Caso para el TPO: EduScale

EduScale – Plataforma Nacional de Admisiones Educativas con Soporte White-Label y API Pública

Contexto

El Ministerio de Educación de un país latinoamericano lanza un programa de becas para escuelas privadas bilingües. Se estima que **1 millón de familias** intentarán inscribirse **en una ventana de 2 horas** el primer día hábil de septiembre.

La solución debe ser desplegada como una **plataforma SaaS white-label**, es decir:

- Cada institución educativa (colegio, red, fundación) puede **personalizar colores, logo, dominio y mensajes** sin cambiar el core del sistema.
- El **frontend debe ser genérico pero tematizable** (ej: mediante variables CSS, configuración por tenant).
- Al finalizar cada postulación, los datos deben estar disponibles **mediante una API RESTful segura** para:
    - Integración con sistemas internos del colegio (SiMS, ERP, Google Workspace).
    - Consumo por portales de gestión o futuros módulos del Data Lake (aunque el Agente no se implementa en este TPO).

El sistema **cubre las fases A, B y C del Ciclo de Vida del Alumno (CVA)**:

- **Fase A**: Interés/Prospección (visitas web, formularios, campañas).
- **Fase B**: Postulación/Admisiones (documentación, entrevistas, comité, resolución).
- **Fase C**: Inscripción y alta académica/administrativa (matrícula, contrato, alta IT).

Requerimientos Técnicos Clave

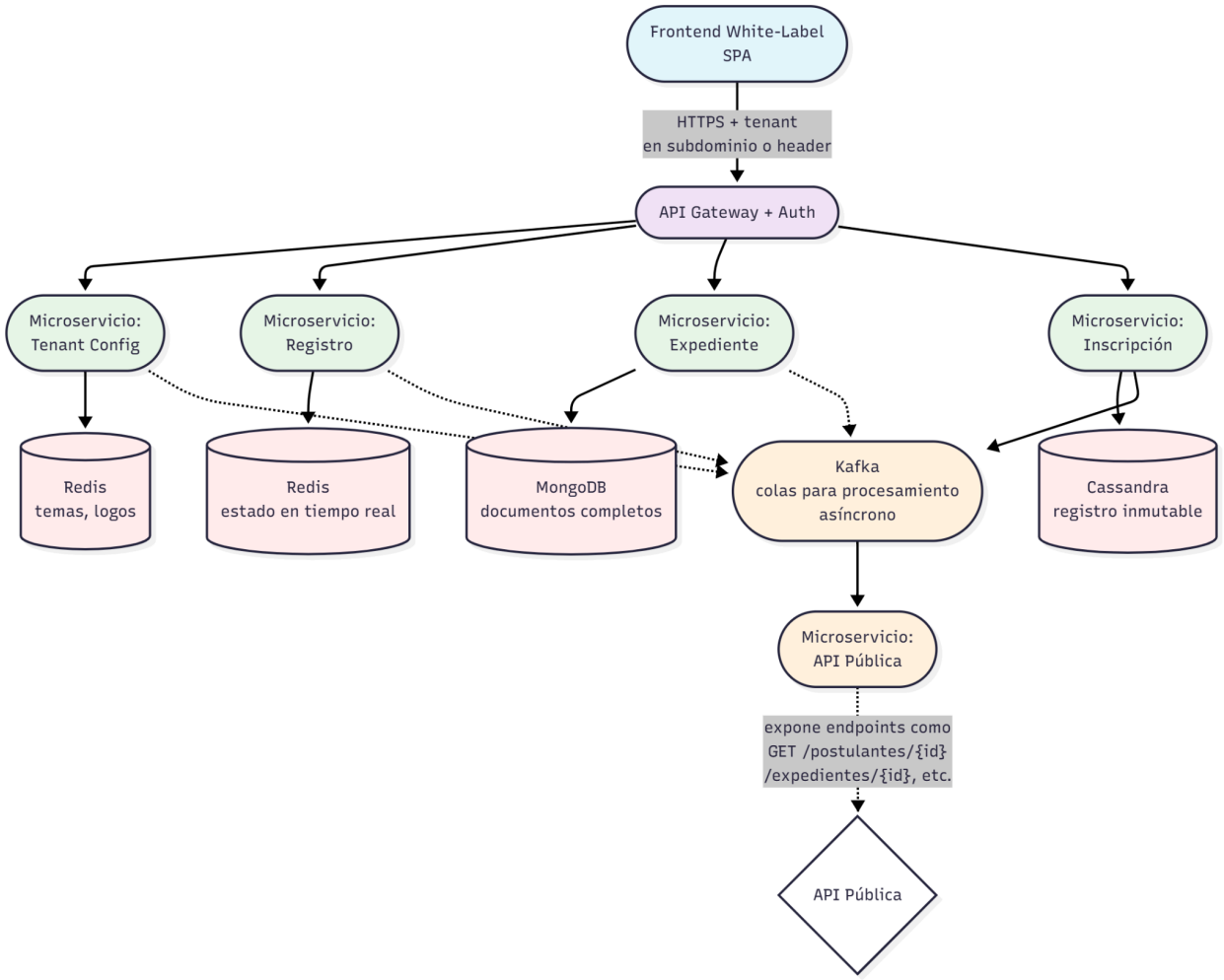
Requisito	Detalle
Fases del CVA cubiertas	A (Interés), B (Admisiones), C (Inscripción y alta)
Volumen extremo	Soportar **1.000.000 de inscripciones simultáneas** en pico (escritura masiva)
Persistencia políglota obligatoria	Uso de **al menos 2 modelos NoSQL distintos** (clave-valor + documental + tabular)
Frontend white-label	SPA (React/Vue) con soporte multi-tenant: temas, logos, textos, flujos ligeros por institución
Backend robusto	Microservicios o serverless, con colas asíncronas (Kafka/Redis Streams)
API pública post-inscripción	RESTful, autenticada (OAuth2/API Key), con endpoints para consultar estado, expediente y matrícula
Alta disponibilidad y tolerancia a fallos	Diseño activo-activo, replicación multi-zona, retry + dead-letter queues

Justificación de Persistencia Políglota

Subsistema	Modelo NoSQL	Tecnología sugerida	Razón
Registro inicial (Fase A)	**Clave-valor**	Redis / DynamoDB	Validación inmediata de duplicados, rate limiting, estado en tiempo real (“Ya iniciaste tu postulación”)
Expediente de admisión (Fase B)	**Documental**	MongoDB / Couchbase	Estructura jerárquica flexible: entrevistas, documentos PDF, decisiones del comité, comentarios
Registro inmutable de inscripciones (Fase C)	**Tabular/columnar**	Cassandra / ScyllaDB	Alta escritura concurrente, inmutabilidad, auditoría, particionamiento por tenant/institución
Configuración white-label	**Clave-valor o documental**	Redis + MongoDB	Temas, logos, textos, URLs por tenant (institución)
Cola de procesamiento asíncrono	**Stream + mensaje**	Kafka + Redis Streams	Desacoplar frontend de validación, generación de contratos, notificaciones

💡 Deben modelar explícitamente los datos por tenant (ej: `institution_id` como clave de particionamiento) y justificar por qué no usan SQL ni un solo motor NoSQL.

Arquitectura Esperada (orientativa)



Entregables del TPO “Realización” (Plan de Sistemas)

Los grupos deberán entregar un Plan de Sistemas que incluya:

1. Análisis del dominio: entidades, flujos, volúmenes, picos (basado en CVA Fases A–C + workflow de admisiones).
2. Justificación de la persistencia políglota: por qué se eligen ≥2 modelos NoSQL y cómo se particionan por tenant.
3. Diseño de la arquitectura: diagrama de componentes, tecnologías, flujos de datos, manejo de multi-tenancy.
4. Modelo de datos por subsistema: esquemas lógicos y físicos (colecciones, tablas, claves, índices).
5. Evaluación CAP: ¿qué se sacrifica y por qué? (ej: consistencia eventual en Fase A, alta disponibilidad en Fase C).
6. Prototipo funcional mínimo:
    a. Frontend white-label (tematizable)
    b. Backend que persiste en ≥2 motores NoSQL
    c. API pública con al menos 2 endpoints (GET /postulante/{id}, GET /inscripcion/{id})
    d. Simulación de carga (opcional, pero valorada)

Vinculación con la Bibliografía y Plataformas

- Harrison (2015): Capítulos sobre escalabilidad, modelos NoSQL y arquitecturas distribuidas.
- Seven Databases: MongoDB (documental), Redis (clave-valor), Cassandra (tabular).
- Plataformas:
    - MongoDB University → modelado de expedientes
    - Redis University → caché de tenant y estado
    - DataStax Academy → escritura masiva en Cassandra

- OWASP → buenas prácticas para APIs públicas