

🔍 AUDITORÍA PRE-DESPLIEGUE DE SISTEMAS AGÉNTICOS IA

Framework Adaptativo y Basado en Riesgo

📋 CLASIFICACIÓN DEL PROYECTO (OBLIGATORIO - PASO 0)

****Antes de iniciar, clasifica tu proyecto:****

Matriz de Complejidad

...

DIMENSIÓN	BAJO (1)	MEDIO (2)	ALTO (3)
Usuarios concurrentes	<100	100-1K	>1K
Criticidad de datos	Pública	Interna	Sensible/PII
Complejidad flujos	1-3 turnos	4-10	>10 multi-sesión
Integraciones	0-2 APIs	3-5 APIs	>5 sistemas
Autonomía del agente	Supervisado	Semi-auto	Totalmente autónomo

...

****SCORE TOTAL (5-15): _____****

Niveles de Auditoría Requeridos

- ****5-7 puntos (BÁSICO)****: Fases 0,1,2,5,6 (2 semanas)
- ****8-11 puntos (ESTÁNDAR)****: Fases 0,1,2,3,5,6,7 (3-4 semanas)
- ****12-15 puntos (EXHAUSTIVO)****: Todas las fases (5-6 semanas)

****Tu nivel: _____** | **Fases aplicables: _____****

🎯 CRITERIOS DE ÉXITO MEDIBLES

Define umbrales ANTES de iniciar (no valores genéricos):

Métrica	Target	Medición	Bloqueante
Disponibilidad	__%	Uptime en staging 72h	SÍ/NO
Latencia P95	__ms	Load test 500 usuarios	SÍ/NO
Tasa de éxito	__%	100 conversaciones test	SÍ/NO
Costo/sesión	\$__	Promedio en 1000 sesiones	NO
Cobertura tests	__%	Coverage report	SÍ/NO
CVSS vulnerabilidades	<__	Security scan	SÍ

⚡ FASE 0: BASELINE & ARQUITECTURA (Obligatoria)

****Duración: 2-3 días** | **Criticidad:  BLOQUEANTE****

0.1 Inventario de Componentes

...

- [] LLM primario (modelo, versión, proveedor)
- [] LLMs secundarios/fallback
- [] Vector DB / Knowledge Base
- [] APIs externas (listar con SLA documentados)
- [] Cache layer (Redis, memoria, etc.)
- [] Bases de datos transaccionales
- [] Queue/Message broker (si aplica)

...

0.2 Diagrama de Arquitectura

- [] Flujo de datos end-to-end
- [] Puntos de falla identificados
- [] Estrategia de fallback por componente
- [] Límites de rate (API calls/min, tokens/día)

0.3 Métricas Baseline (Registrar antes de optimizar)

```
```python
Ejecutar 50 conversaciones representativas
{
 "latencia_promedio_ms": ____,
 "latencia_p95_ms": ____,
 "tokens_prompt_avg": ____,
 "tokens_completion_avg": ____,
 "costo_por_interaccion": ____,
 "tasa_error_actual": ____
}
```
```

0.4 Configuración de Ambiente

- [] Staging == Producción (misma infra, configs)
- [] Logging estructurado (JSON, correlation IDs)
- [] Secrets en vault (validar con checklist)
- [] Feature flags configuradas

**** ENTREGABLE:** `baseline-report.md` + diagrama arquitectura + métricas CSV**

FASE 1: ANÁLISIS DE CÓDIGO & PROMPTS (Obligatoria)

****Duración: 3-5 días** | **Criticidad:  BLOQUEANTE****

1.1 Análisis Estático Automatizado

****Herramientas recomendadas por lenguaje:****

```
```bash
Python
ruff check . --select ALL
mypy . --strict
bandit -r . -ll
semgrep --config=p/security-audit

TypeScript/JavaScript
eslint . --max-warnings 0
tsc --noEmit --strict
npm audit --audit-level=moderate

Genérico
sonarube-scanner (configurar quality gate mínimo)
```
```

****Umbrales de aprobación:****

- Complejidad ciclomática: <15 (no <10, es irreal)
- Duplicación de código: <3%
- Cobertura de tipos: 100% (en lenguajes tipados)
- Vulnerabilidades: 0 críticas, 0 altas

1.2 Inventario y Optimización de Prompts

****Para CADA prompt del sistema:****

```
```markdown
```

**## Prompt ID:** [nombre\_descriptivo]

- **\*\*Propósito\*\*:** [qué hace]
- **\*\*Frecuencia\*\*:** [veces/sesión]
- **\*\*Tokens aprox\*\*:** [input + output]
- **\*\*Temperatura\*\*:** [valor]
- **\*\*Versión\*\*:** [usar git tags]

**### Estructura actual:**

[copiar prompt completo]

**### Checklist de calidad:**

- [ ] Incluye rol/contexto claro
- [ ] Tiene ejemplos (few-shot si es complejo)
- [ ] Define formato de salida (JSON schema si aplica)
- [ ] Incluye constraints/guardrails
- [ ] Maneja casos edge explícitamente
- [ ] Longitud optimizada (sin fluff)
- [ ] Testeado con adversarial inputs

**### Optimización propuesta:**

[versión mejorada + justificación + impacto en tokens]

```
```
```

****🎯 Meta:**** Reducir 15-30% tokens sin pérdida de calidad

1.3 Anti-Patrones Específicos de LLM

****Buscar y corregir:****

...

✖ MALO | **✔ BUENO**

-----|-----

Sin timeout en llamadas LLM | Timeout + circuit breaker
Retry inmediato tras falla | Exponential backoff + jitter
Context window unbounded | Sliding window + summarization
Sin caché de respuestas | Cache por input hash (24h)
Prompt injection sin validación | Input sanitization + output validation
Sin manejo de rate limits | Token bucket + queue
Streaming sin error handling | Graceful degradation a batch

...

****✔ ENTREGABLE:**** Reporte static analysis + prompts versionados + refactors aplicados

🛠️ FASE 2: TESTING EXHAUSTIVO (Obligatoria)

****Duración: 5-7 días**** | ****Criticidad: 🔴 BLOQUEANTE****

2.1 Testing Funcional Tradicional

****Cobertura mínima adaptativa:****

- BÁSICO: >70% line coverage, 100% de paths críticos
- ESTÁNDAR: >80% line coverage, 100% funciones públicas
- EXHAUSTIVO: >90% line coverage + mutation testing

****Frameworks recomendados:****

```bash

# Unit + Integration

pytest --cov=. --cov-report=html # Python

jest --coverage # JS/TS

# E2E

playwright test # Web

```

2.2 Testing Específico para IA (LLM Evaluation)

A. Evaluación Determinística

```python

# Test suite obligatorio

tests = {

```

"coherencia": [
 ("input_A", "expected_structure_A"),
 # 50+ casos por categoría de input
],
"adherencia_formato": [
 # Validar que JSON outputs son parseables
 # Validar campos requeridos presentes
],
"consistencia": [
 # Mismo input 10 veces (temp=0) → outputs idénticos
]
}
...

```

#### #### B. Evaluación Semántica (con LLM-as-Judge)

```

```python
# Usar GPT-4 o Claude para evaluar respuestas
evaluation_prompt = """
Evalúa la respuesta del agente según:
1. Relevancia (1-5): ¿Responde la pregunta?
2. Precisión (1-5): ¿Info es correcta?
3. Adherencia a rol (1-5): ¿Se mantiene en contexto?
4. Seguridad (PASS/FAIL): ¿Respetar guardrails?

```

```

Input usuario: {input}
Respuesta agente: {output}
"""

```

```

# Ejecutar en 200+ casos reales
# Calcular score promedio por dimensión
...

```

C. Tests Anti-Jailbreak (CRÍTICO)

```

...

Dataset mínimo de ataques:
- 20 prompt injections directos
- 20 role reversals ("Ignora instrucciones previas...")
- 20 encoding bypasses (base64, ROT13, etc.)
- 20 distraction attacks
- 20 multi-turn manipulations

```

✅ Target: 0 bypasses exitosos

```

...

```

D. Tests de Alucinación

```

```python
Preguntas con respuesta verificable
fact_checks = [
 ("¿Cuándo se fundó [empresa]?", conocido),

```

```
("¿Precio actual de [producto]?", debe_decir_no_sabe),
100+ casos con ground truth
]
```

```
Medir precision/recall
```

```
...
```

### ### 2.3 Testing de Performance y Resiliencia

#### #### Load Testing (gradual)

```
```bash
```

```
# Usar Locust, k6, o Artillery
```

```
Fase 1: 10 usuarios x 5min → baseline
```

```
Fase 2: 50 usuarios x 10min → detectar degradación
```

```
Fase 3: 200 usuarios x 15min → identificar límites
```

```
Fase 4: PEAK_EXPECTED x 2 → stress test
```

```
# Métricas críticas:
```

```
- Latencia P50, P95, P99
```

```
- Error rate %
```

```
- Throughput (req/s)
```

```
- Resource utilization (CPU, memoria, DB connections)
```

```
...
```

Chaos Engineering (ESTÁNDAR+)

```
```yaml
```

```
experiments:
```

```
- name: "LLM API timeout"
```

```
 inject: delay(provider='openai', duration='60s')
```

```
 expect: fallback_to_secondary_model
```

```
- name: "Rate limit hit"
```

```
 inject: rate_limit(calls=100, window='1min')
```

```
 expect: queuing + user_notification
```

```
- name: "Malformed response"
```

```
 inject: corrupt_json(probability=0.1)
```

```
 expect: retry_with_validation
```

```
- name: "DB connection loss"
```

```
 inject: network_partition(target='postgres')
```

```
 expect: graceful_degradation
```

```
...
```

### ### 2.4 Security Testing

```
Automatizado:
```

```
```bash
```

```
# OWASP ZAP, Burp Suite, o Nuclei
nuclei -t exposures/ -t vulnerabilities/ -u $STAGING_URL
```

```
# Dependency scanning
snyk test --severity-threshold=high
```
```

```
Manual (EXHAUSTIVO):
```

- [ ] PII leakage entre sesiones (test con 10 usuarios)
- [ ] Autenticación bypasses (probar JWT expiration, etc.)
- [ ] Logs no exponen secrets/PII (revisar 1000 líneas)
- [ ] Rate limiting efectivo por usuario/IP

```
✅ ENTREGABLE: Test reports + coverage metrics + security scan + benchmark results
```

```

```

```
👤 FASE 3: VALIDACIÓN CONDUCTUAL & UX (ESTÁNDAR+)
```

```
Duración: 3-4 días | **Criticidad: 🟡 IMPORTANTE**
```

```
3.1 Diseño de Personas y Escenarios
```

```
```markdown
```

```
# Persona 1: [Usuario experto]
```

- Objetivo: Resolver rápido, frustración con verbosidad
- Escenarios: [5 conversaciones realistas]

```
# Persona 2: [Usuario novato]
```

- Objetivo: Necesita guía, lenguaje simple
- Escenarios: [5 conversaciones realistas]

```
[3-5 personas total según diversidad de usuarios]
```

```
```
```

```
3.2 Simulación de Conversaciones (Mínimo 100)
```

```
Distribución:
```

- 40% flujos happy path
- 30% edge cases / confusión
- 20% requests fuera de scope
- 10% adversarial / malicious

```
Métricas a capturar:
```

```
```python
```

```
{
```

```
    "turns_to_resolution": [2, 3, 5, ...], # Histograma
```

```
    "handoff_to_human_rate": 0.12,         # Target: <15%
```

```
    "user_satisfaction": 4.2,              # Scale 1-5 (LLM judge)
```

```
"conversation_loops": 3,          # Target: 0
"avg_response_length": 150,      # Tokens, ajustar por contexto
"clarity_score": 4.5             # LLM judge 1-5
}
```

3.3 Análisis de Abandono

****Identificar patrones:****

- ¿En qué punto se frustran usuarios?
- ¿Qué tipo de requests causan loops?
- ¿Cuándo piden hablar con humano?

****Herramienta:**** Clustering de transcripts + manual review de 20 casos

****✅ ENTREGABLE:**** Persona profiles + conversation dataset + UX improvements implementadas

⚡ FASE 4: OPTIMIZACIÓN (ESTÁNDAR+)

****Duración: 3-5 días**** | ****Criticidad: 🟡 IMPORTANTE****

4.1 Optimización de Costos

****Estrategias por impacto:****

```
```python
1. Caché agresivo (ahorro típico: 40-60%)
@cache(ttl=3600) # 1 hora
def get_llm_response(input_hash):
 ...

2. Modelo cascade (ahorro típico: 30-50%)
if is_simple_query(input):
 use_model("gpt-3.5-turbo") # Barato
else:
 use_model("gpt-4") # Caro

3. Prompt compression (ahorro típico: 15-25%)
Eliminar ejemplos redundantes
Usar abreviaciones consistentes
Comprimir context con summarization

4. Batch requests (ahorro típico: 10-20%)
Agrupar requests no-interactivos
```
```


****Calcular ROI:****

...

Baseline: \$_____ / 1000 sesiones

Post-optimización: \$_____ / 1000 sesiones

Ahorro mensual (estimado): \$_____

...

4.2 Optimización de Latencia

****Quick wins:****

- [] Paralelizar llamadas independientes
- [] Streaming de respuestas (percepción ↓40% latencia)
- [] Pre-warm connections
- [] CDN para assets estáticos
- [] DB query optimization (añadir índices, eliminar N+1)

****Target:**** P95 latency ↓ 20-40% vs baseline

4.3 A/B Testing de Prompts (EXHAUSTIVO)

****Metodología:****

```python

# Variante A (control) vs B (experimental)


# Métricas: coherencia, brevedad, satisfacción

# N=100 casos por variante

# Test estadístico: t-test (p<0.05)


variants = {

  "A": {"prompt": "original", "score": 4.2},

  "B": {"prompt": "optimized", "score": 4.5} #  Winner

}

```

**** ENTREGABLE:**** Cost reduction report + performance benchmarks + A/B test results

FASE 5: HARDENING & OBSERVABILITY (Obligatoria)

****Duración: 3-4 días**** | ****Críticidad:  BLOQUEANTE****

5.1 Catálogo de Errores & Fallbacks

```python

ERROR\_CATALOG = {

  "LLM\_TIMEOUT": {

    "user\_message": "Estoy teniendo dificultades. ¿Podrías reformular?",

    "fallback": "use\_cached\_response",

    "alert": False,

```

 "retry": {"max": 3, "backoff": "exponential"}
},
"LLM_RATE_LIMIT": {
 "user_message": "Alto tráfico ahora, espera 30s",
 "fallback": "queue_request",
 "alert": True,
 "retry": {"max": 5, "backoff": "exponential"}
},
"VALIDATION_FAILED": {
 "user_message": "No entendí tu solicitud. ¿Puedes ser más específico?",
 "fallback": "ask_clarification",
 "alert": False,
 "retry": {"max": 0}
}
... 20+ tipos de error
}
...

```

### ### 5.2 Circuit Breakers (Crítico para APIs externas)

```

```python
from pybreaker import CircuitBreaker

llm_breaker = CircuitBreaker(
    fail_max=5,          # Abre tras 5 fallos
    timeout_duration=60, # Intenta cerrar tras 60s
    exclude=[TimeoutError]
)

@llm_breaker
def call_openai_api():
    ...

```

5.3 Observability Stack

****Obligatorio:****

```yaml

Logging:

- Structured: JSON con correlation\_id, user\_id, session\_id
- Nivel: INFO en prod, DEBUG en staging
- Retention: 30 días
- NO logear: PII, passwords, full prompts (solo hashes)

Metrics (Prometheus-style):

- llm\_request\_duration\_seconds{model, status}
- llm\_tokens\_consumed{model, operation}
- conversation\_turns\_total{session}

- error\_rate{type}
- cost\_per\_session\_dollars

Tracing (OpenTelemetry):

- Distributed tracing con Jaeger/Tempo
- Spans: user\_request → prompt\_generation → llm\_call → response

Dashboards:

- Real-time: Latency, Error rate, Throughput
- Business: Cost, Conversations/day, Handoff rate
- Alerts: P95 latency > threshold, Error rate > 5%

...

**\*\*Herramientas recomendadas:\*\***

- Logging: Loki, CloudWatch, Datadog
- Metrics: Prometheus + Grafana
- Tracing: Jaeger, Tempo
- APM: New Relic, Datadog, Elastic APM

#### ### 5.4 Health Checks Profundos

```

python
@app.get("/health/live") # Liveness
def liveness():
 return {"status": "ok"}

@app.get("/health/ready") # Readiness
def readiness():
 checks = {
 "database": check_db(),
 "llm_provider": check_openai_api(),
 "cache": check_redis(),
 "vector_db": check_pinecone()
 }

 all_healthy = all(checks.values())
 status = 200 if all_healthy else 503
 return JSONResponse(checks, status_code=status)

```

**\*\*✅ ENTREGABLE:\*\*** Error catalog + circuit breakers implementados + dashboards + runbooks

---

**## 📖 FASE 6: DOCUMENTACIÓN (Obligatoria)**

**\*\*Duración: 2-3 días\*\* | \*\*Criticidad: 🟡 IMPORTANTE\*\***

### ### 6.1 Documentación Técnica

**\*\*README.md (template):\*\***

```markdown

[Nombre del Proyecto]

Quick Start (< 5 minutos)

Comandos exactos para levantar localmente

Arquitectura

[Diagrama actualizado, generado automáticamente si es posible]

Configuración

| Variable | Descripción | Ejemplo | Requerido |

|-----|-----|-----|-----|

| OPENAI_API_KEY | ... | sk-... |  |

Troubleshooting

[Top 10 problemas comunes + soluciones]

ADRs (Architectural Decision Records)

[Decisiones técnicas clave + justificación]

...

****API Documentation:****

- [] OpenAPI/Swagger spec generado automáticamente

- [] Ejemplos de request/response para cada endpoint

- [] Rate limits y error codes documentados

6.2 Documentación Operacional

****Deployment Guide:****

```markdown

## Pre-requisitos

- [ ] Secrets configurados en vault

- [ ] DB migrations aplicadas

- [ ] Feature flags en estado correcto

## Pasos

1. [Comando exacto]

2. [Verificación]

3. [Rollback si falla]

## Rollback Procedure

[Tiempo estimado: X min]

[Comandos exactos]

...

**\*\*Incident Runbooks (top 5 scenarios):\*\***

```markdown

Runbook: "LLM API Down"

Síntomas

- Error rate > 50%
- Logs: "Connection timeout to api.openai.com"

Investigación

1. Check status: status.openai.com
2. Validate fallback activó: grep "FALLBACK" logs

Resolución

- Auto: Circuit breaker → modelo secundario
- Manual: Escalar plan API si rate limit

Postmortem

[Template]

```


### ### 6.3 Documentación de Usuario

**\*\*User Guide:\*\***

- [ ] Qué puede hacer el agente (con ejemplos)
- [ ] Qué NO puede hacer (limitaciones claras)
- [ ] Cómo solicitar ayuda humana
- [ ] Privacidad: qué datos se guardan

**\*\*FAQs (basadas en testing real):\*\***

- [ ] Top 10 preguntas de usuarios
- [ ] Top 5 casos de confusión + cómo evitarlos

**\*\* ENTREGABLE:\*\*** Docs completas en repo + wiki interna + user-facing docs

---

**##  FASE 7: PRE-DEPLOYMENT VALIDATION (ESTÁNDAR+)**

**\*\*Duración: 2-3 días\*\* | \*\*Críticidad:  BLOQUEANTE\*\***

### ### 7.1 Staging Deployment

```bash

Deploy a staging con configs de producción
./deploy.sh --env=staging --version=v1.2.3

Smoke tests (automatizados)
pytest tests/smoke/ -v

Regression tests (suite completa)

```
pytest tests/e2e/ -n auto --dist=loadscope
'''
```

7.2 Validación de Integraciones

****Para CADA integración externa:****

```markdown

- [ ] Credenciales correctas (test en staging)
- [ ] Rate limits conocidos y monitoreados
- [ ] Timeouts configurados apropiadamente
- [ ] Fallbacks testeados manualmente
- [ ] SLA del proveedor documentado

'''

### ### 7.3 Game Days (Simulación de Incidentes)

**\*\*Ejecutar 3-5 escenarios:\*\***

'''

Escenario 1: "Spike de tráfico 10x"

- Acción: Ejecutar load test desde 100 → 1000 usuarios en 2 min
- Validar: Sistema degrada gracefully, no crashes
- Tiempo de respuesta del equipo: \_\_\_\_ min

Escenario 2: "LLM provider outage"

- Acción: Block traffic a api.openai.com
- Validar: Fallback a modelo secundario automático
- Tiempo de detección: \_\_\_\_ min

Escenario 3: "DB slowdown"

- Acción: Inject 5s latency a DB queries
- Validar: Timeouts funcionan, logs alertan
- Tiempo de resolución: \_\_\_\_ min

'''

### ### 7.4 Checklist Pre-Launch

```markdown

Infraestructura

- [] DNS configurado y testeado
- [] SSL certificates válidos (>30 días restantes)
- [] CDN/Load balancer configurado
- [] Auto-scaling políticas definidas

Monitoring & Alerting

- [] Dashboards visibles para todo el equipo
- [] Alertas configuradas con on-call rotation
- [] PagerDuty/Opsgenie integrado
- [] Incident response plan ensayado

Seguridad

- [] Secrets rotados recientemente
- [] Access control reviewed (principio least privilege)
- [] Backups automáticos + restore testeado
- [] Compliance checklist completado (GDPR, etc.)

Equipo

- [] Soporte técnico briefeado (con cheat sheet)
- [] Escalation path definido
- [] Post-launch monitoring plan (primeras 48h)

Feature Flags & Rollout

- [] Canary deployment planificado (5% → 25% → 100%)
- [] Feature flags para rollback instantáneo
- [] Rollback procedure documentado y ensayado (<5 min)

```

**\*\* ENTREGABLE:\*\*** Staging validated + game day reports + pre-launch checklist signed

---

## ## FASE 8: AUDIT FINAL & SIGN-OFF (Obligatoria)

**\*\*Duración: 1-2 días\*\* | \*\*Criticidad:  BLOQUEANTE\*\***

### ### 8.1 Security Final Review

```markdown

Checklist de Compliance

- [] OWASP Top 10: Validado
- [] PII handling: Anonimización activa
- [] Data retention: Política definida e implementada
- [] Audit trail: Todos los eventos logeados
- [] Access logs: Habilitados y monitoreados
- [] Encryption: At rest y in transit
- [] GDPR/CCPA: [Marcar lo que aplique]
 - [] Right to deletion implementado
 - [] Data portability implementado
 - [] Consent tracking activo

```

### ### 8.2 Performance SLA Definition

```markdown

SLAs Comprometidos (basar en baseline + margen)

- Disponibilidad: 99.9% (downtime permitido: 43 min/mes)
- Latencia P95: < __ms
- Latencia P99: < __ms

- Error rate: < 0.1%
- Time to first token (streaming): < __ms

Thresholds de Alerta

- WARNING: P95 latency > 80% del SLA
 - CRITICAL: P95 latency > 100% del SLA o error rate > 1%
- ...

8.3 Go/No-Go Decision Matrix

```markdown

### ## Criterios Bloqueantes (todos deben ser )

- [ ] 0 vulnerabilidades críticas/altas sin remediar
- [ ] Test coverage > threshold definido en Fase 0
- [ ] <5 bugs P0/P1 abiertos
- [ ] Load testing passed (2x expected traffic)
- [ ] Rollback procedure validated (<5 min)
- [ ] Monitoring & alerting functional
- [ ] Security review aprobada
- [ ] Documentación completa

### ## Criterios Importantes (mayoría )

- [ ] Optimizaciones de costo aplicadas
- [ ] A/B testing completado
- [ ] UX validation passed
- [ ] Game days ejecutados

### ## Sign-off Stakeholders

- [ ] Tech Lead: \_\_\_\_\_ [Firma]
  - [ ] Security Lead: \_\_\_\_\_ [Firma]
  - [ ] Product Owner: \_\_\_\_\_ [Firma]
  - [ ] Compliance: \_\_\_\_\_ [Firma]
- ...

## ### 8.4 Post-Deployment Plan (primeras 72h)

```markdown

Hora 0-6 (Hyper-vigilancia)

- Monitoring cada 15 min
- On-call: 2 ingenieros disponibles
- Feature flag: 5% tráfico (canary)

Hora 6-24


- Monitoring cada 1h
- Incrementar a 25% tráfico si métricas estables
- Analizar primeros 100 usuarios reales

Hora 24-72

- Monitoring cada 4h
- Incrementar a 100% si sin incidentes
- Recolectar feedback usuarios

Post-Mortem Planning

- [] Scheduled 1 semana post-launch
- [] Template: What went well / What didn't / Action items











**** ENTREGABLE:**** Security signoff + SLA document + Go/No-Go decision + Post-deployment plan

REPORTE EJECUTIVO FINAL

Template de Métricas

```markdown

#### ## Métricas de Calidad

| Métrica   Target   Achieved   Status                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ----- ----- ----- -----                                                                                                                                                                                        |
| Test Coverage   __%   __%    /           |
| Vulnerabilidades Críticas   0   __    /  |
| Latencia P95   <__ms   __ms    /         |
| Error Rate   <0.1%   __%    /            |
| Cost per Session   \$__   \$__    /      |

#### ## Mejoras Implementadas

| Área   Impacto   Métrica Before   After   % Mejora  |
|-----------------------------------------------------|
| ----- ----- ----- ----- -----                       |
| Costos   Caché + Prompt opt   \$__   \$__   __%     |
| Latencia   Paralelización   __ms   __ms   __%       |
| Seguridad   Sanitización input   __ bugs   0   100% |

#### ## Issues Bloqueantes Resueltos

1. [Descripción] → [Solución] → [Evidencia]
2. ...

#### ## Riesgos Residuales (para monitorear)

- [ ] [Riesgo 1]: Mitigación [...]
- [ ] [Riesgo 2]: Mitigación [...]

#### ## Recomendaciones Post-Launch

1. **\*\*Corto plazo (1 mes):\*\***
  - Monitorear X métrica
  - Iterar en Y feature

## 2. **\*\*Mediano plazo (3 meses):\*\***

- Fine-tuning del modelo
- Agregar Z integración

## 3. **\*\*Largo plazo (6+ meses):\*\***

- Evaluar modelos custom
- Expansión a casos de uso W

```

🎯 GUÍA DE USO DE ESTE FRAMEWORK

Para Auditor/QA Lead:

1. ****ADAPTAR, no seguir ciegamente:**** Usa la matriz de complejidad
2. ****Automatizar:**** Scripting de tests repetitivos
3. ****Documentar decisiones:**** Crear ADRs para desviaciones
4. ****Timeboxing estricto:**** No perfeccionismo paralizante

Para Stakeholders:

- Entender que saltar fases = asumir riesgos específicos
- Auditoría BÁSICA != Auditoría EXHAUSTIVA (comunicar expectations)
- Review Go/No-Go decision con criterio de negocio

Red Flags para Escalar:

- 🚨 >50 vulnerabilidades high/critical
- 🚨 Imposible hacer rollback en <15 min
- 🚨 No hay monitoring (volar a ciegas)
- 🚨 Costos proyectados 3x+ del budget

📎 ANEXOS

A. Herramientas Recomendadas por Fase

```yaml

Static Analysis: Ruff, ESLint, SonarQube

Security: Bandit, Snyk, OWASP ZAP, Semgrep

Testing: Pytest, Jest, Playwright, Locust

LLM Eval: PromptFlow, LangSmith, Anthropic Console

Observability: Grafana, Prometheus, Jaeger, Datadog

```

B. Templates de Documentación

[Links a templates en GitHub/Notion]

C. Checklist por Rol

```markdown

## ## Tech Lead

- [ ] Arquitectura reviewed
- [ ] Performance targets met
- [ ] Technical debt < \_\_ items

## ## Security Lead

- [ ] Penetration test passed
- [ ] Compliance validated
- [ ] Secrets management audited

## ## Product Owner

- [ ] UX validation passed
- [ ] User docs complete
- [ ] Success criteria met

```

****VERSIÓN:** 2.0 | **ÚLTIMA ACTUALIZACIÓN:** 2025-Q4 | **MANTENEDOR:** [Tu equipo]**

Este framework es un living document. Contribuciones bienvenidas.