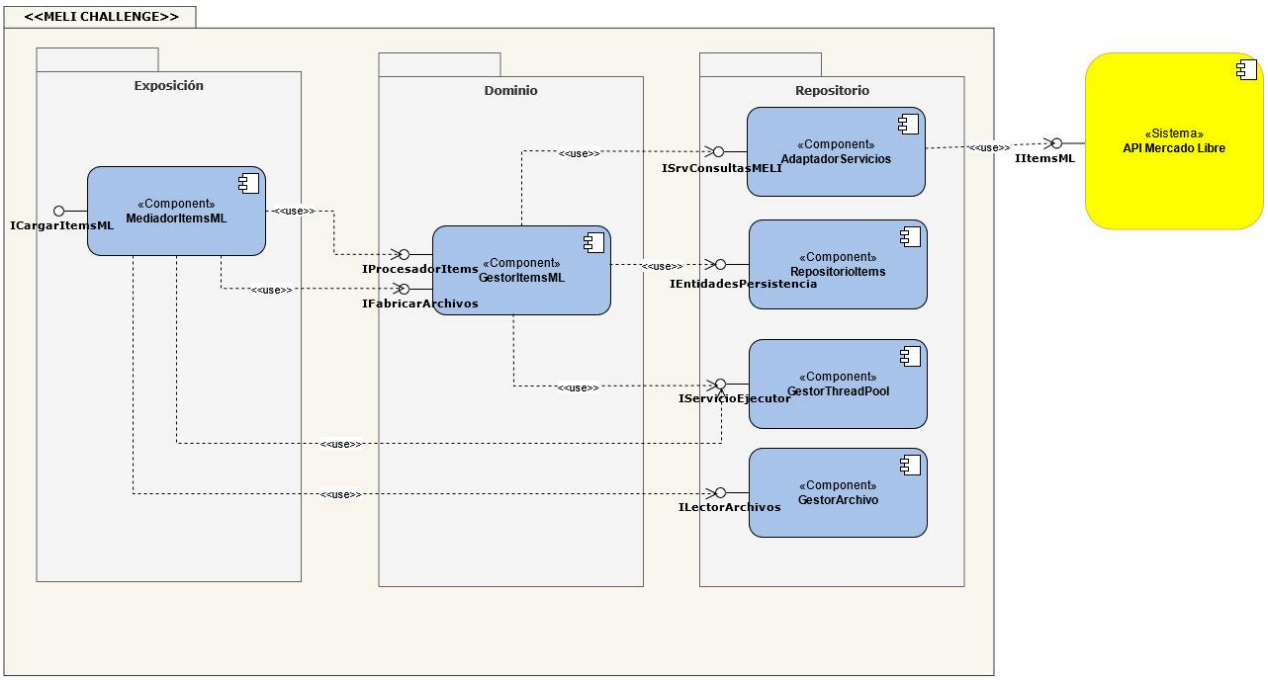
DOCUMENTO TÉCNICO DE LA SOLUCIÓN CHALLENGE ML

Documento Técnico de la Solución Challenge ML

Como base de la arquitectura se ha utilizado el estándar o patrón de Arquitectura Hexagonal para el diseño del microservicio y para modelar la lógica del negocio Domain Driven Design.

1. MODELO DE COMPONENTES DE ALTO NIVEL

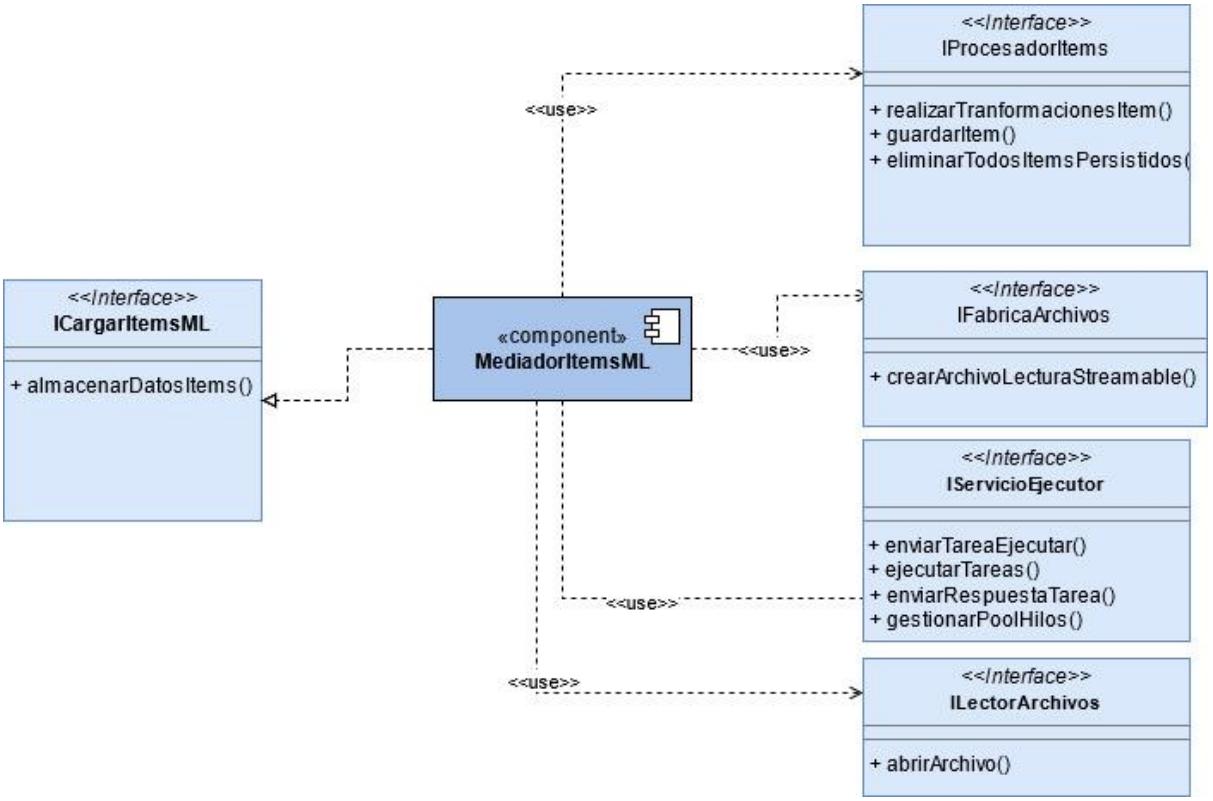


Nombre	Responsabilidades
MediadorItemsML	• Exponer el Servicio REST Cargar Ítems de MercadoLibre sobre HTTP, interfaz ICargarItemsML. • Usar la interfaz de la capa de dominio IProcesadorItems, para realizar el procesamiento de los registros de un archivo (items). • Usar la interfaz de la capa de dominio IFabricarArchivos, para construir un objeto de dominio que representa el archivo streamables que se desea procesar. • Usar la capa de infraestructura o repositorio para Gestionar la apertura del archivo a procesar.
GestorItemsML	• Exponer la interfaz IProcesadorItems, que permite gestionar el procesamiento de cada ítem, sus transformaciones y su persistencia. • Usar la interfaz ISrvConsultasMELI del adaptador de los servicios hacia la API de mercado libre. • Usar la capa de infraestructura o repositorio para persistencia de los ítems y eliminación de todos los ítems (IEntidadesPersistencia). • Usar la interfaz de de la capa de infraestructura o repositorio para realizar la ejecución concurrente de la invocación de los tres servicios que se pueden ejecutar en paralelo (Nivel 2), esperar la respuesta y la gestión de los hilos ante el sistema.
AdaptadorServicios	• Expone la interfaz ISrvConsultasMELI para llamar a los servicios de consulta de Mercado Libre (API MELI) Nivel 1: Ítem y Nivel 2: Categoría, Moneda y Vendedor. • Acceder a los servicios de la API de MELI, para recuperar la información de ítems, categorías, monedas y vendedores (Usuarios).

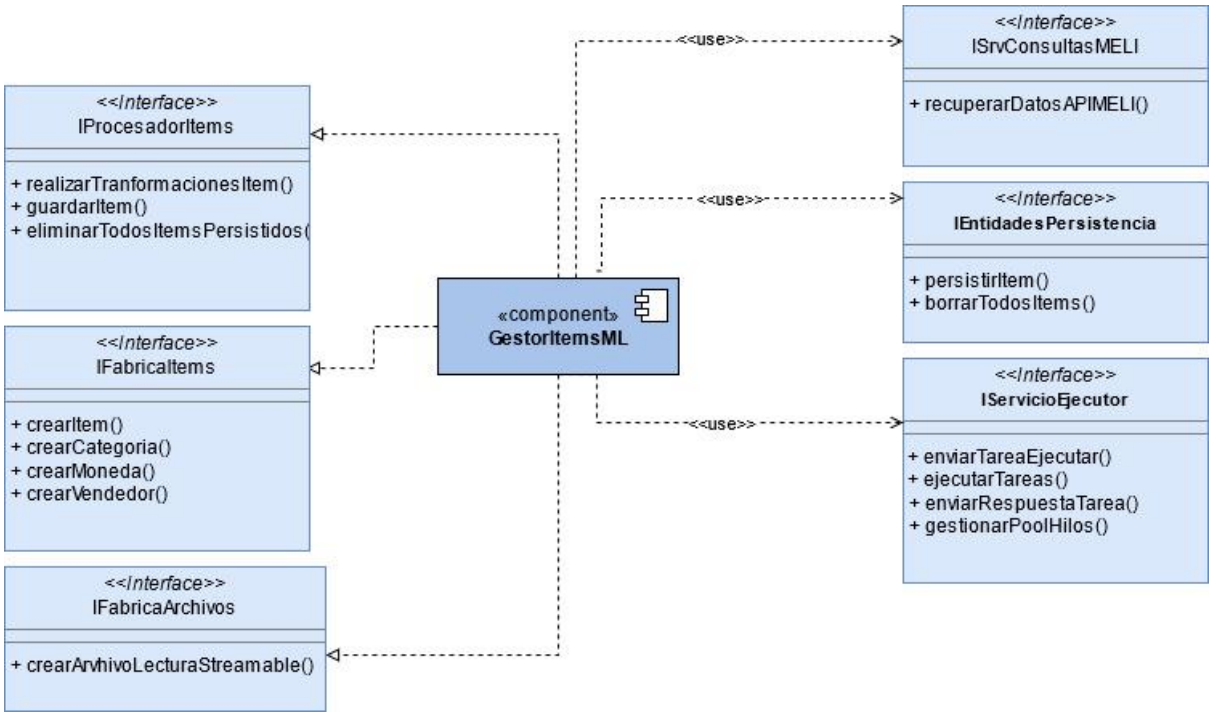
RepositorioItems	• Persistir los datos de ítems y su eliminación. • Exponer la interfaz IEntidadesPersistencia. • Su implementación se apoya en el uso del módulo de Python Flask-SQLAlchemy, el cual nos modela las tablas de base de datos como objetos ORM(Object Relational Mapping).
GestorThreadPool	• Exponer la interfaz IServicioEjecutor, que permite gestionar el ciclo de vida de los hilos, el pool, la ejecución de las tareas que se desean realizar de forma concurrente y el envío de las respuestas (futuros). • El componente es utilizado para el procesamiento concurrente de los registros que se encuentran en el archivo y su procesamiento (consulta servicios API MELI y persistencia en BD). • El componente es utilizado para el consumo en paralelo de los servicios de nivel 2 que son; consulta Categoría, consulta Moneda y consulta Usuarios (Vendedor). • Su implementación se realiza con base al framework concurrent.futures ThreadPoolExecutor, el cual gestiona el ciclo de vida de los hilos, ejecución y respuestas (futuros)
GestorArchivo	• Exponer la interfaz ILectorArchivos, que permite leer el archivo que se desea procesar (open) • Se Implementa con los módulos de python.

2. DETALLE DE PRINCIPALES COMPONENTES

2.1 Mediador Items ML (Capa de exposición) - Diagrama de Interfaces Provistas y Requeridas

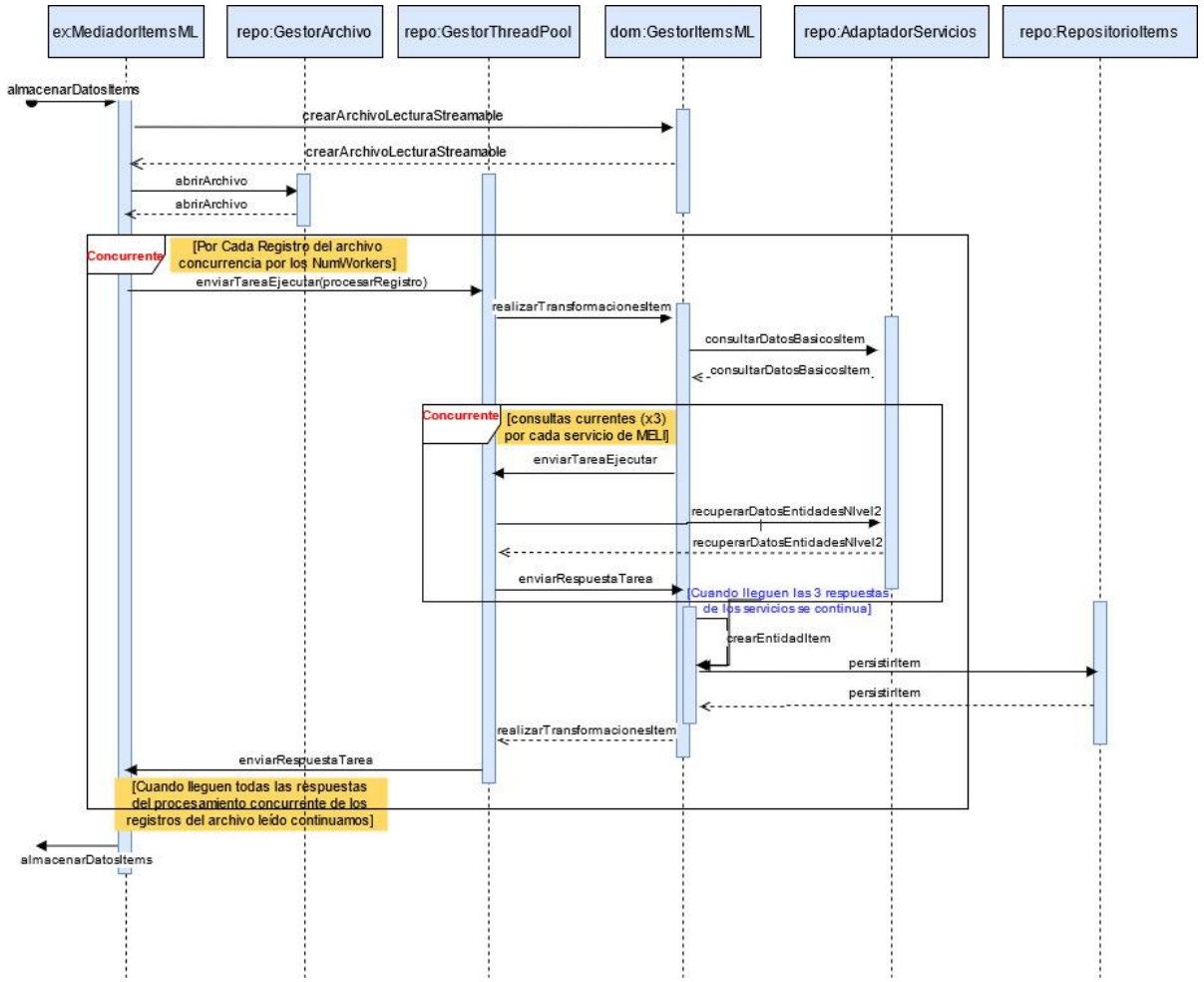


2.2 Gestor Items ML (Capa de exposición) - Diagrama de Interfaces Provistas y Requeridas

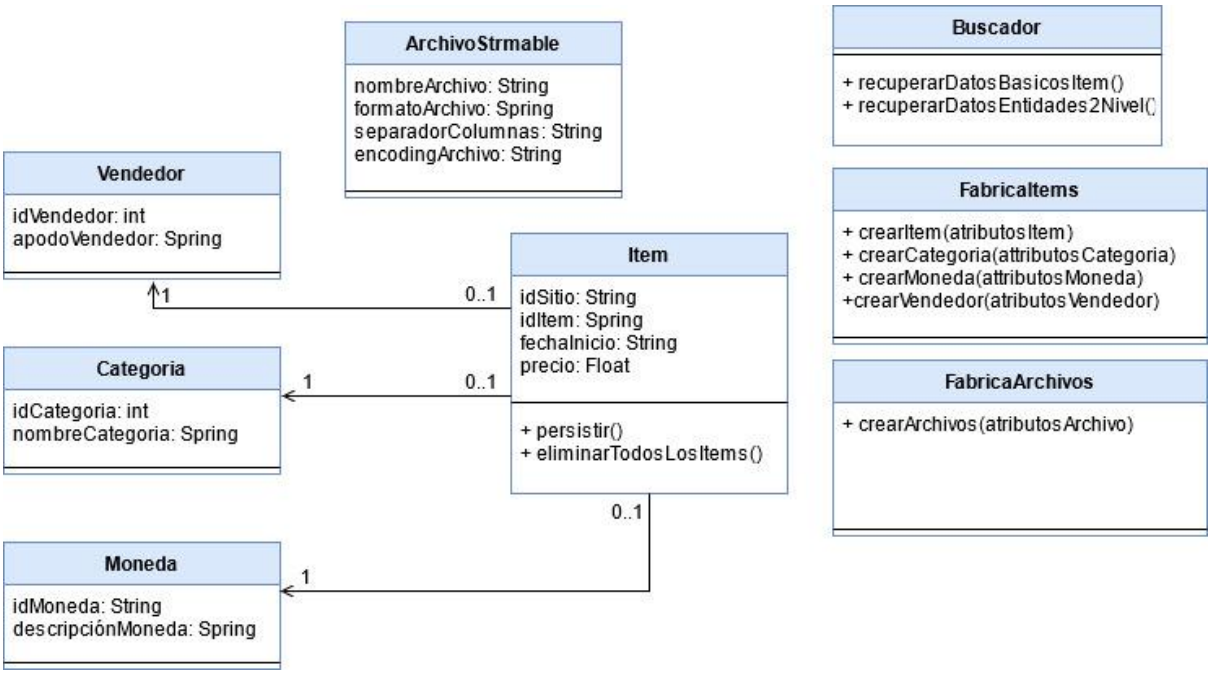


3. INTERACCIÓN COMPONENTES

3.1 DIAGRAMA DE SECUENCIA ALMACENARDATOSITEMS(): PRINCIPALES INTERACCIONES Y OPERACIONES DE CADA UNO DE LOS COMPONENTES.



4. MODELO BÁSICO DE DOMINIO DEL CHALLENGE ML



5. MODELO E/R (LA TABLA ITEM EN BASE DE DATOS)