CASE STUDY

BlockEstate

Enterprise PropTech Real Estate Listing & Search Platform

Prepared by: **CodeVix Labs**

Industry: **PropTech / Real Estate Technology**

Document Type: **Technical Case Study**

Classification: **Confidential**

Date: **February 2026**

Version: **1.0**

Table of Contents

01 Executive Summary
02 Project Overview
03 The Challenge
04 Solution Architecture
05 Technology Stack
06 Core Features & Capabilities
07 Data Pipeline & Processing
08 Search & Discovery Engine
09 Compliance & Governance
10 Admin Portal & Operations
11 Partner Ecosystem & Billing
12 Security & Access Control
13 Infrastructure & DevOps
14 Key Metrics & Results
15 Conclusion

01 Executive Summary

CodeVix Labs was engaged to architect, develop, and deliver **BlockEstate** — an enterprise-grade PropTech platform designed to aggregate, normalize, and serve real estate listings from multiple MLS (Multiple Listing Service) data sources. The platform provides a modern, consumer-facing search experience backed by powerful admin tooling, compliance automation, and a scalable partner ecosystem.

BlockEstate transforms raw MLS data into actionable, searchable real estate intelligence through a sophisticated multi-stage pipeline. The system ingests data via RESO Web API standards, normalizes it against a canonical schema, indexes it in Elasticsearch for sub-second search, and presents it through a responsive Next.js web application. Every listing passes through an 11-state moderation lifecycle with automated compliance scoring, ensuring adherence to MLS/IDX display regulations.

Key Achievement: A production-ready platform capable of processing thousands of listings per hour, with real-time search across normalized multi-source data, automated compliance enforcement, and a complete partner API ecosystem — all built on a modern TypeScript monorepo architecture with full infrastructure-as-code.

15+	25+	22+	11
Tech Stack	API Endpoints	DB Migrations	Moderation States
Libraries	REST v1	PostgreSQL	FSM Lifecycle

02 Project Overview

BlockEstate is a full-stack real estate listing platform that bridges the gap between fragmented MLS data sources and modern consumer expectations. The platform addresses the core challenge in the PropTech industry: aggregating, standardizing, and serving real estate data from disparate sources while maintaining strict regulatory compliance.

Project Scope

- **Data Ingestion:** Multi-MLS data aggregation via RESO Web API with OAuth 2.0 authentication, delta and full sync capabilities
- **Data Processing:** Normalization pipeline mapping 100+ vendor-specific fields to a canonical schema with deduplication and enrichment
- **Search Platform:** Elasticsearch-powered full-text and geospatial search with faceted navigation and relevance tuning
- **Consumer Experience:** Server-rendered Next.js application with responsive design, interactive maps, and lead capture
- **Admin Operations:** Comprehensive admin portal for moderation, compliance, audit, and system monitoring
- **Partner Ecosystem:** API key management, webhook integrations, tiered billing, and usage metering
- **Infrastructure:** Docker/Kubernetes deployment with Terraform-managed AWS infrastructure and CI/CD automation

Client Objectives

The client required a platform that could scale horizontally to accommodate growing MLS partnerships, maintain sub-second search response times, enforce complex MLS/IDX compliance rules automatically, and provide a foundation for a partner ecosystem with self-service API access and billing. The system needed to be production-ready with enterprise-grade observability, security, and operational tooling.

03 The Challenge

The real estate technology space presents a unique set of engineering challenges that traditional web applications don't encounter. CodeVix Labs identified and addressed the following critical challenges:

Data Fragmentation

MLS providers use varying data schemas, field naming conventions, and value enumerations. A single property attribute like "status" can have dozens of different representations across sources. The system needed a robust normalization layer to unify these disparate formats into a single canonical schema.

Regulatory Compliance

MLS/IDX display rules are complex and vary by region. Each MLS imposes specific requirements for how listings must be displayed, including mandatory disclaimers, agent attribution, freshness requirements (data must be refreshed within specific intervals), and opt-out enforcement. Non-compliance can result in data access revocation.

Data Freshness & Scale

Real estate data changes frequently — new listings, price updates, status changes, and delistings occur continuously. The platform needed to process thousands of records per sync cycle while maintaining sub-second search response times and data freshness within 15-minute windows.

Search Relevance

Real estate search requires multi-dimensional filtering (price, location, beds, baths, property type) combined with full-text search, geospatial radius queries, and intelligent relevance ranking. The search experience had to feel instantaneous while handling complex query compositions.

Audit & Traceability

Every data change, moderation action, and access event needed to be immutably recorded for compliance audits. The audit system had to be tamper-proof with cryptographic hash-chain verification.

04 Solution Architecture

CodeVix Labs designed BlockEstate as a modular monorepo architecture, decomposing the system into well-defined packages with clear boundaries. This approach enables independent development, testing, and deployment of each subsystem while maintaining type safety across the entire codebase.

High-Level Data Flow

Step	Component	Function	Package
1	MLS Provider	RESO Web API with OAuth 2.0	External
2	Ingest Worker	Fetch, paginate, store raw data	reso-client
3	Raw Storage	Encrypted JSON in S3/MinIO	media-pipeline
4	Normalizer	Map to canonical schema, enrich	normalizer
5	PostgreSQL + PostGIS	Canonical listings storage	db
6	Search Indexer	Transform & index to ES	search-indexer
7	Elasticsearch	Full-text + geo search index	search-indexer
8	API Gateway	REST endpoints with auth	api
9	Next.js Frontend	SSR consumer & admin UI	web

Monorepo Structure

The project is organized as a Turborepo-managed monorepo with clear separation between applications and shared packages:

Path	Description
apps/web	Next.js 14 frontend with App Router — consumer search, listing detail, admin portal
apps/api	Express.js REST API — v1 endpoints with middleware stack
packages/shared	30+ shared utilities: types, compliance, moderation, billing, feature flags
packages/reso-client	RESO Web API OAuth client with delta/full sync logic
packages/normalizer	MLS-to-canonical schema transformation with enrichment

packages/search-indexer	Elasticsearch indexing pipeline and query builder
packages/media-pipeline	S3 upload, image processing (Sharp), presigned URLs
packages/db	PostgreSQL migrations, repository pattern, query builders
packages/pipeline	BullMQ orchestrator wiring all workers together
infra/terraform	AWS infrastructure-as-code (RDS, OpenSearch, S3, ECS)
infra/k8s	Kubernetes manifests and Helm charts

05 Technology Stack

The technology stack was carefully selected to balance developer productivity, runtime performance, operational reliability, and ecosystem maturity. Every choice was made with production-readiness and long-term maintainability in mind.

Layer	Technology
Runtime & Language	Node.js 20, TypeScript 5.6
Frontend Framework	Next.js 14 (App Router), React 18, Tailwind CSS 3.4
Backend Framework	Express.js with structured middleware
Database	PostgreSQL 16 + PostGIS 3.4 (geospatial)
Search Engine	Elasticsearch 8.15
Task Queue	BullMQ 5.12 + Redis 7
Object Storage	AWS S3 / MinIO (local development)
Image Processing	Sharp (resize, thumbnails, WebP conversion)
Mapping	React Leaflet 4.2 (interactive maps)
Logging	Pino 9.5 (structured JSON logging)
HTTP Client	Axios (with retry, interceptors)
Build System	Turborepo (monorepo orchestration)
Containerization	Docker (multi-stage builds)
Orchestration	Kubernetes (Helm charts)
Infrastructure	Terraform (AWS: RDS, OpenSearch, S3, ECS, ALB)
CI/CD	GitHub Actions (lint, test, build, deploy)
Payments	Stripe (subscriptions, invoices)

Architecture Principle: The entire stack is TypeScript end-to-end, from database queries to API handlers to frontend components. This provides compile-time type safety across service boundaries, reducing runtime errors and enabling confident refactoring across the monorepo.

06 Core Features & Capabilities

6.1 Consumer Search Experience

The consumer-facing search interface is built with Next.js 14 using server-side rendering for SEO optimization and fast initial page loads. Key capabilities include:

- **Full-Text Search** with fuzzy matching, stemming, and synonym expansion powered by Elasticsearch
- **Advanced Filtering** by price range, bedrooms, bathrooms, property type, and listing status
- **Geospatial Radius Search** with interactive React Leaflet maps and distance-based results
- **Faceted Navigation** with real-time aggregation counts for property type, status, and price buckets
- **Configurable Pagination** with sort options (price, date, relevance)
- **Listing Detail Pages** with photo galleries, agent contact, compliance disclaimers, and property details
- **Lead Capture Forms** with email/phone validation and source tracking
- **Saved Searches & Price Alerts** with email notification subscriptions

6.2 Multi-MLS Data Integration

BlockEstate supports ingesting data from multiple MLS providers simultaneously through the RESO Web API standard. The integration layer handles:

- **OAuth 2.0 Client Credentials** flow with automatic token refresh and caching
- **Delta Sync** (15-minute intervals) for efficient incremental updates
- **Full Reconciliation** (weekly) to catch any missed updates and maintain data integrity
- **Paginated Data Retrieval** with configurable batch sizes (200 records default) and rate limiting (5 req/sec)
- **Checkpoint Tracking** for resumable sync operations after failures
- **Schema Drift Detection** to automatically alert when MLS providers change field definitions

6.3 11-State Moderation Lifecycle

Every listing in BlockEstate passes through a formal finite state machine (FSM) with 11 distinct states and controlled transitions. This ensures regulatory compliance and data quality at every stage of the listing lifecycle:

State	Description
DRAFT	Initial state for newly ingested listings

PENDING_REVIEW	Queued for manual or automated compliance review
ACTIVE	Approved and visible to consumers
COMPLIANCE_WARNING	Minor compliance issues detected, still visible
COMPLIANCE_FAILED	Critical compliance failures, hidden from consumers
SUSPENDED	Temporarily removed by admin action
STALE	Data freshness exceeded threshold (>24h without update)
ARCHIVED	Listing closed/sold, moved to archive index
REMOVED	Removed by MLS provider or agent opt-out
KILLED	Emergency delisting via two-factor kill switch
TOMBSTONE	Permanently archived with full audit trail

6.4 Quality Assurance & Fraud Detection

The platform includes automated quality checks and anomaly detection to maintain data integrity:

- **Completeness Scoring:** Flags listings missing critical fields (images, description, price)
- **Duplicate Detection:** Fuzzy address matching to identify and merge duplicate entries from different MLS sources
- **Price Anomaly Detection:** Statistical analysis to flag unrealistic pricing (e.g., $1 listings, extreme outliers)
- **Bulk Change Detection:** Alerts when suspicious mass updates occur (potential data corruption or unauthorized access)
- **HTML Sanitization:** Removes script tags and malicious content from listing descriptions

6.5 Advanced Platform Features

- **Feature Flag Management:** Runtime toggles for canary deployments, A/B testing, and gradual rollouts with percentage-based targeting
- **White-Label Capability:** Customizable branding per partner/tenant for reseller partnerships
- **Data Lineage Tracking:** Full schema registry with field-level provenance tracking from source MLS to consumer display
- **Paid Listing Boosts:** Auction system for premium listing placement in search results
- **SLO Monitoring:** System-level Service Level Objective tracking with error budget calculations
- **Analytics Pipeline:** Event buffering for downstream data warehouse ingestion

07 Data Pipeline & Processing

The heart of BlockEstate is its multi-stage data processing pipeline, orchestrated by BullMQ with Redis-backed queues. Each stage is independently scalable and fault-tolerant.

Pipeline Stages

Stage 1: Ingest Worker (reso-client)

The ingest worker connects to MLS providers via the RESO Web API standard. It acquires OAuth 2.0 tokens, paginates through OData endpoints with configurable batch sizes (default 200), and stores raw JSON responses in S3 with encryption at rest. Rate limiting (5 requests/second) prevents provider throttling. Checkpoint tracking enables resumable syncs after failures.

Stage 2: Normalizer Worker (normalizer)

The normalizer transforms raw MLS data into BlockEstate's canonical schema. It maps 100+ RESO-standard and vendor-specific fields, handles type coercion (numeric parsing, date normalization, boolean detection), performs geocoding integration, calculates enriched fields (price per square foot, days on market), and executes HTML sanitization on description fields. Deduplication uses fuzzy address matching to prevent duplicate entries.

Stage 3: Search Indexer Worker (search-indexer)

Normalized listings are transformed into Elasticsearch documents with optimized mappings. The indexer configures full-text analysis (tokenization, stemming, synonym expansion), spatial indexing for geo_point queries, and facet aggregations. Index alias management enables zero-downtime reindexing. Active and archived listings use separate indices.

Stage 4: Media Pipeline Worker (media-pipeline)

The media worker downloads listing images from RESO URLs, generates multiple thumbnail sizes (thumb, small, medium, large) using Sharp, optimizes for web delivery (WebP with JPEG fallback), uploads to S3 with server-side encryption, and generates time-limited presigned URLs for CDN-ready delivery.

Queue Infrastructure

- **Redis-Backed Persistence:** Reliable message delivery with at-least-once guarantees
- **Exponential Backoff:** Configurable retry logic with max attempts per queue type
- **Dead-Letter Queues:** Failed jobs archived for manual review and forensics
- **Concurrency Control:** Per-worker concurrency limits (CPU-bound vs I/O-bound optimization)

- **Job Status Dashboard:** Real-time visibility into pending, active, completed, and failed jobs

08 Search & Discovery Engine

BlockEstate's search engine is built on Elasticsearch 8.15 with carefully tuned mappings, analyzers, and query strategies to deliver fast, relevant results across multiple dimensions.

Search Capabilities

Capability	Implementation
Full-Text Search	Multi-field BM25 scoring with stemming, fuzzy matching, and synonym expansion
Geospatial Search	Radius-based queries using geo_point fields with distance sorting
Faceted Navigation	Real-time aggregations for property type, status, and price bucket counts
Autocomplete	City/ZIP prefix matching for type-ahead suggestions
Range Filtering	Price, beds, baths, square footage with composite range queries
Admin Search	Unfiltered access for moderation and compliance review
Archive Search	Query historical/removed listings for audit purposes

Relevance Tuning

Search relevance is fine-tuned through multiple mechanisms:

- **BM25 Scoring:** Term frequency / inverse document frequency for baseline relevance
- **Configurable Boost Profiles:** Admin-adjustable weights for recency, completeness, and agent reputation
- **Feature-Based Ranking:** Signal boosting based on photo count, description length, and listing quality
- **Paid Boost System:** Auction mechanism allowing premium placement for sponsored listings

09 Compliance & Governance

MLS/IDX compliance is a first-class concern in BlockEstate. The platform automates regulatory adherence through a comprehensive compliance engine that evaluates every listing against configurable rule sets.

MLS/IDX Display Requirements

- **Agent & Broker Attribution:** Mandatory display of listing agent name, broker name, and office on all listing pages
- **Copyright & Disclaimers:** Automated insertion of MLS copyright notices and data accuracy disclaimers
- **Last Updated Timestamps:** Visible freshness indicators showing when listing data was last refreshed
- **Opt-Out Enforcement:** Automatic removal within 24 hours when agents/brokers exercise do_not_display rights
- **PII Protection:** Automatic masking of sensitive fields (private remarks, showing instructions, owner contact) from public API responses

Audit Trail System

BlockEstate implements an immutable audit log system with cryptographic hash-chain verification. Every data change, moderation action, and access event is permanently recorded with tamper detection capabilities:

- **Hash-Chain Verification:** Each audit entry's hash incorporates the previous entry's hash, creating a tamper-evident chain
- **Complete Event Capture:** Entity type, action, detailed before/after values, actor identity, timestamp
- **Monthly Compliance Reports:** Automated generation of compliance summaries for MLS provider review
- **Quarterly Audit Readiness:** Pre-built report templates for MLS compliance audits
- **Listing History Tracking:** Field-level change tracking with complete before/after diffs for every modification

Data Security

- **Encryption at Rest:** S3 server-side encryption for all stored media and raw data
- **Encryption in Transit:** TLS for all API communication and inter-service traffic
- **Secrets Management:** Environment-based configuration with AWS Secrets Manager support
- **Input Sanitization:** XSS prevention, SQL injection protection at middleware level

10 Admin Portal & Operations

The admin portal provides comprehensive operational tooling for managing all aspects of the platform. Built as a section within the Next.js application, it offers real-time visibility and control.

Module	Capabilities
Dashboard	Real-time statistics: total listings, sync status, compliance score, active alerts, system health
Listing Management	Search, filter, and bulk-moderate listings with state transition controls and compliance overrides
Compliance Dashboard	Compliance scoring breakdown, rule configuration per MLS source, violation trends
Audit Log Viewer	Searchable, filterable event history with hash-chain verification and export capabilities
Agent Management	Agent directory, opt-out processing, contact information management
Job Monitoring	Pipeline queue status, retry failed jobs, dead-letter queue inspection
Media Management	Upload, delete, regenerate thumbnails for listing photos
Alert Configuration	System alert rules, notification channels, escalation policies
Legal Documents	Version-tracked consent management and terms administration

Emergency Kill Switch

Two-Factor Emergency Delisting: For situations requiring immediate listing removal (legal orders, safety concerns, data breaches), the admin portal includes a kill switch with two-factor confirmation. This triggers immediate removal from all search indices, generates a signed kill token for audit purposes, and records the action in the immutable audit log. The listing transitions to KILLED state with full traceability.

11 Partner Ecosystem & Billing

BlockEstate provides a complete partner ecosystem enabling third-party integrations, reseller partnerships, and monetization through a tiered subscription model.

API Access & Webhooks

- **API Key Management:** Self-service key generation, rotation, and revocation with per-key rate limiting
- **Webhook Subscriptions:** Partners subscribe to listing events (new, updated, removed) with customizable filters
- **Delivery Guarantees:** Automatic retry with exponential backoff, delivery logging, and failure alerting
- **Rate Limiting:** Tiered rate limits per subscription plan (free, starter, professional, enterprise)
- **Usage Metering:** Real-time tracking of API calls, feature access, and data exports per partner

Subscription Plans

Plan	Features
Free	Basic search API access, 100 req/day, no webhooks
Starter	Extended API, 1,000 req/day, basic webhooks, saved searches
Professional	Full API access, 10,000 req/day, all webhooks, bulk exports, analytics
Enterprise	Unlimited API, custom rate limits, white-labeling, SLA, dedicated support

Billing is managed through Stripe integration with automated subscription lifecycle management, invoice generation, and self-service usage dashboards for partners.

12 Security & Access Control

Security is implemented at every layer of the BlockEstate architecture, from network perimeter to application logic to data storage.

Authentication & Authorization

- **JWT Authentication:** Signed tokens with 12-hour expiration for admin sessions
- **RBAC (Role-Based Access Control):** Four roles (Admin, Moderator, Agent, User) with granular permission matrices
- **API Key Authentication:** Optional key-based auth for partner API access with tiered permissions
- **OAuth 2.0:** Client credentials flow for MLS provider authentication with automatic refresh

API Security

Security Layer	Implementation
Helmet	Security headers (CSP, HSTS, X-Frame-Options, etc.)
CORS	Whitelist-based cross-origin request control
Rate Limiting	100 req/min general, 10/15min auth, 30/min admin writes
Input Sanitization	XSS prevention, SQL injection protection
Request Tracing	Correlation ID generation for distributed tracing
PII Masking	Automatic redaction of email/phone for unauthenticated requests
Content-Type Validation	Strict request body type enforcement

13 Infrastructure & DevOps

Local Development Environment

The development environment is fully containerized using Docker Compose, providing a complete local stack that mirrors production:

Service	Purpose	Port
PostgreSQL 16	Primary database with PostGIS	5432
Redis 7	Queue backend & caching	6379
Elasticsearch 8.15	Search engine	9200
MinIO	S3-compatible object storage	9000/9001
Mock RESO Server	Local MLS data provider for testing	4000

Production Deployment

Production infrastructure is managed entirely through code:

- **Terraform (AWS):** Infrastructure-as-code for RDS (PostgreSQL), OpenSearch (Elasticsearch), S3, ECS/Fargate, ALB, VPC, and IAM
- **Kubernetes:** Container orchestration with Helm charts for API, web, and worker deployments with horizontal pod autoscaling
- **Docker:** Multi-stage builds optimized for minimal image size and fast cold starts
- **GitHub Actions:** CI/CD pipeline with linting, type checking, unit testing, Docker build, ECR push, and Kubernetes deployment

Observability

- **Structured Logging:** Pino 9.5 with JSON output, correlation IDs, and log level management
- **Request Tracing:** X-Correlation-ID header propagation across all services
- **Health Checks:** Dedicated /health endpoint for load balancer and Kubernetes readiness/liveness probes
- **SLO Monitoring:** System-level Service Level Objective tracking with error budget alerts
- **Anomaly Detection:** Automated flagging of unusual access patterns and data changes

14 Key Metrics & Results

The BlockEstate platform was delivered as a production-ready system with the following key technical metrics and capabilities:

11+ Packages	**30+** Shared Utilities	**25+** API Endpoints	**22+** DB Migrations
11 Moderation States	**100+** RESO Fields Mapped	**15 min** Sync Interval	**4** Auth Roles

Delivery Highlights

- **End-to-End Type Safety:** Full TypeScript coverage from database to frontend with zero runtime type errors
- **Compliance Automation:** 100% of MLS/IDX display rules enforced programmatically with automated scoring
- **Immutable Audit Trail:** Hash-chain verified logging covering every data mutation and access event
- **Multi-MLS Ready:** Architecture supports unlimited MLS provider connections with independent sync schedules
- **Horizontal Scalability:** Stateless API servers and independent worker pools for elastic scaling
- **Zero-Downtime Deployments:** Elasticsearch index alias swapping and Kubernetes rolling updates
- **Full Infrastructure-as-Code:** Every environment component reproducible through Terraform and Kubernetes manifests
- **Developer Experience:** Turborepo-managed monorepo with mock RESO server for local development

15 Conclusion

BlockEstate represents a comprehensive, enterprise-grade PropTech platform that addresses the full spectrum of challenges in real estate data aggregation, compliance, and consumer experience. CodeVix Labs delivered a production-ready system that transforms fragmented MLS data into actionable real estate intelligence through a sophisticated, multi-stage processing pipeline.

The platform's compliance-first design ensures regulatory adherence through automated scoring, immutable audit trails, and a formal moderation lifecycle. Its partner ecosystem enables monetization through tiered API access, webhook integrations, and white-label capabilities. The modern TypeScript monorepo architecture, infrastructure-as-code deployment, and comprehensive observability tooling ensure long-term maintainability and operational excellence.

BlockEstate demonstrates CodeVix Labs' expertise in building complex, compliance-sensitive platforms that balance technical sophistication with operational pragmatism. The system is designed not just for today's requirements, but as a foundation for continued innovation in the PropTech space.

Authorized by

CodeVix Labs
Engineering Division
February 2026