Data:
fichero expediciones_test.xlsx con datos de expediciones (entrega de pedidos a clientes) de muestra con las siguientes columnas con datos:
- idlinea: entero, identificador de la linea servida con una cantidad determinada de determinado material a determinado cliente.
- idReferencia: entero, identificador del material servido en la linea.
- referencia: string, nombre de la referencia
- cantidadPedida: numerico, cantidad a servir para esa linea.
- cantidadServida: numerico, cantidad realmente servida para esa linea.
- fechaTransporte: date, fecha en la que se envio la expedicion.

Fichero ubicaciones_test.xlsx con datos de muestra del stock actual de cada material por ubicaciones en el almacen con las siguientes columnas con datos:
- Ubicación: string, nombre de la ubicación en el almacén.
- referencia: string, nombre de la referencia.
- HU: string, nombre del bulto que consolida el material en la ubicación.
- Piezas: numérico, cantidad del material en el HU y la ubicación.
- fecha: fecha de entrada del HU del material en el almacén.

Configuración del dashboard realizado con Dash:
- Filtros por mes y año para filtrar los datos de las expediciones (en caso de estar vacio serian todos los meses y años).
- filtro por cantidad de clientes a ser analizados en el dashboard cuando corresponda (datos y visualizaciones, de 1 hasta 8, por defecto 5).
- filtro por cantidad de referencias a ser analizados en el dashboard cuando corresponda (datos y visualizaciones, de 1 hasta 8, por defecto 5).
- Pestaña para análisis de nivel de servicio a los clientes en las expediciones para los hasta 8 primeros clientes ordenados por la mayor cantidad pedida en expediciones. Debe incluir gráficos de conteo de expediciones, cantidad pedida, servida y nivel de servicio (calculado como cantidad servida/cantidad pedida) con la media representada.
- Pestaña para análisis de la importancia de las referencias en las expediciones para los hasta 8 primeras referencias ordenados por la mayor cantidad pedida en expediciones. Debe incluir la serie temporal de la referencia en cuanto a la cantidad servida y una estimación del valor esperado de la demanda en el proximo mes de dicha referencia basado en los datos.
- Pestaña para análisis de la importancia de las referencias en las ubicaciones para los hasta 8 primeras referencias ordenados por la mayor cantidad piezas ubicadas. Debe incluir el tiempo medio de los HU de la referencia en almacén.

Aspectos a tener en cuenta:
- Las funciones que sirvan los datos de entradas al dashboard deben de estar diseñadas y separadas bajo el enfoque que luego sirvan de herramientas (tools) a agentes de inteligencia artificial que serán incorporados como parte de la funcionalidad del dashboard.
- Todo debe ser en inglés, incluido docstring, comentarios y los elementos del dashboard.

Arquitectura de agentes de Inteligencia artificial:
- Se crea 3 Agentes especializados con tools, uno por cada pestaña del dashboard (siendo las tools las funciones que sirven los gráficos y tablas del dashboard).
Se instruyen como asistentes consultores especializados en cada aspecto de la gestión que se agrupa en cada pestaña del dashboard, de acuerdo a las funciones que se les brinda.

- Luego se crea un agente orquestador que usa como tools a los agentes previamente creados.
- Se le instruye que redireccione las consultas del usuario a alguno de los agentes especializados, y de ser necesario combine los resultados para ofrecer insights más complejos de acuerdo a la información brindada en el dashboard.
Esta arquitectura se diseña de esa forma para intentar corresponder el diseño visual del dashboard con las funcionalidades potenciadas con inteligencia artificial, haciendo la experiencia mas orgánica para la obtención de valor agregado para la toma de decisiones de los analisis que se muestran.

📝 Resumen de la Arquitectura y Optimización de la Solución de IA

El sistema de análisis del almacén emplea una **arquitectura multi-agente** jerárquica, compuesta por un **Agente Orquestador** central y **tres Agentes Especializados** (uno por dominio: Servicio al Cliente, Expediciones y Análisis de Stock), cada uno dotado de herramientas para calcular Indicadores Clave de Gestión (KPIs).

1. Arquitectura Inicial

- **Diseño:** Agente Orquestador → Tres Agentes Especializados (con herramientas de funciones KPI).
- **Problema Inicial:** La arquitectura generaba una alta tasa de llamadas API simultáneas (un "burst" por interacción de usuario), lo que resultaba en errores persistentes **429 RESOURCE_EXHAUSTED**.
- **Causa Raíz:** Se estaba utilizando el modelo `gemini-2.5-pro` para el Orquestador bajo el **Nivel Gratuito (Free Tier)** de la API. Este nivel impone un límite extremadamente bajo de **2 peticiones por minuto (RPM)** para el modelo Pro, un límite que la arquitectura multi-agente superaba instantáneamente.

---

2. Estrategia de Optimización (Capstone Project)

Dado el objetivo académico (Capstone Project) y la necesidad de permanecer en el Nivel Gratuito, se implementó una **solución de ingeniería de recursos y *backoff***:

A. Unificación y Migración de Modelos

Para evitar el límite de 2 RPM, todos los agentes fueron migrados a un único modelo más eficiente con límites de cuota superiores en el Free Tier:

- **Modelo Único: `gemini-2.5-flash`** (para el Orquestador y los 3 Agentes Especializados).
- **Beneficio:** El modelo Flash tiene un límite de RPM mucho más alto (típicamente ≥60 RPM en el Nivel Gratuito), permitiendo la **concurrencia** que la arquitectura de agentes anidados requiere.

B. Implementación de Gestión de Errores

Se configuró un mecanismo de reintento con **Exponential Backoff** para añadir resiliencia frente a picos de tráfico:

- **`HttpRetryOptions`:** Configuradas con **`attempts=4`** y **`exp_base=2`**.
- **Función:** Cuando se recibe un error 429, el sistema espera de manera creciente (ej., 2s, 4s, 8s) antes de reintentar la llamada, asegurando que el límite de tasa de la API se restablezca antes del siguiente intento.

C. Ajuste Fino de Instrucciones (Prompt Engineering)

Se ajustaron las instrucciones del **Agente Orquestador** para ser más explícito y directo, optimizando su rendimiento con el modelo `gemini-2.5-flash`:

- Se añadió una **Regla de Prioridad** para forzar el uso de herramientas (`tools`) antes de generar cualquier texto.
- Se aclaró la **Estructura de Respuesta Final** para asegurar la concisión y la integración de los hallazgos de los agentes.

---

3. Conclusión

La solución final adapta la potencia de la arquitectura multi-agente a las restricciones del Nivel Gratuito, garantizando un sistema **funcional, estable** y **eficiente en el uso de recursos** para su demostración en el Capstone Project.