Plan: Procesamiento Asíncrono de Limpieza de PDFs

Contexto

Actualmente el subscriber llama a `clean_pdf_on_upload` de forma síncrona al subir PDFs, bloqueando al usuario. La función base debe mantenerse síncrona porque también se usa desde vistas.

Infraestructura actual:

- 20 máquinas virtuales
- 4 ZEO Clients por máquina (zc1, zc2, zc3, zc4) con 4 threads cada uno
- 25 sitios Plone por máquina (servidos por los 4 ZCs)
- **Total: 80 procesos Zope sirviendo 500 sitios Plone**

> ⚠️ **Situación RAM Crítica (medición real):**
>
> - RAM Total por máquina: 5.7GB
> - RAM Usada: 4.3GB (75% ocupado)
> - **RAM Disponible: 1.2GB** (buffer muy ajustado)
> - Zope consume: 3.6GB (zc1: 1GB, zc2: 908MB, zc3: 945MB, zc4: 784MB)
> - Memon reinicia instancias automáticamente cuando se queda sin RAM

Opciones Disponibles

Opción 1: Celery (Gold Standard)

> ⏱️ **Estimación: 40-60 horas**
>
> - **Desarrollo código tareas Celery: 8h**
> - **Setup y configuración Redis (1 por máquina): 16h**
> - **Configuración workers en buildout/supervisor: 12h**
> - **Testing y QA: 8h**
> - **Deployment en 20 máquinas: 8h**
> - **Documentación: 4h**

- **Monitoreo y ajustes post-deploy: 4-8h**

💰 **Recursos necesarios por máquina:**

- Redis: +500MB-1GB RAM
- Workers Celery (2-3): +400-900MB RAM
- **Total: +1-2GB RAM por máquina**
- CPU: +10-20% en picos

❌ **INVIABLE con RAM actual**

RAM resultante: 1.2GB - 1.5GB = **NEGATIVO** (no cabe)

Requiere ampliar VMs:

- Opción A: Ampliar cada VM de 5.7GB → 8GB mínimo (20 VMs × +2.3GB = **+46GB total**)
- Opción B: Crear 20 VMs nuevas dedicadas Redis (2GB cada una = **+40GB total**)
- Coste: **[CONSULTAR CON INFRAESTRUCTURA]**

✅ **Pros:**

- Solución estándar y robusta del ecosistema Python
- Sistema de reintentos automáticos configurable
- Monitoreo y observabilidad con Flower
- Escalable horizontalmente (múltiples workers)
- Integración con New Relic ya disponible
- Comunidad grande y muy buena documentación

⚠️ **Contras:**

- Requiere infraestructura adicional (Redis/RabbitMQ + workers)
- Mayor complejidad de deployment (20 Redis + supervisord)
- Más configuración inicial
- **Requiere inversión significativa en hardware**

Coste total Opción 1: 40-60h desarrollo + **[CONSULTAR coste hardware]**

Opción 2: collective.taskqueue2 (Específico Plone 6)

⏱️ **Estimación: 24-32 horas**

- **Investigación y pruebas collective.taskqueue2: 4h**
- **Desarrollo código tareas: 6h**
- **Configuración variables entorno (template para 500 instancias): 4h**
- **Testing en desarrollo: 4h**
- **Deployment y verificación en 500 instancias: 8h**
- **Documentación: 2h**
- **Ajustes y monitoreo post-deploy: 4h**

💰 **Recursos necesarios por máquina:**

- Huey workers: 75MB por ZC × 4 = **+300MB RAM total**
- CPU: <5% (I/O bound, espera respuesta API)
- Disco: SQLite ~50MB por ZC × 4 = **+200MB disco**

🟡 **JUSTO con RAM actual**

RAM resultante: 1.2GB - 0.3GB = **900MB disponibles**

Viable pero buffer reducido:

- Cabe en recursos actuales (0€ hardware obligatorio)
- Riesgo: Buffer estrecho podría aumentar reinicios memon
- Opcional: Ampliar +1-2GB por VM para mayor confort

⚠️ **IMPORTANTE:** Para Plone 6 debes usar `collective.taskqueue2` (no el original). Usa Huey como base.

✅ **Pros:**

- **Persiste tareas**: Sobrevive a reinicios de instancias (crítico con memon)
- **Compatible Plone 6**: Mantenido y diseñado para Plone 6 + Python 3
- **Múltiples backends disponibles**:
  - `sqlite://` - Archivo SQLite local (sin infraestructura extra)
  - `file://` - Sistema de archivos
  - `redis://` - Redis (si lo tienes)
- Integración nativa con Zope/Plone
- ZEO-aware: Diseñado para múltiples instancias Zope

- Reintentos automáticos: Manejo de errores built-in

⚠️ **Contras:**

- **Proyecto pequeño**: Solo 3 stars en GitHub, menos maduro que Celery
- Desarrollado por un solo equipo: Universidad de Bologna/Andreas Jung
- Configuración por instancia: Cada instancia necesita variables de entorno
- Menos documentación/comunidad que Celery

Coste total Opción 2: 24-32h desarrollo + **0€ hardware obligatorio** (opcional: ampliar VMs)

Opción 3: Threading + afterCommitHook (Solución ligera)

⏱️ **Estimación: 12-16 horas**

- **Desarrollo código threading + afterCommitHook: 4h**
- **Implementación lógica reintentos: 2h**
- **Testing en desarrollo: 2h**
- **Deployment (solo actualizar código): 2h**
- **Documentación: 1h**
- **Monitoreo y ajustes: 2-4h**

💰 **Recursos necesarios por máquina:**

- Threads background: 35MB por ZC × 4 = **+140MB RAM total**
- CPU: <5% (I/O bound)
- Threads: Dinámicos (solo cuando hay uploads)
- Disco: 0

✅ **SEGURO con RAM actual**

RAM resultante: 1.2GB - 0.14GB = **1.06GB disponibles**

Impacto mínimo en RAM crítica:

- Mantiene buffer razonable para memon
- No aumenta presión sobre memoria
- Sin necesidad de ampliar hardware

✅ **Pros:**

- Sin dependencias externas
- Implementación rápida
- No requiere infraestructura
- Deployment ultra-simple (solo código)
- **0€ en hardware**

⚠️ **Contras:**

- **Sin persistencia de tareas**: Si memon reinicia durante limpieza, se pierde tarea
- Sin sistema de reintentos robusto (manual)
- Difícil de monitorear (solo logs)
- Crítico con reinicios memon frecuentes

Mitigación: Añadir cronjob mensual batch (+8h) para limpiar PDFs que se perdieron

Coste total Opción 3: 12-16h desarrollo + **0€ hardware**

Opción 4: Guardar sin limpiar + Cronjob procesador

⏱️ **Estimación: 16-20 horas**

- **Modificar subscriber (marcar como pendiente): 3h**
- **Crear campo/annotation para estado: 2h**
- **Desarrollar script procesador batch: 4h**
- **Configurar cronjobs en 20 máquinas: 4h**
- **Testing: 3h**
- **Documentación: 2h**
- **Ajustes: 2h**

💰 **Recursos necesarios:** Picos temporales de RAM (~200MB) durante ejecución batch. Sin incremento permanente.

Coste total Opción 4: 16-20h desarrollo + **0€ hardware**

Comparación Crítica con Infraestructura Real

Aspecto	Opción 1 (Celery)	Opción 2 (taskqueue2)	Opción 3 (Threading)
Estimación horas	40-60h	24-32h	12-16h
RAM adicional/máquina	+1-2GB	+300MB	+140MB
RAM resultante disponible	❌ NEGATIVO	🟡 900MB	✅ 1.06GB
Viable con RAM actual	❌ NO	🟡 Justo	✅ SÍ
Coste hardware	⚠️ [CONSULTAR]	0€	0€
Persiste tareas	✅ SÍ	✅ SÍ	❌ NO
Sobrevive reinicios memon	✅ SÍ	✅ SÍ	❌ NO
Complejidad deployment	Alta	Media	Baja
Madurez tecnológica	✅ Alta	🟡 Media	✅ Alta

¿Qué pasa si se pierde una tarea?

- **Con Threading**: PDF queda guardado SIN limpiar (metadatos presentes)
- **Con taskqueue2**: Tarea se retoma al reiniciar, PDF se limpia eventualmente
- **Con Celery**: Tarea se retoma, máxima garantía de limpieza

Comparativa Total Recursos (20 máquinas)

Opción	RAM total adicional	Coste desarrollo	Coste hardware	Requiere aprobación presupuesto
1. Celery	+46GB (ampliar) o +40GB (nuevas VMs)	40-60h	**[ALTO - Consultar]**	⚠️ SÍ
2. taskqueue2	+6GB (300MB × 20)	24-32h	**0€**	❌ NO
3. Threading	+2.8GB (140MB × 20)	12-16h	**0€**	❌ NO
4. Cronjob	0 (picos temporales)	16-20h	**0€**	❌ NO

Recomendación DEFINITIVA

Escenario 1: SIN presupuesto para ampliar hardware

→ Opción 3 (Threading)

- ✅ Única opción 100% segura con RAM actual (1.06GB disponibles)
- ✅ 0€ en hardware
- ✅ 12-16h desarrollo (más rápido)
- ✅ No agrava problema de memon
- ⚠️ Sin persistencia: Si memon reinicia durante limpieza, se pierde tarea (pero PDF ya está guardado)

Mitigación: Añadir cronjob mensual batch (+8h) para limpiar PDFs perdidos → **Total: 20-24h**

Escenario 2: CON presupuesto limitado + reinicios memon >1/semana

→ Opción 2 (collective.taskqueue2)

- 🟡 Cabe justo en RAM actual (900MB disponibles)
- ✅ Persiste tareas (crítico si reinicios frecuentes)
- ✅ 0€ hardware obligatorio

- 🟡 Opcional: Ampliar VMs +1-2GB para mayor confort
- 24-32h desarrollo

Escenario 3: CON presupuesto amplio + quieren gold standard

→ Opción 1 (Celery)

- ⚠️ Requiere ampliar VMs a 8GB o crear 20 VMs Redis
- ✅ Máxima robustez y observabilidad
- ✅ Persiste tareas
- 40-60h desarrollo
- Coste: **[CONSULTAR con infraestructura]**

Próximos Pasos para Tomar Decisión

1. **Confirmar con infraestructura:**
   - ¿Es posible ampliar RAM VMs de 5.7GB → 8GB?
   - ¿Cuál sería el coste?
   - ¿Cuánto tardaría el proceso?
2. **Confirmar con equipo Plone:**
   - ¿Con qué frecuencia reinicia memon actualmente?
   - ¿Qué % de PDFs debe limpiarse obligatoriamente?
3. **Evaluar presupuesto disponible:**
   - ¿Hay presupuesto para ampliar hardware si fuera necesario?
   - ¿Cuál es el presupuesto máximo para esta mejora?

Plan generado: 12 de Febrero de 2026

Para dudas o aclaraciones sobre este plan, contacta con el equipo de desarrollo