope: "route";
  from: string;
  to: string;
  defaultUrgency: "STANDARD" | "URGENT";
  enabled: boolean;
};

// User
type User = {
  id: number;
  username: string;
  role: "admin" | "operator" | "worker";
  active: boolean;
  homeStorageCode?: string | null;
};
```

```
// Order
type Order = {
  id: number;
  from: string;
  to: string;
  urgency: "STANDARD" | "URGENT";
  note?: string | null;
  status: "new" | "in_progress" | "done" | "canceled";
  assigneeId?: number | null;
  createdAt: string;
  takenAt?: string | null;
  doneAt?: string | null;
};
```

Implementační poznámky (aby to „lítalo“)

- **Sequelize migrations**: inicializační migrace + seed (admin/admin, ukázkové úložiště A01/G22).
- **Transakce**:
    - **`POST /orders/:id/take (v1)`** → `UPDATE ... WHERE id=? AND status='new'` vrátí `rowCount=1` jen prvnímu (optimistické zamčení).
    - `PUT /routes/bulk/:fromCode (pro hromadnou aktualizaci/revert tras) → jedna databázová transakce, která nejprve **odstraní všechny trasy** pro dané fromCode a poté **vloží nové** z přijatého seznamu toCodes.`

Metriky pro sloty (efektivně):

```
SELECT from_code   AS from,
       COUNT(*)    AS count,
       BOOL_OR(urgency='URGENT') AS hasUrgent,
       MIN(created_at) AS oldestCreatedAt,
```

```
       EXTRACT(EPOCH FROM (NOW() - MIN(created_at)))/60.0 AS
ageMinutes
FROM orders
WHERE status='new'
GROUP BY from_code;
```

-
- **Validace**:
    - Route create: ověř existence `from` i `to`, a unikátnost páru.
    - Layout editor (FE): nepustit uložit duplicity/unknown code.
- **Bezpečnost**:
    - Hash hesel, rate-limit, audit log přihlášení (volitelně).
- **DX**:
    - `.env`: `PORT`, `DATABASE_URL`, `JWT_SECRET`, `CORS_ORIGIN`.
    - Skripty: `dev`, `migrate`, `seed`, `start`.
- **Nasazení**:
    - FE build `vite build` → Nginx/Static.
    - BE na PM2/Docker. Healthcheck `/health`.
    - DB: managed Postgres; zálohy + role.

Sekce: „Datové_modely (TypeScript/JSON kontrakty) → // Order“

Kde: definice typu `Order`.

```
type Order = {
  id: number;
  from: string;
  to: string;
  urgency: "STANDARD" | "URGENT";
  note?: string | null;
  status: "new" | "in_progress" | "done" | "canceled";
  assigneeId?: number | null;
```

```
  createdAt: string;
  takenAt?: string | null;
  doneAt?: string | null;
  canceledAt?: string | null;
};
```