# EcoWatch – Documentación Técnica del Sistema de Monitoreo Ambiental

**Autor**:
Ezequiel Ferrario

EcoWatch es un sistema modular de ingeniería de datos creado para recolectar, validar, procesar y reportar información ambiental proveniente de sensores en diferentes salas. El sistema busca facilitar la toma de decisiones rápidas y precisas, asegurando datos de calidad y generando reportes ejecutivos adaptables y fácilmente extensibles.

# Arquitectura

## Estructura general de carpetas

```
src/
  main.py
  log_reader.py
  cache.py
  domain/
    log.py
    sensor.py
    sala.py
    reporte.py
  reportes/
    base.py
    factory.py
    estado_por_sala.py
    alertas.py
    temp_extremas.py
  utils/
    decorators.py
data/
  logs_ambientales_ecowatch.csv
tests/
  test_log.py
requirements.txt
```

## Flujo general del trabajo

1. **Ingesta de logs** desde archivo CSV.
2. **Validación y parseo** de registros.

3. **Procesamiento modular y OO:** Se crean objetos para cada entidad (log, sensor, sala).

4. **Caché temporal**: Registros recientes en memoria para consultas rápidas.

5. **Generación de reportes** ejecutivos usando patrones de diseño.

6. **Exportación** y visualización de resultados.
7.

# Estructuras de datos

La base del sistema utiliza **listas** y **diccionarios** para manipular los logs y facilitar filtros rápidos por sala y timestamp. Se emplean **clases orientadas a objetos** (`Log`, `Sensor`, `Sala`) que encapsulan atributos y comportamientos del dominio, favoreciendo la extensión y el mantenimiento.

En la caché temporal, el uso de diccionarios indexados permite consultas y eliminación eficiente de registros antiguos. Además, **pandas DataFrame** se incorpora solo en los reportes, para análisis tabulares y exportación a CSV/XLSX, integrando así lo mejor del procesamiento ligero y las herramientas profesionales de análisis.

Esta elección permite lograr velocidad en la lógica real-time y flexibilidad/extensibilidad a futuro. Se descartaron estructuras como DataFrame en el pipeline principal para evitar overhead de memoria y se postergó el uso de bases de datos para una futura versión.

# Validación y aseguramiento de la calidad de datos

Para garantizar la integridad, el sistema valida que cada log contenga los campos esperados y que sus valores estén dentro de rangos físicos realistas (ej: temperatura, humedad, $CO_2$).
Se implementan setters y validaciones en los objetos de dominio para prevenir la entrada de datos corruptos o maliciosos.
Además, se utilizan pruebas unitarias automáticas para asegurar la robustez de las validaciones y la lógica central.

# Patrones de diseño implementados

## Factory

El Factory centraliza la creación de reportes. Permite instanciar diferentes tipos de reportes según el requerimiento, asegurando que se puedan agregar nuevas funcionalidades (como nuevos tipos de reportes) sin modificar código existente. Esto cumple el principio Open/Closed de SOLID.

## Strategy

Cada tipo de reporte encapsula su lógica específica. Así, agregar un reporte nuevo (por ejemplo, de alertas, extremos, etc.) solo implica crear una clase nueva y registrarla en el Factory, sin cambiar lo demás.

## Decorator

Se implementa un decorador personalizado para loguear la ejecución de los métodos de reporte, lo cual facilita la depuración y la auditoría, y sienta las bases para incluir benchmarking de tiempos o validaciones extra en el futuro.

# Tecnicas de optimizacion

**Caché en memoria** con ventana de 5 minutos: Mejora la velocidad de acceso y soporta eventos fuera de orden (late events).

**Eliminación automática** de registros antiguos: Evita el uso excesivo de memoria.

**Parseo eficiente:** Solo los registros válidos se procesan.

**Pandas en reportes:** Permite análisis avanzado y exportación directa a Excel/CSV de forma profesional.

**Exportación automática:** Facilita la entrega y el uso de los reportes por otros equipos.

## Consideraciones

Desde el inicio, el sistema fue diseñado para ser **modular y legible**. Cada clase está documentada, y la lógica está separada por responsabilidad (lectura, validación, procesamiento, reportes, etc.).

La simplicidad fue priorizada para facilitar el onboarding de nuevos desarrolladores, mientras que la eficiencia se logra con estructuras ligeras y consultas optimizadas.
La extensibilidad es inmediata: agregar sensores, tipos de reportes, validaciones o nuevas fuentes no requiere refactorizar lo ya probado.

# Ventajas

**Ventajas:**

- **Flexibilidad:** Agregar nuevas funcionalidades es sencillo gracias a la modularidad y los patrones de diseño aplicados.
- **Escalabilidad:** El sistema puede crecer en volumen de datos y en complejidad sin perder eficiencia.
- **Mantenibilidad:** El bajo acoplamiento permite que cualquier programador comprenda y modifique el sistema rápidamente.
- **Integración profesional:** El uso de pandas y exportación a CSV/XLSX permite que los resultados se integren fácil en herramientas de BI o análisis externo.

**Comparativas:**

- Un enfoque monolítico o sin OO dificultaría la evolución del sistema.
- Usar pandas como core hubiera sido ineficiente para flujos en tiempo real, pero sí es la mejor opción para la etapa de reporte.

# Evolucion

El diseño modular y desacoplado garantiza que EcoWatch puede incorporar fácilmente:

- Nuevas fuentes de datos o sensores (agregando clases lectoras).
- Nuevos tipos de reportes (nueva clase + línea en Factory).
- Integración con bases de datos, APIs externas, o visualizaciones avanzadas.

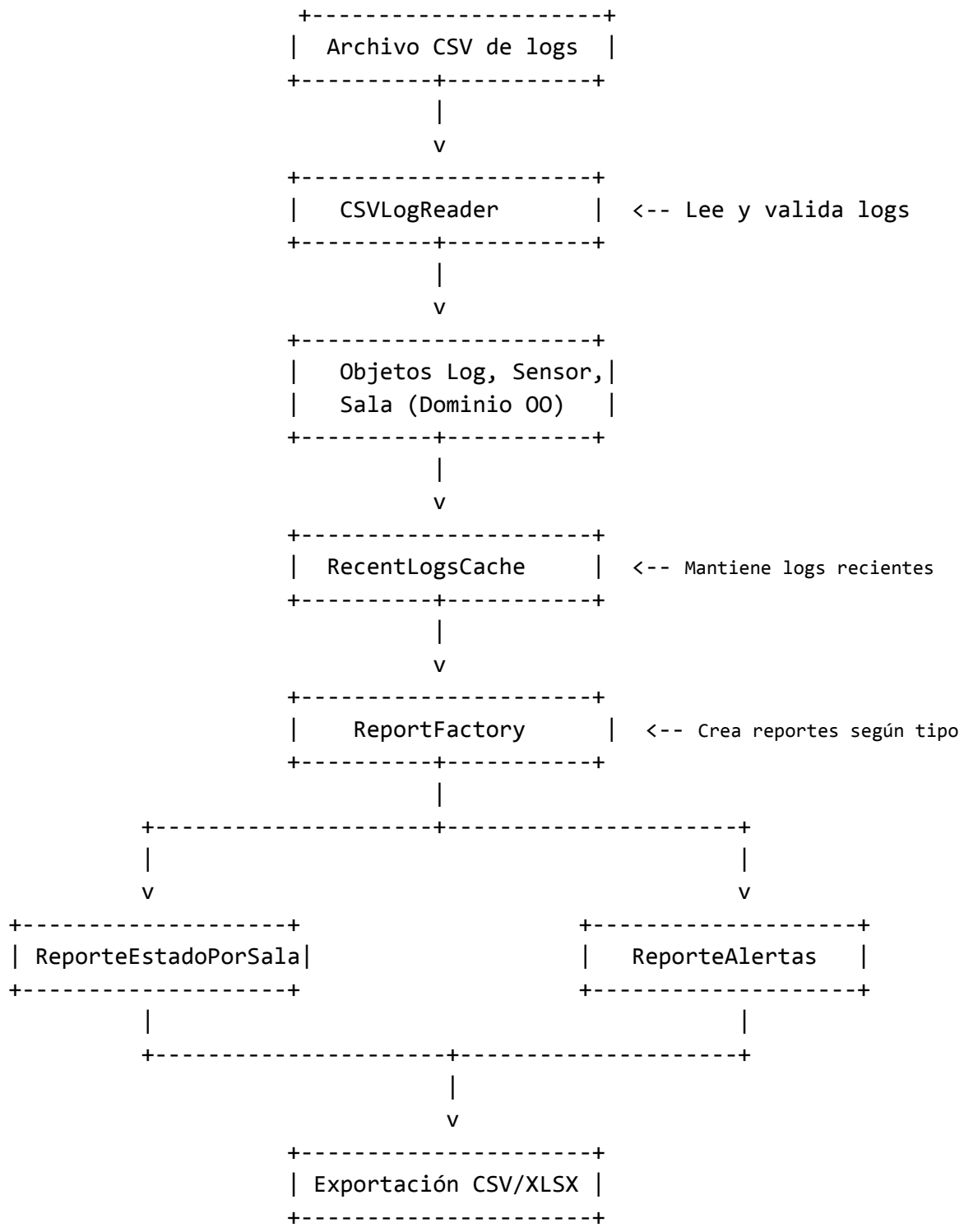
# Test

Se implementaron pruebas unitarias con `pytest` para validar el correcto funcionamiento de validaciones críticas.
Los decoradores permiten extender la trazabilidad y medición de performance de funciones clave.

# Conclusiones

EcoWatch presenta un diseño sólido, escalable y preparado para evolucionar con las necesidades de la empresa.
Entre las futuras mejoras posibles se incluyen:

- Integración con bases de datos relacionales/noSQL.
- Dashboards visuales y alertas en tiempo real.
- Reportes automáticos programados y notificaciones por email o chat.


Archivo CSV de logs
CSVLogReader
<-- Lee y valida logs
Objetos Log, Sensor,
Sala (Dominio OO)
RecentLogsCache
<-- Mantiene logs recientes
ReportFactory
<-- Crea reportes según tipo
ReporteEstadoPorSala
ReporteAlertas
Exportación CSV/XLSX