Architecture

Plock technical architecture, Kotlin/Ktor backend, React MFE frontend, PostgreSQL, Redis

Tech Stack (CEO approved 2026-03-03)

Tech Stack (CEO approved 2026-03-03)

Decision Date

CEO Approval: 2026-03-03 **Supersedes:** Express + TypeScript backend, monolithic Next.js frontend

---

Backend

Component	Choice	Rationale
Language	Kotlin	Type-safe, JVM performance, JetBrains ecosystem
Framework	Ktor	Coroutine-based async, lightweight, idiomatic Kotlin
Database	PostgreSQL 16	Rock-solid relational DB, JSONB for flexible fields
ORM	Kotlin Exposed	Type-safe SQL DSL, no annotation magic
Build tool	Gradle (Kotlin DSL)	Standard for Kotlin/JVM projects
Testing	Kotest + Testcontainers	BDD-style tests, real DB for integration tests
Auth	JWT (24h expiry) + refresh tokens	Standard stateless auth
Queue	Kotlin coroutines + Redis	Async jobs: label generation, carrier calls

---

Frontend — Micro-Frontend Architecture

Component	Choice	Rationale
Framework	React 19 + TypeScript	Stable, large ecosystem
Bundler	Vite + Module Federation	MFE-ready, fast HMR
Architecture	Micro-Frontend (MFE)	Each feature independently deployable
Shared UI	Tailwind CSS 4 + shadcn/ui	Design tokens in packages/ui/
State	Zustand per MFE	Lightweight, no global store coupling

MFE Structure

MFE	Responsibility	Deploy
shell	Layout, routing, auth, navigation	Always
mfe-inventory	Inventory management, bin locations, cycle counts	Independent
mfe-orders	Order management, inbound/outbound	Independent
mfe-picking	Picking workflows, wave management	Independent
mfe-settings	User management, integrations, config	Independent
mfe-ai	Plock AI Chat, analytics, dashboards	Independent

Each MFE is **independently deployable** — a change to mfe-picking does not require redeploying the shell or other MFEs.

AI / ML Layer

Component	Choice	Use Case
LLM	Claude API (claude-sonnet-4-5)	Plock AI Chat — Swedish NL queries
ML framework	Python microservice	Isolation from JVM, flexible ML tooling
Route optimisation	OR-Tools (Google)	Smart Picking — VRP/TSP solver
Anomaly detection	scikit-learn	Z-score + isolation forest

Component	Choice	Use Case
Forecasting (Phase 2)	XGBoost / Prophet	Demand forecasting

Infrastructure

Component	Choice
Monorepo	Turborepo (frontend MFEs) + Gradle (Kotlin backend)
Mobile (Phase 2)	React Native — scanner + picking app
Cache	Redis
Testing (frontend)	Vitest + Playwright
CI/CD	GitHub Actions
Containers	Docker + Docker Compose (local dev)

Architecture Decision Records

ADR-001: Kotlin over TypeScript for backend (2026-03-03)

Context: Previous design used Express + TypeScript. Decision to move to Kotlin + Ktor. Rationale: JVM performance for WMS workloads, stronger type safety with Exposed ORM, coroutine-based concurrency fits async warehouse operations (carrier calls, label generation). Status: Accepted by CEO 2026-03-03.

ADR-002: Micro-Frontend over monolithic Next.js (2026-03-03)

Context: WMS has distinct feature domains that evolve at different rates (picking vs AI vs integrations). Rationale: Independent deployment reduces risk, enables feature teams to work in parallel, allows phased feature releases. Trade-off: Higher initial complexity, but pays off at scale. Status: Accepted by CEO 2026-03-03.

Project Structure

Project Structure

Monorepo Layout

The Plock monorepo uses Turborepo for frontend orchestration and Gradle for the Kotlin backend.

Root folders:

- **backend/** — Kotlin/Ktor backend (Gradle). Namespace: no.alai.plock
  - Application.kt — Ktor entry point
  - plugins/ — Routing, serialization, auth, CORS
  - routes/ — API routes by domain (auth, inventory, orders, picking, carriers, ai)
  - models/ — Data classes + Exposed table definitions
  - services/ — Business logic
  - integrations/ — Fortnox, PostNord, DHL, Instabee
- **frontend/** — Micro-Frontend apps (Vite + Module Federation)
  - shell/ — Shell MFE: layout, nav, auth, routing (always deployed)
  - mfe-inventory/ — Inventory management (independently deployable)
  - mfe-orders/ — Order management (independently deployable)
  - mfe-picking/ — Picking + wave management (independently deployable)
  - mfe-settings/ — Settings + users (independently deployable)
  - mfe-ai/ — AI Chat + analytics (independently deployable)
- **packages/** — Shared libraries
  - ui/ — @plock/ui: shadcn/ui base + Plock custom components, Tailwind CSS 4 design tokens
  - types/ — @plock/types: shared TypeScript types (API contracts)
  - utils/ — @plock/utils: shared utilities
- **docs/** — All documentation (mirrors BookStack)
- **brand/** — Brand assets (colours, logo, typography)
- **design/** — Figma exports, wireframes (via frontend-design skill — ZAKON #3)
- **infrastructure/** — Docker Compose, CI/CD, K8s manifests
- **legal/** — Swedish regulatory compliance (GDPR, moms, Intrastat)
- **marketing/** — GTM materials

Key Conventions

Convention	Detail
Package names	@plock/ui, @plock/types, @plock/utils
Backend namespace	no.alai.plock
Environment secrets	.env.local — never committed to git
API base URL	api.plock.se (production), localhost:8080 (local)
Frontend URL	plock.se (shell), MFEs on sub-paths or CDN
Test coverage target	80%+ for services layer
DB migrations	Exposed schema + migration scripts in backend/src/main/resources/migrations/

Root Config Files

File	Purpose
docker-compose.yml	Local dev services: PostgreSQL, Redis, backend, shell
turbo.json	Turborepo pipeline: build, dev, test, lint
build.gradle.kts	Kotlin backend build config
CLAUDE.md	Project handbook (AI director instructions)
PIPELINE.md	8-gate pipeline tracker