D&O Renewal Decision Support System

Software Requirements & System Design

Version 1.0 – 20 May 2025

1 Purpose & Scope

This in-house web application assists BHSI underwriters in deciding whether to renew, manually review, or pre-cancel Directors & Officers (D&O) policies written in Spain. It calculates traffic-light scores (green for renew, orange for review, red for pre-cancel) for a company based on publicly-available data covering financial, corporate-governance and legal indicators.

2 Functional Requirements

- Search companies by legal name or VAT number.
- Retrieve data from official public sources (BOE, BORME, Publicidad Concursal, etc.).
- Evaluate parameter blocks: Core-business changes, Turnover & Liquidity, Shareholding, M&A, Bankruptcy, Potential corruption conflicts, Pending legal actions.
- Compute overall traffic-light and per-block traffic-lights using a configurable rules engine.
- Display results and the underlying evidence to underwriters in an intuitive dashboard.
- Allow underwriters to save notes, override scores, and export PDF summaries.
- Store all assessments, overrides and underlying raw data for audit and reporting.
- Provide a REST/GraphQL API to retrieve assessments programmatically.
- Built-in authentication & role-based user management. Admins create users and assign underwriter, admin, auditor roles; users log in with email + password (JWT issued).
- Role-based access control (underwriter, admin, auditor).
- Full audit logging of data pulls, user actions and decision overrides.

3 Non-Functional Requirements

- Performance: ≤ 2 s p95 response for interactive queries.
- Availability: 99.5 % monthly uptime on Google Cloud Run.
- Scalability: auto-scale to 50 concurrent users and 500 daily data pulls.
- Security: OWASP ASVS L2, ISO 27001 controls, GDPR compliance.
- Auditability: immutable logs stored 1 year minimum.
- Maintainability: unit-/integration-tests ≥ 80 % coverage and automated CI/CD.
- Portability: containerised Docker images runnable on any OCI-compliant platform.
- Usability: WCAG 2.1 AA; responsive UI for desktop and tablet.

4 System Requirements

- Client: modern Chromium-based browser (Edge, Chrome) or Safari $\geq$ v15.

- Backend: FastAPI Python 3.12 container on Cloud Run (2 vCPU, 1 GiB baseline).
- Database: Cloud SQL for PostgreSQL 15.
- Storage: Cloud Storage bucket for evidence snapshots and PDF exports.

5 Solution Architecture

A React + Vite single-page application communicates with a FastAPI backend through a REST/JSON API. Data ingestion workers (FastAPI async tasks or Cloud Run Jobs) crawl/consume the public sources and normalise them into the PostgreSQL data model. A rules-based Scoring Engine service calculates traffic-lights. Identity-Aware Proxy (IAP) or Google Identity Platform secures all endpoints.

6 System Components

- Frontend (React 18, Vite, TypeScript, Material UI): search form, results table, parameter cards, notes modal.
- API Gateway (FastAPI, Pydantic models, SqlModel ORM).
- Scoring Engine (rule sets stored in DB, evaluated with python-rule-engine).
- Data Ingestion Adapters (async HTTP & web-scraping routines for each source).
- Database (PostgreSQL – company, parameter, evidence, user, audit tables).
- AuthN/AuthZ (Google Identity Platform/IAP; JWT between frontend & backend).
- Logging & Monitoring (Cloud Logging, Cloud Trace; Prometheus / Grafana optional).

7 Technology Stack

- Frontend: React 18, Vite, TypeScript, Material UI, React Query, Axios, Zod.
- Backend: FastAPI 0.111, Uvicorn, Pydantic v2, SqlModel, asyncpg.
- Task Processing: Cloud Run Jobs or Celery-Kubernetes with Cloud Tasks.
- Database: Cloud SQL PostgreSQL 15.
- CI/CD: GitHub Actions → Cloud Build → Artifact Registry → Cloud Run.
- IaC: Terraform 1.7, Google Provider.
- Containerisation: Docker multi-stage build (distroless).

8 Deployment Strategy

- Develop → PR → GitHub Actions tests & lint.
- On merge to main: Cloud Build builds Docker images and pushes to Artifact Registry.
- Terraform pipeline applies infra changes (Cloud Run service revisions, Cloud SQL).
- Blue/Green rollout managed by Cloud Run traffic splitting; automatic rollback on failed health checks.
- Secrets (DB creds, API keys) stored in Secret Manager and injected as runtime env-vars.

9 Authentication & Authorisation

- Built-in user directory maintained by application admins – users are created, activated/deactivated, and assigned one of three roles (underwriter, admin, auditor) in an “User Management” screen..
- Users sign in with email + strong password (stored as salted / bcrypt hashes).

- Successful login returns a short-lived JWT access-token; the token carries the role claim so the front-end can tailor UI and the back-end can enforce RBAC decorators.
- Security features: minimum-length and complexity rules, password-history, automatic lock-out after N failed attempts, optional TOTP-based MFA, and self-service password-reset via secure e-mail link.
- All auth endpoints sit under /auth/ (POST /auth/login, POST /auth/logout, POST /auth/refresh, POST /auth/password-reset, GET /auth/users etc.).

10 API & Parameter Passing

All endpoints follow REST conventions and JSON Schema 2020-12.

- GET /api/v1/companies?query=<name|vat>&page=1 – search companies
- GET /api/v1/companies/{companyId}/score?detail=true – retrieve traffic-lights and evidence
- POST /api/v1/companies/{companyId}/notes – add/override underwriting note
- POST /api/v1/refresh/{companyId} – trigger asynchronous data pull (admin)

Parameter scores are returned as:
{ "overall": "green|orange|red", "blocks": { "turnover": "green", ... }, "evidence": [{ "source": "BORME", "url": "...", "paragraph": "..." }] }

11 Key Features

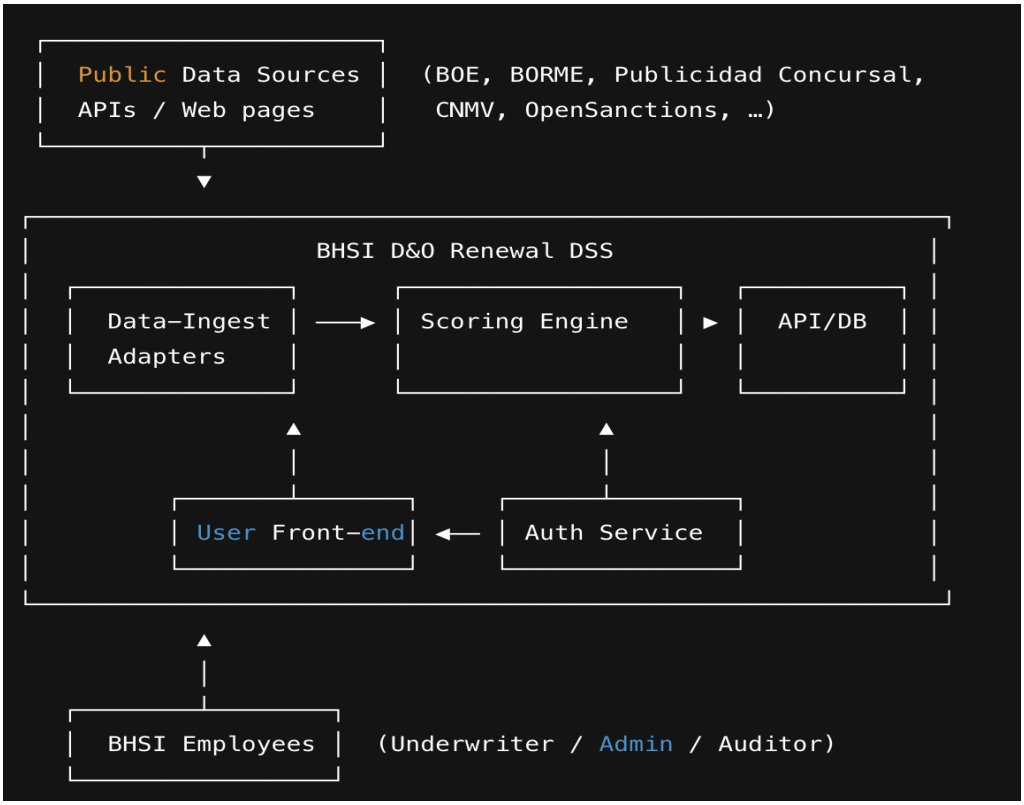
- Real-time traffic-light for any Spanish company using official sources.
- Per-parameter breakdown to justify decisions and ease manual reviews.
- Historical trend graph of scores and underlying financial ratios.
- Exportable PDF summary for file attachment.
- Config-driven rules engine editable by admins without redeploy.

12 Recommended Data Sources

- Publicidad Concursal (bankruptcy stages & restructuration).
- BORME & BOE web services (corporate events, shareholding, M&A).
- CNMV filings for listed companies (governance, liquidity).
- OpenSanctions & OpenCorporates for PEPs, sanctions and UBOs.
- European Business Register (EBR) API for cross-border validation.
- GDELT 2.0 Events & GNews for adverse media.

13 System Architecture Design

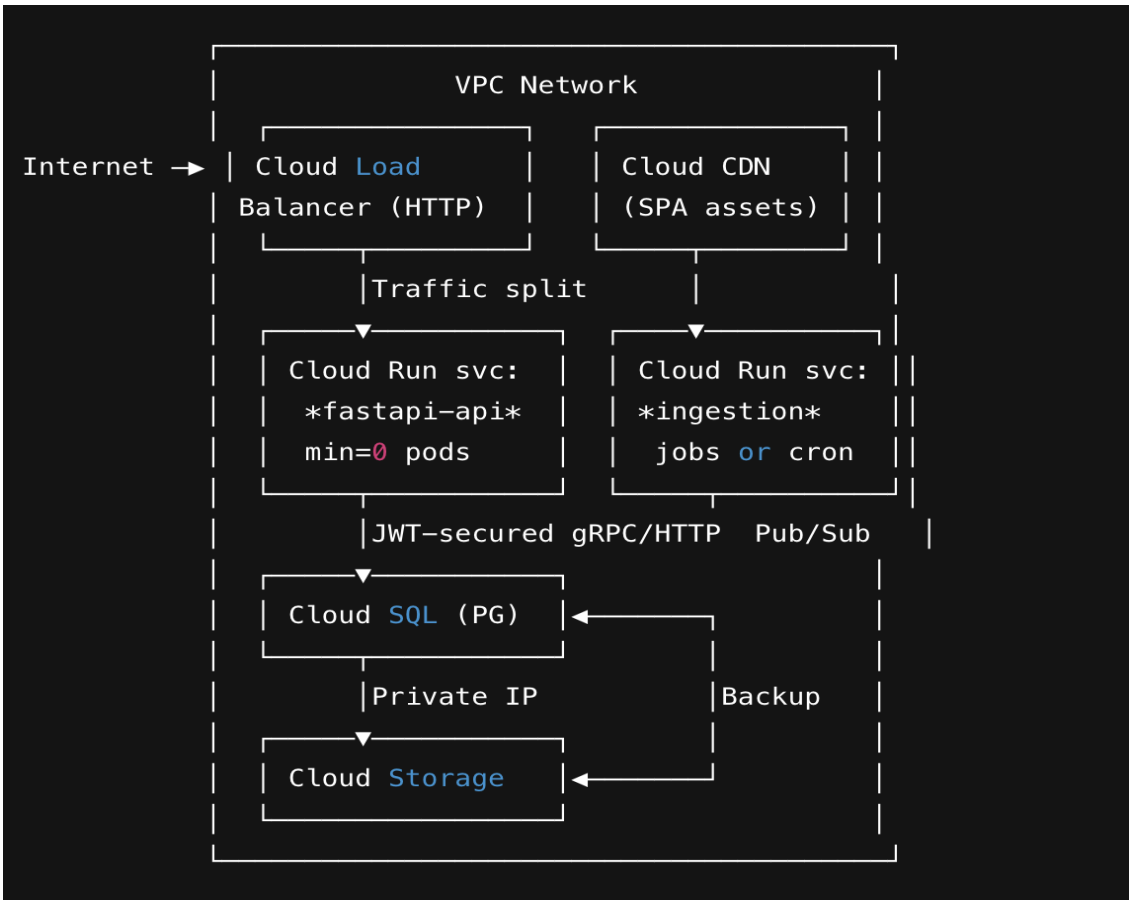
A. Context Diagram



B. Logical Component View

Layer / Domain	Main Components	Responsibility
Presentation	React + Vite SPA, React Router, Material UI	Search, dashboard, parameter cards, note editor
Edge / Auth	FastAPI /auth/* endpoints, JWT service, rate-limit middleware	Email + password login, MFA, token issuance, IP throttling
API Gateway	FastAPI /api/v1/*, Pydantic schema validation, RBAC decorators	Exposes CRUD + score endpoints, orchestrates business logic
Scoring & Rules	Python rule-engine service (same container or separate), YAML/DB-stored rule sets	Calculates overall and per-block traffic lights, exposes score_company(id)
Ingestion	Async tasks (Cloud Run Job) per source, BeautifulSoup / httpx scrapers, PDF-parser	Polls / scrapes sources, normalises records, stores evidence
Persistence	Cloud SQL (PostgreSQL) – company, parameter, evidence, user, audit tables	Durable storage, ACID transactions

C. Deployment View



- Blue/Green rollout: each fastapi-api revision gets 0–100 % traffic via LB.
- Autoscaling: Cloud Run scales 0 → N instances on CPU/RPS; ingestion workers scale by Pub/Sub backlog.
- Secret Manager: DB creds, JWT secret, SMTP creds injected as env-vars.
- VPC-SC & Firewall rules isolate Cloud SQL/private resources.

D. Runtime Sequence

- User hits SPA (served from Cloud CDN).
- Enters email + password → POST /auth/login.
- FastAPI verifies hash, issues JWT (15 min) & refresh token (24 h).
- User searches: GET /api/v1/companies?query=ACME.
- API queries PostgreSQL; if stale or missing, pushes company.refresh message.
- Ingestion Job pulls message, scrapes sources, stores evidence & emits company.scored.
- Scoring Engine recalculates scores; API caches in Redis (optional).
- API returns JSON with overall + per-block lights + evidence.
- Front-end paints traffic light cards; user may add a note (POST /notes).
- All endpoints write audit-log entries (user, action, before/after).