Departamento de fisica
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
EL6046-1 Mecanica cuantica

# Reconstrucción Examen

Estudiante: Simón Villavicencio
Profesor: Luis Foa.

Fecha: 19 de febrero de 2026
Santiago, Chile

# Índice de Contenidos

# 1. Entendimiento del Flujo de Proceso ("La Historia de la Cal")

Se ha realizado un análisis detallado del diagrama de flujo de proceso (PFD) ‘flujo_mineria.pdf‘ y se ha correlacionado con la información existente en el código (‘cal_monitoring_backend/‘) y las referencias a archivos previos (‘Antiguo/‘). Este análisis nos permite establecer una narrativa clara del viaje de la cal a través de la planta, fundamental para la creación de escenarios de prueba realistas.

## 1.1. Etapas del Proceso

El proceso de monitoreo de la lechada de cal se descompone en las siguientes etapas principales:

1. **Almacenamiento de Cal Viva:**
   - *Historia:* La cal viva (materia prima) se recibe y almacena en un gran silo.
   - *Relación con el Proyecto:* El PFD muestra el "Silo de Cal". En el código, los sensores ‘2270-LIT-11825‘ (nivel) y las alarmas asociadas (‘2270-LSHH-11826‘, ‘2270-LSLL-11829‘) se refieren directamente a esta etapa.
2. **Dosificación y Alimentación:**
   - *Historia:* La cal viva se extrae del silo y se alimenta de manera controlada al equipo de hidratación.
   - *Relación con el Proyecto:* El control se realiza mediante el .ªlimentador de Tornillo"(‘2270-SAL-11817‘) y la "Válvula Rotatoria"(‘2270-SAL-11818‘), cuyas señales son utilizadas en ‘data_generator.py‘ para simular la alimentación de cal.
3. **Hidratación (El .ªpagado"de la Cal):**
   - *Historia:* La cal viva reacciona exotérmicamente con agua en un "Slaker.º .ªpagador"para formar hidróxido de calcio (lechada de cal). Esta es la etapa central del proceso.
   - *Relación con el Proyecto:* El PFD muestra el equipo de mezcla. Los sensores ‘2270-FIT-11801‘ (flujo de agua) y ‘2270-TT-11824A/B‘ (temperaturas del slaker) son críticos aquí. La clase ‘ReactivityMonitor‘ en ‘core_logic.py‘ analiza la curva de temperatura del sensor ‘2270-TT-11824B‘, implementando directamente el concepto de *reactividad* explicado en los libros de Guillermo.
4. **Separación y Clasificación:**
   - *Historia:* La lechada de cal se procesa para remover impurezas y partículas gruesas (.ªrenilla.º "grit") en una cámara de separación o hidrociclones.
   - *Relación con el Proyecto:* El PFD muestra los hidrociclones. Los sensores ‘2270-LIT-11850‘ (nivel de la cámara de separación) y alarmas como ‘2270-PALL-11834‘ (presión del hidrociclón) en ‘alarm_config.json‘ confirman la relevancia de esta etapa para el monitoreo.

5. **Almacenamiento y Distribución de Lechada Final:**
    - *Historia:* La lechada de cal lista para su uso se almacena en tanques y se bombea a los puntos de aplicación en la minera.
    - *Relación con el Proyecto:* El PFD finaliza con tanques de lechada y bombas. Sensores como '2270-LIT-11845' (nivel de tanque de descarga) y variables de monitoreo de bombas y sistemas de lubricación ('2270-PIT-11895', '2270-TIT-11893', '2270-FSL-11896') están asociados a esta etapa.

# 2. Relación Clave con los Libros de Guillermo y el Diseño del Software

La comprensión profunda de los libros de Guillermo Coloma Álvarez es **esencial** para desarrollar un software que no solo monitoree, sino que interprete y optimice el proceso de la cal de manera inteligente. Los libros no son solo teoría; proporcionan el marco conceptual para validar y enriquecer nuestro modelo de simulación y lógica de control.

- **La Reactividad de la Cal como Fundamento:** Los libros de Guillermo enfatizan que la *reactividad de la cal* (definida por la velocidad y magnitud del aumento de temperatura durante el apagado) es una propiedad crítica que determina su eficiencia.
    - *Conexión con Datos:* Nuestro sensor '2270-TT-11824B' (temperatura del slaker) mide directamente esta reacción. Un simulador avanzado debe poder generar curvas de temperatura que varíen significativamente según la *calidad de cal simulada* (ej. una cal "dura de apagar.º de baja reactividad mostrará una curva más lenta y menos intensa, como se describe en los Gráficos de Reactividad de los libros).
    - *Impacto en el Software:* La clase 'ReactivityMonitor' de 'core_logic.py' está perfectamente posicionada para clasificar estas curvas, pero su eficacia depende de que el simulador le proporcione datos que reflejen las distintas calidades de cal.
- **La Calidad de la Cal y el ÇaO Equivalente":** Los libros distinguen entre 'CaO libre', 'CaO crudo' y 'CaO combinado/requemado', todos contribuyentes al 'CaO Equivalente' y a la capacidad alcalinizante total. La presencia de impurezas impacta directamente en estas proporciones.
    - *Conexión con Datos:* Actualmente, nuestro simulador genera solo valores de sensores. Para reflejar la riqueza de los libros, el simulador debe permitir definir el **perfil de calidad de la cal de entrada** (ej. 90 % CaO libre, 5 % impurezas). Estos parámetros, aunque no son directamente sensores, deben influir en el comportamiento de los sensores simulados (ej. un mayor porcentaje de 'CaO crudo' podría generar lecturas de temperatura de apagado anómalas o más lentas).
    - *Impacto en el Software:* 'core_logic.py' podría entonces calcular el ÇaO Equivalente"simulado, y este valor, junto con la reactividad, sería un indicador clave de rendimiento de la cal procesada, directamente extraído de la teoría de Guillermo.

- **Impurezas, Agua y Eventos Anormales:** Los libros detallan cómo las impurezas de la caliza o del agua, así como la temperatura o la dosificación incorrecta de agua, pueden llevar a problemas como la formación de .ªrenillas", incrustaciones, menor reactividad o consumo excesivo de energía.

    - *Conexión con Datos:* Nuestros escenarios de prueba deben contemplar estas situaciones. Por ejemplo, simular un .exceso de impurezas en el agua"podría generar datos de sensores que gradualmente lleven a una alarma de ïncrustación"(si desarrollamos una lógica para ello), o un escenario de çal de baja calidadresultaría en una reactividad pobre que el software debería detectar.
    - *Impacto en el Software:* Esto nos guiará para crear lógica de alarmas y monitoreo más sofisticada en ‘core_logic.py‘ que considere estas interacciones complejas, tal como se detalla en los diagramas de control de las plantas de lechada de los libros (ej. el diagrama de calidad de la lechada en la página 117 de ÇaO más alto implica ahorro...").

Esta integración de la teoría de Guillermo con los datos simulados permitirá que nuestro software no solo sea funcional, sino que también actúe como una herramienta de aprendizaje y optimización basada en un conocimiento profundo del proceso de la cal.

# 3. Avances de la Sesión Actual (17 de Febrero de 2026)

Durante esta sesión, se han logrado los siguientes avances significativos:

1. **Recopilación de Contexto Inicial:** Se realizó una revisión exhaustiva de todos los archivos del proyecto, incluyendo ‘resumen_progreso.txt‘, las bitácoras anteriores y el código base en ‘cal_monitoring_backend/‘.

2. **Análisis Profundo de la Documentación Clave:** Se leyeron y analizaron los libros de Guillermo Coloma Álvarez, "La Cal ¡Es un Reactivo Químico!ƶ ÇaO más alto implica ahorro de energía y agua", extrayendo insights críticos sobre la química de la cal, su reactividad, impurezas y la optimización del proceso. Este análisis ha sido fundamental para comprender la "Historia de la Calƶ las bases conceptuales del proyecto.

3. **Diagnóstico y Solución del Problema de Ejecución de la API:** Se identificó la causa del problema reportado (ïmagen en blanco.en el navegador) como una ejecución incorrecta de la aplicación FastAPI. Se proporcionaron instrucciones claras sobre cómo ejecutarla correctamente usando ‘uvicorn cal_monitoring_backend.main:app –reload‘ desde la raíz del proyecto, asegurando que las importaciones relativas funcionen como se espera.

4. **Definición Detallada de la "Historia de la Cal":** Se desglosó el flujo de proceso de la planta de cal en 5 etapas principales (Almacenamiento de Cal Viva, Dosificación y Alimentación, Hidratación, Separación y Clasificación, Almacenamiento y Distribución de Lechada Final). Para cada etapa, se identificaron los equipos clave del PFD (‘flujo_mineria.pdf‘), los TAGs de sensores relevantes del código existente (‘alarm_config.json‘, ‘data_generator.py‘) y su propósito funcional, creando una narrativa coherente del proceso.

5. **Relación Crítica de los Libros con el Diseño del Software:** Se estableció y documentó explícitamente cómo los conceptos de los libros de Guillermo (como la reactividad de la cal, el CaO equivalente, el impacto de las impurezas y la calidad del agua) deben ser integrados en el diseño del software, particularmente en la generación de datos de prueba y la lógica de negocio. Se enfatizó que estos conceptos son cruciales para crear un modelo de simulación realista y una lógica de monitoreo inteligente.

6. **Generación de Datos de Prueba para la Etapa 1 (Almacenamiento de Cal Viva):**

   - Se definieron 5 micro-escenarios específicos para la simulación del nivel del silo y sus alarmas asociadas (‘2270-LIT-11825‘, ‘2270-LSHH-11826‘, ‘2270-LSLL-11829‘): nivel estable, vaciándose, llenándose, alarma de nivel alto y alarma de nivel bajo.
   - Se proporcionó un prompt detallado para un asistente de código IA para generar un script de Python (‘scenario_generator.py‘) que produce estos 5 escenarios en archivos CSV individuales.
   - Se confirmó la exitosa generación del script ‘scenario_generator.py‘ y los cinco archivos CSV correspondientes dentro de la carpeta ‘cal_monitoring_backend/‘.

Los avances realizados en esta sesión han sentado una base sólida para el desarrollo futuro, asegurando que el software se construya con una comprensión profunda del proceso y una alineación directa con los principios fundamentales establecidos en la documentación del proyecto y los libros de referencia.

# 1. Introducción y Objetivos

El presente informe detalla el progreso, los análisis realizados y los desarrollos implementados en el proyecto de software para el **monitoreo del flujo y proceso de la cal** en una operación minera. El objetivo principal es construir un sistema de software robusto que permita la supervisión integral y el análisis operacional del sistema de preparación y distribución de lechada de cal. Esto se alinea con los requisitos de la **Etapa 2 (Supervisión Integral)** y sienta las bases para la futura **Etapa 3 (Control Avanzado y Optimización)**.

# 2. Análisis de Documentación Fundamental

Se realizó un análisis exhaustivo de la documentación técnica proporcionada, extrayendo el contexto esencial para el diseño y desarrollo del software.

## 2.1. La Cal como Reactivo Químico (Libro de G. Coloma)

Este texto proporcionó la base teórica sobre la química de la cal, destacando la importancia crítica de la **reactividad**, la hidratación (apagado) y las variables de proceso (temperatura, calidad del agua, etc.) que afectan la calidad final de la lechada.

## 2.2. Especificación de Instrumentación y Software

Este documento es la hoja de ruta del proyecto. Define las **Etapas 2 y 3**, detallando los requisitos funcionales como dashboards en tiempo real, registro histórico, sistema de alarmas y la necesidad de monitorear la instrumentación existente (LT, FT, pHT, DT).

## 2.3. Filosofía de Control

Proporciona el "cómo" se controla la planta actualmente. Se extrajeron tablas de **alarmas e interlocks**, secuencias operacionales, y los **TAGs** específicos de la instrumentación del Sistema de Control Distribuido (DCS), que son la base para nuestra configuración de alarmas.

## 2.4. Diagrama de Flujo de Proceso (PFD)

El PFD ('flujo_mineria.pdf') ofreció una representación visual completa de la planta, permitiendo unir los conceptos teóricos y los requisitos de software en una vista arquitectónica unificada del proceso físico.

# 3. Estructura y Lógica del Software Actual

El núcleo del proyecto reside en la carpeta 'cal_monitoring_backend/', que contiene una aplicación Python moderna basada en FastAPI.

- **‘main.py‘**: Es el punto de entrada de la **API REST**. Está diseñado para exponer los datos y la lógica del sistema a través de endpoints, como ‘/api/v1/status‘, permitiendo que una futura interfaz de usuario (frontend) consuma la información en tiempo real.
- **‘core_logic.py‘**: Es el cerebro de la aplicación. Contiene la lógica de negocio para:
  - Cargar configuraciones y datos.
  - **Evaluar un sistema de alarmas complejo** basado en condiciones absolutas, relativas a setpoints y lógicas (‘multiple_and‘).
  - Determinar el **modo de operación** de la planta (ej. “produciendo”, “lavando”, “inactivo”).
  - La clase **‘ReactivityMonitor‘**, una implementación clave inspirada en los libros de Guillermo Coloma, diseñada para detectar y clasificar las curvas de reactividad de la cal basándose en la evolución de la temperatura.
- **‘config/alarm_config.json‘**: Archivo de configuración que **externaliza toda la lógica de alarmas**. Contiene los TAGs de los sensores, los equipos asociados, los umbrales y las descripciones de cada condición de alarma, replicando la filosofía de control del DCS.
- **‘data_generator.py‘ y ‘scenario_generator.py‘**: Scripts de Python dedicados a la **simulación de datos de sensores**. Permiten generar escenarios realistas y dinámicos para probar la lógica del backend y facilitar el desarrollo del frontend sin depender de datos de una planta real.

# 4. Avances Significativos y Decisiones Clave

## 4.1. Definición de la “Historia de la Cal”

Se ha establecido una narrativa clara y detallada del proceso de la cal, correlacionando el PFD con los conceptos de los libros de Guillermo Coloma y los TAGs de los sensores del proyecto. Este entendimiento es fundamental para crear simulaciones y lógicas de monitoreo que reflejen fielmente la realidad operativa. El proceso se ha dividido en 5 etapas:

1. Almacenamiento de Cal Viva (Silo de Cal).
2. Dosificación y Alimentación (Alimentador de Tornillo).
3. Hidratación o “Apagado” (Slaker).
4. Separación y Clasificación (Hidrociclones).
5. Almacenamiento y Distribución de Lechada Final.

## 4.2. Integración de Conocimiento Experto

Se ha documentado explícitamente cómo los conceptos de los libros de Guillermo Coloma (reactividad, CaO equivalente, impurezas) deben guiar el desarrollo. El software no solo debe monitorear, sino **interpretar el proceso** a la luz de esta teoría, permitiendo una optimización futura.

### 4.3. Desarrollo de un Simulador de Escenarios

Reconociendo la necesidad de datos dinámicos para el desarrollo y las pruebas, se ha priorizado y completado la creación de un generador de escenarios.

- Se creó el script **'scenario_generator.py'**.
- Este script genera archivos **CSV** con datos que simulan 5 micro-escenarios para el nivel del silo de cal:
  1. Nivel estable.
  2. Vaciándose (en producción).
  3. Llenándose (recargando).
  4. Alarma de nivel alto.
  5. Alarma de nivel bajo.
- Estos escenarios son la primera implementación del generador de datos y sirven como base para futuras simulaciones más complejas.

### 4.4. Aclaración de Requisitos Futuros

Se ha establecido que la arquitectura a mediano plazo deberá migrar hacia el uso de servicios en la nube de **Amazon Web Services (AWS)** para la persistencia y gestión de datos (ej. S3, RDS, Kinesis), reemplazando las soluciones temporales basadas en archivos.

## 5. Próximos Pasos

El plan de acción inmediato se centra en expandir la funcionalidad de la API y la simulación:

1. **Expandir la API ('main.py'):** Desarrollar y refinar los endpoints de la API para que consuman los datos generados por el simulador y expongan de manera clara el estado de la planta, las alarmas activas y las curvas de reactividad.
2. **Integrar Escenarios Generados:** Modificar el simulador principal ('data_generator.py') para que pueda leer y utilizar los archivos CSV generados por 'scenario_generator.py', permitiendo ejecutar pruebas controladas desde la API.
3. **Ampliar el Generador de Escenarios:** Crear nuevos escenarios de simulación que cubran otras etapas del proceso, como las curvas de reactividad (alta, media, baja) y condiciones de alarma en el Slaker.
4. **Iniciar Desarrollo de Interfaz de Usuario (Frontend):** Con una API funcional que provee datos dinámicos, se puede comenzar el desarrollo de una página web que visualice en tiempo real el estado de la planta, las alarmas y los gráficos de tendencia.