Planning Mejorado - Sistema Tutor IA con Framework Alex Xu

🎯 Fase 0: Análisis y Definición del Sistema (Semana 0)

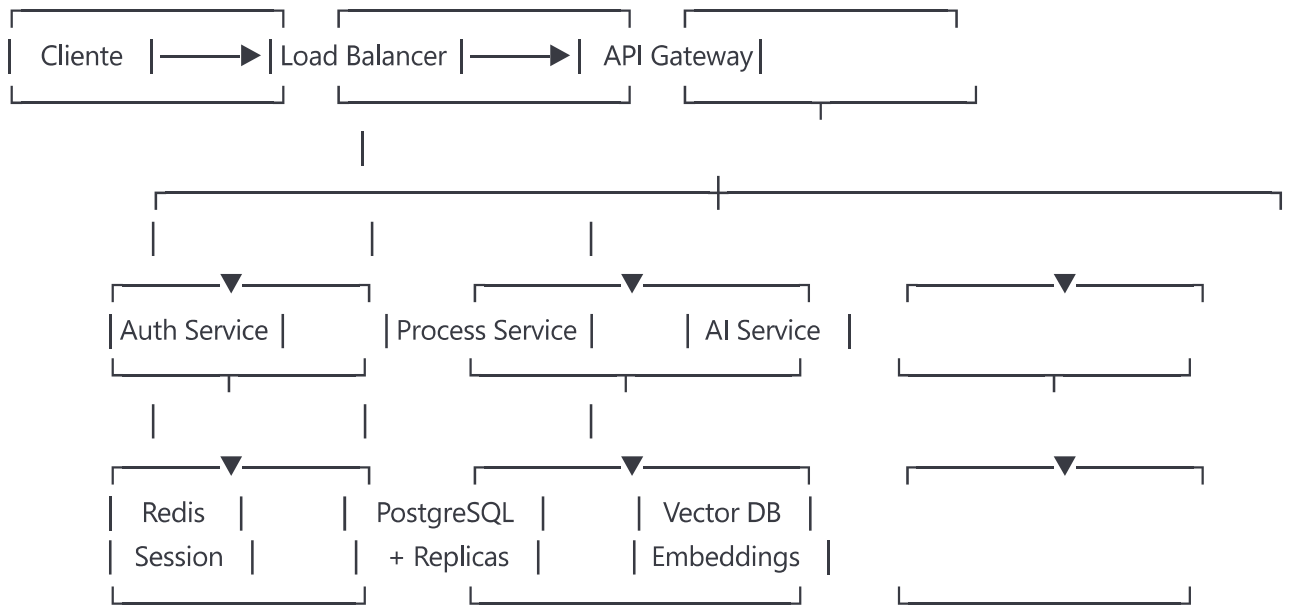
Requisitos No Funcionales

- **Latencia**: P95 < 200ms para respuestas de IA
- **Disponibilidad**: 99.9% uptime (43 minutos downtime/mes)
- **Escalabilidad**: Soportar 10,000 usuarios concurrentes
- **Almacenamiento**: 10TB inicial, crecimiento 100% anual
- **Procesamiento**: 1000 PDFs/hora en hora pico

Estimaciones de Capacidad Inicial

```
Usuarios objetivo año 1: 50,000
DAU esperado: 10,000 (20%)
Requests por usuario: 50/día
Total RPS promedio: 5.8
RPS pico (3x): 17.4
Almacenamiento por usuario: 200MB
Total almacenamiento: 10TB
```

Arquitectura de Alto Nivel



📦 Estructura del Proyecto Mejorada

```
tutor-ia-backend/
├── docker-compose.yml          # Orquestación local
├── k8s/                        # Configuración Kubernetes
│   ├── deployments/
│   ├── services/
│   └── configmaps/
├── src/
│   ├── api-gateway/            # API Gateway y rate limiting
│   ├── services/
│   │   ├── auth/               # Servicio de autenticación
│   │   ├── processor/          # Procesamiento de archivos
│   │   ├── ai/                 # Lógica de IA
│   │   └── analytics/          # Métricas y análisis
│   ├── shared/                 # Código compartido
│   │   ├── database/           # Conexiones y migraciones
│   │   ├── cache/              # Estrategias de caché
│   │   ├── queue/              # Sistema de colas
│   │   └── monitoring/         # Métricas y logs
│   └── infrastructure/         # IaC y configuración
├── tests/
│   ├── unit/
│   ├── integration/
│   └── load/                   # Tests de carga
└── docs/
    ├── architecture/
    ├── api/
    └── runbooks/
```

🚀 Fases de Implementación Actualizadas

FASE 1: Fundación y Arquitectura Base (Semanas 1-2)

Tarea	Responsable	Descripción	Prioridad
1.1	Ambos	Workshop de requisitos y capacidad	P0
1.2	Dev B	Setup Docker + docker-compose con todos los servicios	P0
1.3	Dev B	PostgreSQL con réplicas de lectura	P0
1.4	Dev B	Redis cluster para sesiones y caché	P0
1.5	Dev B	RabbitMQ/Kafka para procesamiento asíncrono	P0
1.6	Dev A	API Gateway con rate limiting	P0
1.7	Dev A	Servicio de autenticación JWT	P0
1.8	Ambos	CI/CD pipeline básico	P1
1.9	Dev B	Monitoring stack (Prometheus + Grafana)	P1
1.10	Dev B	Logging centralizado (ELK stack)	P1

FASE 2: Servicios Core (Semanas 3-4)

Tarea	Responsable	Descripción	Prioridad
2.1	Dev B	Servicio de procesamiento con OCR (Tesseract)	P0
2.2	Dev B	Sistema de colas para procesamiento pesado	P0
2.3	Dev B	Almacenamiento S3-compatible (MinIO)	P0
2.4	Dev A	Integración LLMs con fallback	P0
2.5	Dev A	Vector DB para embeddings	P0
2.6	Dev A	Servicio de análisis educativo	P0
2.7	Ambos	Tests de integración entre servicios	P1
2.8	Dev B	Health checks y circuit breakers	P1

FASE 3: Optimización y Caché (Semanas 5-6)

Tarea	Responsable	Descripción	Prioridad
3.1	Dev B	Caché multi-nivel (L1: app, L2: Redis, L3: CDN)	P0
3.2	Dev B	Optimización queries DB (índices, explain analyze)	P0
3.3	Dev A	Caché de embeddings y prompts frecuentes	P0
3.4	Dev A	Router inteligente para LLMs (costo/calidad)	P0
3.5	Ambos	Load testing y optimización	P1
3.6	Dev B	Database connection pooling	P1

FASE 4: Confiabilidad y Escalabilidad (Semanas 7-8)

Tarea	Responsable	Descripción	Prioridad
4.1	Dev B	Kubernetes config para auto-scaling	P0
4.2	Dev B	Backup automatizado y recovery plan	P0
4.3	Dev B	Blue-green deployment setup	P0
4.4	Ambos	Chaos engineering tests	P1
4.5	Dev B	Rate limiting distribuido	P1
4.6	Dev A	Graceful degradation para servicios IA	P1

FASE 5: Funcionalidades Avanzadas (Semanas 9-10)

Tarea	Responsable	Descripción	Prioridad
5.1	Dev B	WebSockets para real-time con Redis Pub/Sub	P1
5.2	Dev A	Sistema de feedback con ML pipeline	P1
5.3	Dev A	A/B testing framework	P2
5.4	Ambos	Dashboard de métricas de negocio	P2
5.5	Ambos	Documentación completa y runbooks	P1

📊 Métricas y SLOs

SLIs (Service Level Indicators)

- **Latencia API**: P50, P95, P99
- **Tasa de error**: Errores 5xx / Total requests
- **Disponibilidad**: Uptime del servicio
- **Throughput**: Requests procesados/segundo
- **Queue depth**: Trabajos pendientes en cola

SLOs (Service Level Objectives)

- Disponibilidad: 99.9% mensual
- Latencia P95: < 200ms para API calls
- Latencia P95: < 5s para procesamiento PDF
- Error rate: < 0.1%
- Procesamiento PDF: 95% en < 30s

Alertas Críticas

yaml

```
alerts:
  - name: high_error_rate
    condition: error_rate > 1%
    duration: 5m
    action: page_oncall

  - name: high_latency
    condition: p95_latency > 500ms
    duration: 10m
    action: slack_notification

  - name: queue_backup
    condition: queue_depth > 1000
    duration: 5m
    action: auto_scale_workers
```

🔄 Git Workflow Mejorado

Estrategia de Branching

```
main
└── develop
    ├── feature/auth-service (Dev B)
    ├── feature/ai-integration (Dev A)
    └── feature/monitoring (Dev B)
```

Política de Merge

- Feature branches requieren PR review
- Tests automatizados deben pasar
- Coverage mínimo: 80%
- Deploy automático a staging en merge a develop
- Deploy manual a producción desde main

💰 Estimación de Costes Actualizada

Desarrollo (Meses 1-3)

- **Infraestructura**:
  - 3x t3.medium (API, Worker, DB): ~$120/mes
  - Load Balancer: $25/mes
  - S3 + Transfer: $50/mes
  - **Total: ~$195/mes**

Producción (1000 usuarios)

- **Infraestructura**:
  - 2x c5.large (API) con auto-scaling: $170/mes
  - 2x t3.large (Workers): $140/mes
  - RDS PostgreSQL Multi-AZ: $200/mes
  - ElastiCache Redis: $100/mes
  - S3 + CloudFront: $150/mes
  - Monitoring y logs: $50/mes
  - **Total: ~$810/mes**

Escala (10,000 usuarios)

- Multiplicar costos x5-7 con economías de escala
- Considerar reserved instances (-30%)
- Negociar créditos con cloud provider

🎯 KPIs por Sprint

Sprint 1-2 (Fundación)

- ✓ Todos los servicios base running en Docker
- ✓ Pipeline CI/CD funcional
- ✓ Monitoring básico activo

Sprint 3-4 (Core)

- ✓ Procesamiento PDF < 30s
- ✓ API latencia P95 < 200ms
- ✓ 0 puntos únicos de falla

Sprint 5-6 (Optimización)

- ✓ Cache hit ratio > 80%
- ✓ DB queries optimizadas < 50ms
- ✓ Load test: 100 usuarios concurrentes

Sprint 7-8 (Confiabilidad)

- ✓ Auto-scaling funcional
- ✓ Recovery time < 5 minutos

- ✓ 99.9% uptime en staging

Sprint 9-10 (Polish)

- ✓ Todas las features documentadas
- ✓ Runbooks completos
- ✓ Dashboard métricas negocio

📝 Decisiones de Arquitectura (ADRs)

ADR-001: Monolito Modular vs Microservicios

Decisión: Monolito modular con servicios separados **Razón**: Balance entre simplicidad inicial y escalabilidad futura **Trade-offs**: Mayor acoplamiento inicial vs menor complejidad operacional

ADR-002: PostgreSQL vs NoSQL

Decisión: PostgreSQL con JSONB para flexibilidad **Razón**: ACID para datos críticos, JSONB para esquemas flexibles **Trade-offs**: Menor rendimiento en escrituras masivas

ADR-003: Estrategia de Caché

Decisión: Redis + CDN + Application cache **Razón**: Minimizar latencia y carga en DB **Trade-offs**: Complejidad de invalidación

🚨 Riesgos y Mitigaciones

Riesgo	Impacto	Probabilidad	Mitigación
Límites API LLM	Alto	Media	Multiple providers + fallback
Crecimiento inesperado	Alto	Baja	Auto-scaling + alerts
Costos LLM elevados	Medio	Alta	Cache agresivo + router inteligente
Seguridad datos estudiantes	Alto	Baja	Encriptación + compliance GDPR

✅ Checklist Pre-Producción

- [ ] Load testing completado (1000 usuarios concurrentes)
- [ ] Backup y recovery probado
- [ ] Monitoreo completo configurado
- [ ] Runbooks para incidentes comunes
- [ ] Seguridad auditada
- [ ] GDPR compliance verificado
- [ ] SLOs definidos y medidos
- [ ] Documentación API completa
- [ ] Plan de rollback definido

- [ ] Equipo on-call configurado