Backend Aduanas - resumen detallado

Arranque y config:
- npm start -> src/server.js; crea carpeta backend/uploads si falta; escucha PORT \(env o 4000\).
- app.js: Express 5, CORS, JSON/urlencoded 100MB, timeout 12h en req/res; ruta base /api.
- env.js: carga .env, exige DB_HOST/USER/PASSWORD/NAME, JWT_SECRET fuerte; DB_PORT por defecto 3377.
- db.js: pool mysql2/promise con waitForConnections, limit 10.

Seguridad y JWT:
- auth.middleware requireAuth: lee Authorization: Bearer <token>, verifica con JWT_SECRET, setea req.user o 401.
- auth.routes: POST /api/auth/login con rate limit 10 intentos/15min.
- auth.controller: busca usuario \(reg_usuarios\), valida pass; si estaba en texto plano y coincide, migra a bcrypt.
  Firma JWT \(8h\) con payload {id, nombre_completo}; retorna usuario sin password + token.

Rutas montadas en /api \(routes/index.js\):
- Publica: /auth
- Protegidas \(requireAuth\): /cajas, /registros, /inventario, /download.

Middlewares:
- auth.js: verifica JWT y exporta JWT_SECRET.
- upload.js: multer diskStorage a backend/uploads, nombres fieldname-timestamp.ext, solo Excel, max 100MB.

Modelos \(src/models\):
- caja.model: tabla reg_caja; filtros permitidos \(id, anaquel, cuerpo, nivel, fila, posicion, f_creacion, z_ubicacion, locacion, created_at/by, updated_at/
- registro.model: tabla ad_inventario_reg; filtros con LIKE en campos textuales; paginacion; findDuplicate combinando n_caja, n_paquete, n_registro,
- inventario.model: tabla inventario; filtros con LIKE en textos \(nombrearchivo, nromemo, nmesa, obs_estado, estado, uni_org_id\); paginacion; findD
- usuario.model: findByNombre TRIM; updatePasswordHash.

Controladores \(src/controllers\):
- caja.controller: CRUD usa req.user para created_by/updated_by; PDF con pdfmake \(fuentes backend/fonts\); importExcel lee XLSX, valida campos
- registro.controller: CRUD, normaliza codigos, PDF; importExcel valida numericos, campos obligatorios, detecta duplicados via findDuplicate; genera
- inventario.controller: CRUD, bloquea duplicados con findDuplicate; PDF.
- auth.controller: login/JWT \(detalle arriba\).
- download.controller: descarga segura de archivos _errores_.xlsx/.xls desde uploads; sanitiza nombre \(path.basename\) y comprueba extensi?n y n

Rutas funcionales clave:
- /api/cajas: GET lista con filtros/paginacion; GET :id; POST; PUT :id; DELETE :id; GET /report/pdf; POST /import/excel \(multer\).
- /api/registros: GET lista; GET :id; POST; PUT :id; DELETE :id; GET /report/pdf; POST /import/excel; GET /download/errors/:filename.
- /api/inventario: GET lista; GET :id; POST; PUT :id; DELETE :id; GET /report/pdf.
- /api/download/:filename: descarga archivo de errores \(con requireAuth\).

Uploads y descargas:
- backend/uploads se crea al arrancar; aqu? guardan Excel importados y errores.
- Descargas verifican nombre con path.basename y solo permiten _errores_.xlsx/.xls; borran el archivo al finalizar.

PDFs y reportes:
- pdfmake con fuentes locales \(Roboto en backend/fonts\); orientation landscape; cabeceras y tablas por m?dulo.

Importaci?n Excel:
- Usa XLSX para leer; valida fila a fila; inserta v?lidos, registra errores \(mensaje y fila original\).
- Si hay errores, genera Excel de errores con columnas y detalle; devuelve nombre del archivo para descarga.

Notas de entorno:
- Ajusta DB_PORT si tu MySQL usa 3306 \(env.js pone 3377 por defecto\).
- Script docs:build en package.json apunta a ../docs/generate-docs.js \(no existe\).