



# Fuiden

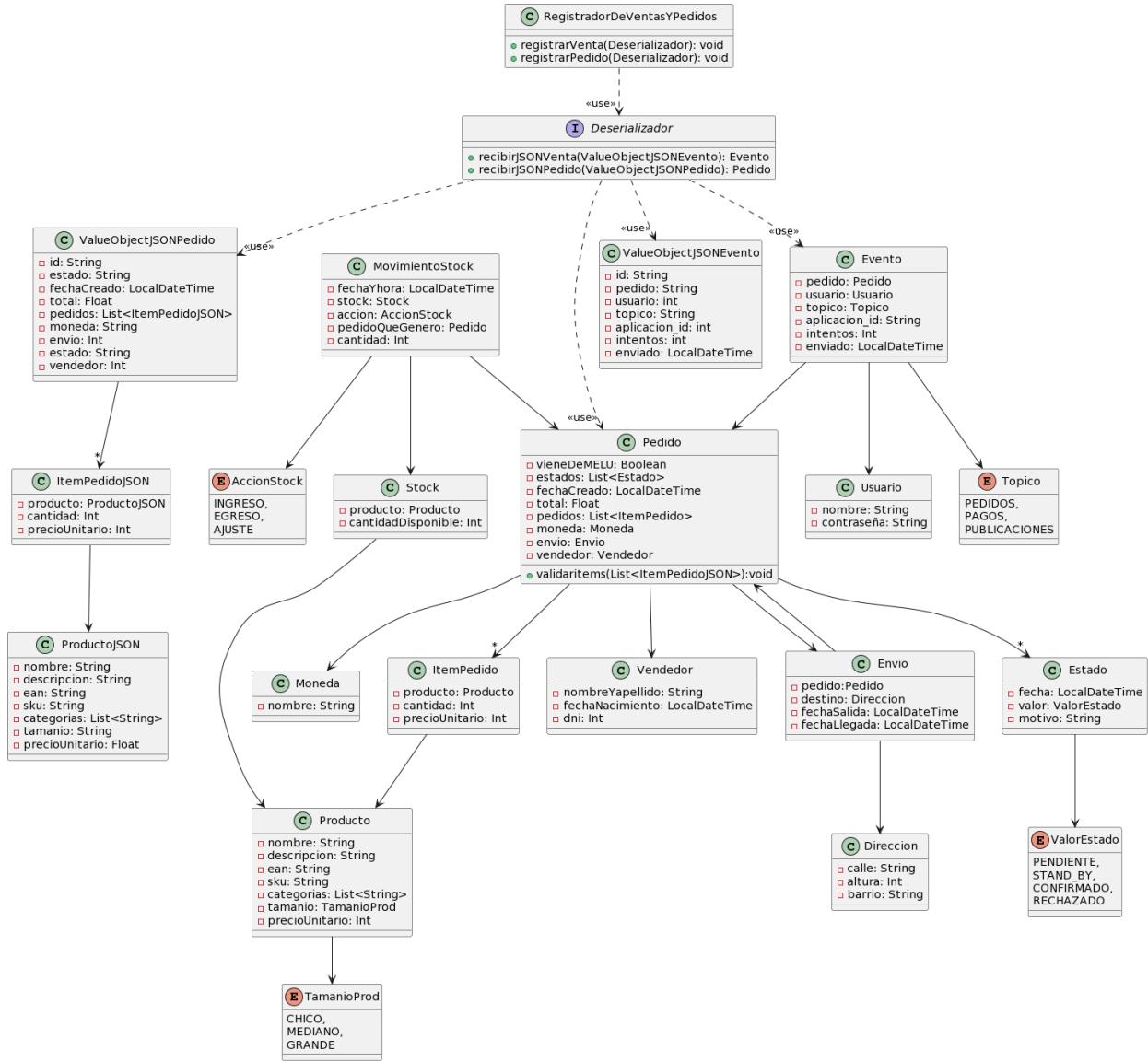
[Final DDS 20230805.pdf](#)

**Clase de consulta:** no hay

**Hecho por:** Chabela Lamas

## Modelado de objetos

### Diagrama de clases



## Decisiones de diseño

Por un lado, se decide crear el registrador de ventas y pedidos que sea el encargado de recibir los JSON que nos envía MELU. Es por ello que se envía por el parámetro de la función, un deserializador, una instancia de la clase creada por el controller. Debido a ello, éste es el encargado de convertir toda la información del JSON en la entidad correspondiente. No obstante, se decidió guardar constancia del ValueObject de llegada ya que el enunciado menciona “es importante guardar registro de todos los datos recuperados desde MELU para poder recrear las respuestas de las request”. A su vez, cuando se crea la entidad Pedido, el empleado es aquel que valida los ItemProducto del JSON y los añade a la lista (Los itemsProductoJSON no se persisten

pero se pueden guardar en una memoria secundaria o instanciarse en otra lugar que no sea la base de datos). Por consiguiente, el controller le ingresa por parámetro a la función la lista de ItemProducto que vino por el JSON. Cabe aclarar que, se generó una interfaz para deserializador para ganar mantenibilidad y extensibilidad al diseño, en caso de que se quieran crear diversos métodos para deserializar el JSON de llegada.

Por otro lado, los enumerados AccionStock, TamanioProd, ValorEstado y Tópico fueron creados para evitar que si se ingresan como String, se escriban de manera distinta pero representen lo mismo. Por ejemplo, "Chico" significa lo mismo que "ChlcO" o "CHICO".

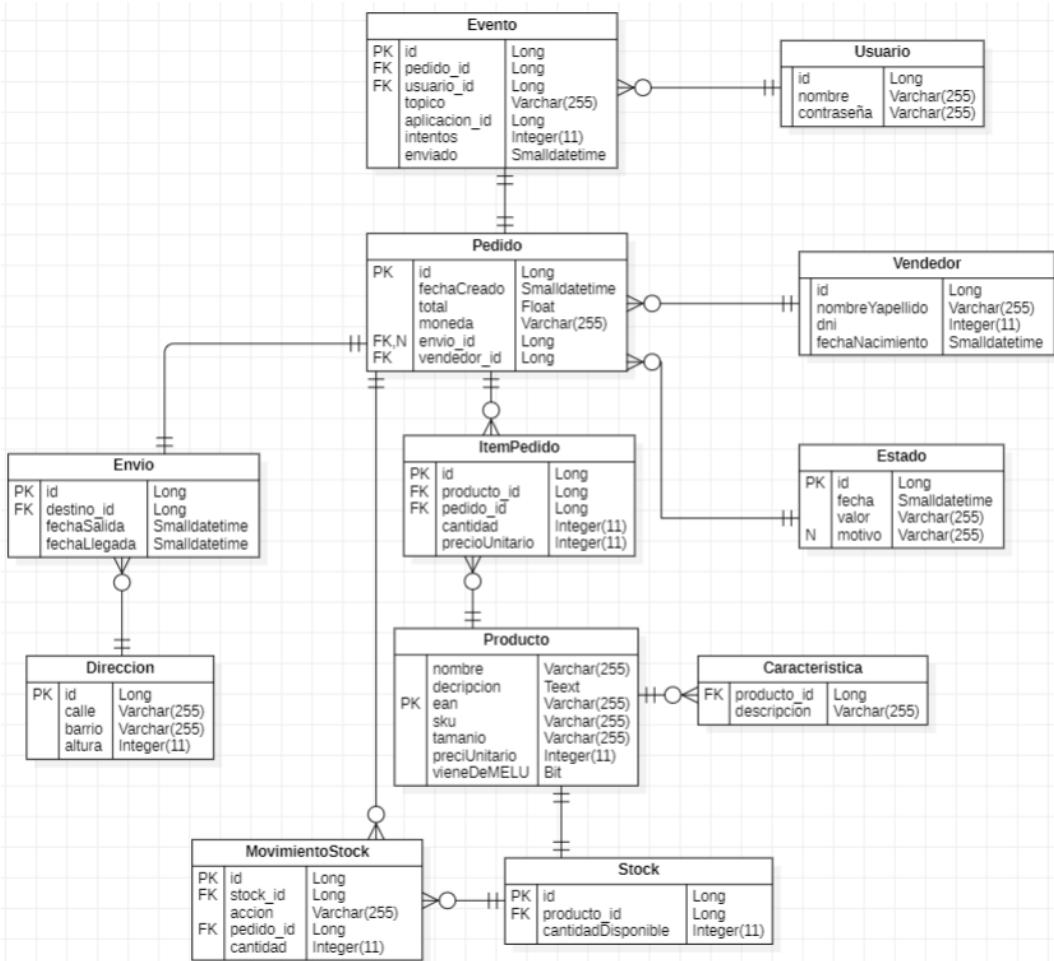
Para representar la trazabilidad de los estados de los pedidos, se decide crear una clase Estado que posea un valor, una fecha y un motivo (campo que será null en caso de que el valor no sea RECHAZADO).

La clase Moneda fue incluida para ganar consistencia de datos, permitiendo aumentar la mantenibilidad y extensibilidad del diseño.

Por lo que respecta a quién puede modificar o crear instancias de ciertas clases como es la de acción de stock, la capa de controllers regularán eso con el rol de cada persona que ingresa a la misma. Por ejemplo, ni bien se egresa un pedido, se puede determinar que el controller permita que los administradores instancien la disminución del stock.

## Persistencia

### Modelo de datos



## Decisiones de diseño

Se utiliza un converter para especificar la moneda, el tamaño del producto, el valor del estado del pedido, el tópico de la venta y la acción de stock en el movimiento. A su vez, se utiliza un element collection para especificar las categorías de los productos. Cabe aclarar que no hay una clase y tabla de aplicación\_id ya que consideramos que es información no relevante para este diagrama o sección del sistema.

## Entidades necesarias a persistir

- Vendedor
- ItemProducto
- Producto
- Stock

- MovimientoStock
- Dirección
- Envío
- Estado
- Usuario
- Evento

## Impedance mismatch

Tipo de dato objetos	Tipo de dato relacional
LocalDateTime	SMALLDATETIME
String	VARCHAR(255)
String	TEXT
Int	INTEGER(11)
Bool	BIT

Respecto al impedance mismatch de Identidad, tuvimos que agregar un campo “id” (clave subrogada) para identificar únicamente a cada una de nuestras entidades.

## Modelo desnормalizado

Se desnормaliza el precio unitario del producto en el ítem producto por consistencia de datos

## Arquitectura

1. Considerando que MELU espera que, ante la notificación de un evento, los Sistemas le respondan con un código de estado HTTP 200 en menos de 5 ms; ¿Cuál cree que sería la mejor forma de obtener los datos del recurso asociado en la notificación para evitar que MELU corte la

conexión por no contestar dentro del tiempo esperado? Detalle todo el mecanismo completo.

Se podría implementar una obtención de datos de los recursos manera asíncrona. Llegan los recursos, se almacenan en una cola y, luego de mandar el código de estado a MELU, se van procesando cada recurso de la cola por separado (se sacan de la cola y se analizan los datos). Por consiguiente, el componente involucrado es una cola de trabajo.

2. Fuiden nos ha mencionado que solamente utilizarán el Sistema a través del navegador

de una PC (no desde celulares). Además, sabemos que la interfaz debe ser fluida, evitando recargar páginas mientras sea posible.

a. ¿Qué tipo de cliente implementaría (cliente liviano o cliente pesado)? ¿Por qué?

b. ¿Cómo manejaría las sesiones de los usuarios?

a. Implementaría un cliente pesado ya que la primera request es la más pesada, permitiendo que luego la interfaz sea más fluida. A su vez, se menciona que se accede al sistema a través de una PC permitiendo que estos equipos puedan utilizar este método. De la misma manera, no se sobrecargan los servidores con este tipo de cliente, logrando manejar la alta concurrencia.

b. Las sesiones de los usuarios las manejaría con TOKENS al utilizar un cliente pesado. Mecanismo que es más seguro que las cookies del cliente liviano, al ser securizado. No obstante, se podría optar por un SSO que permita el fácil acceso de los clientes a sus sesiones. La desventaja de este método es que si se cae el SSO, no puedo entrar a las aplicaciones que hagan uso de esa sesión. Por ello, optaría más por la primera alternativa presentada.