Mapa de las dependencias de los módulos del proyecto, revelando una arquitectura de aplicación de tipo **multi-capa** o **MVC (Modelo-Vista-Controlador)**, característica de sistemas de gestión empresarial como un **MES/ERP** (Sistema de Ejecución de Manufactura/Planificación de Recursos Empresariales).

📚 Documentación - Arquitectura y Flujo del Código Fuente

I. 🏗️ Arquitectura de Capas y Componentes Principales

La aplicación se organiza en una arquitectura bien definida con una clara separación de responsabilidades, lo que promueve la modularidad, la mantenibilidad y la escalabilidad.

1. Las Tres Capas Fundamentales

Capa	Carpeta/Prefijo	Responsabilidad Principal	Interacciones Clave
Controladores	controladores/	Manejo de las **solicitudes HTTP**, enrutamiento y gestión de la respuesta al cliente.	**Llama** a métodos en la capa de **Servicios**.
Servicios	servicios/	Contiene la **lógica de negocio** de la aplicación y la coordinación	**Llama** a funciones en la capa de **Servicios de**

		transaccional.	**Infraestructura**.
Servicios de Infraestructura	servicios_infraestructura/	Proporciona **utilidades de bajo nivel**, acceso a la base de datos, seguridad, y manejo de archivos.	Utiliza librerías de terceros (ej., pandas, mariadb). No depende de capas superiores.

2. Capas Auxiliares

- **Middlewares (middlewares/):** Lógica ejecutada en el **flujo de la solicitud** (antes o después del controlador). Se usa para **autenticación** (middleware2_usuario_req.py con jwt), **autorización** (middleware3_permiso.py) y pre/post-procesamiento de datos.
- **Aplicación (app.py):** El punto de entrada que inicializa el framework **Flask** y ensambla los componentes (controladores, middlewares, servicios).

3. Librerías de Terceros Críticas

El proyecto utiliza una rica colección de bibliotecas para funciones específicas de un sistema industrial:

Área Funcional	Librerías de Python	Módulos Clave
Persistencia/Web	flask, mariadb, flask_cors	app.py, servicios_infraestructura/mariadb.py
Seguridad	jwt, bcrypt, base64	servicios_infraestructura/token_servicios.py, servicios_infraestructura/hash_servicios.py
Archivos/Reportes	pandas, openpyxl,	servicios_infraestructura/ex

	reportlab	cel_servicios.py, servicios_infraestructura/pdf_servicios.py
Optimización/Gráficos	pyomo, networkx	controladores/produccion/planificador_controlador.py
Integración	suds, webdav4, requests	servicios_infraestructura/libertya.py, servicios_infraestructura/webdav_servicios.py

---

II. □ Flujo de Interacción por Área Funcional

Las interacciones del código siguen un patrón estricto de **Controlador $\rightarrow$ Servicio $\rightarrow$ Infraestructura** para todas las áreas funcionales.

1. Área de Producción (produccion/ y pisodeplanta/)

Esta área gestiona el ciclo de vida de la orden de producción (OP) y la ejecución en planta.

Fase	Módulos de Servicio Clave	Interacción Detallada
Planificación	lanzamientodeop_servicios.py, analiticacapacidad_servicios.py	El controlador recibe la solicitud de planificación. Los servicios de Producción usan servicios_infraestructura para consultar fichas técnicas y el setup de máquinas. El planificador_controlador.py utiliza las librerías **pyomo** y

		networkx para modelos de optimización avanzados.
Ejecución (Piso de Planta)	partediario_servicios.py, valesdeconsumo_servicios.py	Se registra el avance de las OPs (Parte Diario) y el consumo de materiales. valesdeconsumo_servicios.py actualiza el inventario vía servicios/comunes e invoca a la capa de **Sincronizadores** para notificar al sistema ERP externo (servicios/sincronizadores/salientes/consumos_sinc.py).

2. Área de Costos (costos/)

Responsable de la contabilidad, la valoración de inventarios y el cálculo del costo de producción.

Flujo de Costos	**Módulos de Servicio Clave**	**Interacción Detallada**
Contabilización	contabilizacion/*, diario_servicios.py	Los módulos de contabilizacion (consumo_materias_primas.py, recepciones_mercaderias.py) son disparados por eventos de Inventario/Piso de Planta. Estos servicios utilizan intensivamente las funciones de **diario_servicios_helper.py** y servicios_infraestructura/m

		ariadb.py para generar y registrar los asientos.
Costeo	costeo/productos_fabricados_costos.py	Este módulo calcula el **costo unitario** real o estándar. Depende de servicios_infraestructura para obtener historiales de tarifas (tarifashistorial_servicios.py) y movimientos de inventario por fecha (datetime).

3. Área de Inventario (inventario/)

Gestiona el control físico y lógico de los materiales y productos terminados.

Transacción	Módulos de Servicio Clave	Interacción Detallada
Recepción	recepcionmercaderia_servicios.py, consultaderecepciones_servicios.py	El servicio consulta al área de **Comunes** (servicios/comunes/oc_servicios.py) para obtener datos de la Orden de Compra (OC). Luego, coordina con el área de **Calidad** (servicios/calidad/ingresoadeposito_servicios.py) para el control de ingreso a depósito o inspección.
Movimientos	stock_servicios.py, stockdetalle_servicios.py	Son los módulos atómicos que interactúan directamente con la base de datos a través de

		servicios_infraestructura para leer y actualizar los niveles de stock.
Despacho	despachodefabricados_servicios.py	Llama a los servicios de **Sincronización** (servicios/sincronizadores/salientes/remitos_sinc.py) para notificar la salida de producto terminado al ERP o sistema de facturación.

4. Área de Calidad (calidad/)

Controla las especificaciones técnicas, las inspecciones y la documentación de calidad.

Flujo de Calidad	**Módulos de Servicio Clave**	**Interacción Detallada**
Control de Materia Prima	recepciontecnica_servicios.py, mantenimientocoladas_servicios.py	Los datos de lote (colada) y barrado son gestionados por servicios específicos que usan json para el manejo de estructuras de datos internas y servicios_infraestructura para la persistencia.
Documentación	gestiondecertificados_servicios.py, gestionexternadecertificados_controlador.py	Este flujo es de alta complejidad. El controlador de gestión externa utiliza servicios_infraestructura/certificado_de_calidad.py, el cual implementa **openpyxl** para la manipulación de plantillas Excel y **subprocess** para la

		generación o validación de documentos a través de procesos externos del sistema operativo.
Ensayos	gestiondeprobetas_servicios.py, _resultadosdeensayos_servicios.py	Lógica para la trazabilidad de las probetas de ensayo, gestión de resultados, y vinculación de estos con los certificados.

5. Área de Administración (administracion/)

Aloja la configuración y la gestión del maestro de datos y seguridad.

Maestro de Datos/Función	Módulos de Servicio Clave	Interacción Detallada
Seguridad	usuarios_servicios.py, roles_servicios.py	login_servicios.py inicia la autenticación. Utiliza servicios_infraestructura/hash_servicios.py (bcrypt) para validar la contraseña y servicios_infraestructura/token_servicios.py (jwt) para generar el token de sesión.
Configuración	maquinas_servicios.py, depositos_servicios.py, cc_servicios.py	Módulos CRUD básicos que dependen de servicios_infraestructura para la persistencia.
Tareas Pesadas	opciones_controlador.py	Este controlador utiliza el módulo **concurrent** (probablemente futures o threading) para ejecutar tareas potencialmente

		lentas (ej., importación masiva o regeneración de reportes) en segundo plano, mejorando la experiencia del usuario.
Auditoría	logs_servicios.py	Accede directamente al sistema de archivos mediante el módulo **os** para leer los archivos de registro (logs) generados por la aplicación.

6. Área de Mantenimiento (mantenimiento/)

Administra las Órdenes de Trabajo (OT) y el mantenimiento preventivo/correctivo.

Flujo de Mantenimiento	Módulos de Servicio Clave	Interacción Detallada
Órdenes de Trabajo	ordenesdetrabajo_servicios.py, ordenesdetrabajodetalle_servicios.py	Lógica para la creación, asignación y cierre de OTs. El detalle de la OT puede requerir datos de **Servicios Comunes** (ej., inventario de repuestos).
Matricería	opmatriceria_servicios.py	Gestión específica para las OTs de las matrices o herramentales, usando servicios_infraestructura para el acceso a datos.

III. □ Servicios Comunes y Sincronizadores

1. Servicios Comunes (servicios/comunes/)

Actúan como una capa de abstracción de datos compartidos, siendo consumidos por múltiples áreas funcionales (Ingeniería, Inventario, Costos).

- **articulo_servicios.py, proveedor_servicios.py:** Maestros de datos centrales.
- **stock_y_movimiento_*_servicios.py:** Lógica de negocio sobre el stock que es compartida por Inventario y Piso de Planta.
- **op_servicios.py:** Funciones transversales para la gestión de Órdenes de Producción.

2. Sincronizadores (servicios/sincronizadores/)

Gestionan la integración con sistemas externos (ERP/Contabilidad) utilizando la capa de infraestructura.

- **Entrantes (entrantes/*_sinc.py):** Actualizan los datos maestros en el sistema a partir de datos del ERP (ej., tarifas_sinc.py, cuentas_sinc.py).
- **Salientes (salientes/*_sinc.py):** Envían las transacciones generadas en el sistema (ej., asientos_sinc.py, consumos_sinc.py, fabricados_sinc.py) al sistema ERP/Contable. La comunicación se realiza a través de servicios_infraestructura/libertya.py, lo que implica el uso de **suds** para el protocolo SOAP.
