DOCUMENTACIÓN API ORDERS

MÓDULO APPLICATION

PACKAGE CONTROLLER

OrdersController

Descripción:

OrdersController proporciona endpoints RESTful para la gestión de pedidos, interactuando con un servicio de pedidos (OrdersService) para importar, consultar y resumir datos de pedidos desde una API externa y una base de datos interna.

Endpoints

1. Importar y Resumir Pedidos
    - Método HTTP: GET
    - Path: /orders/import-summary
    - Descripción: Importa pedidos desde la API externa de Katas, los guarda en la base de datos y genera un resumen de los datos importados.
    - Parámetros:
        - page: Número de página para la solicitud (por defecto: 1).
        - maxPerPage: Máximo número de pedidos por página (por defecto: 100).
    - Respuesta: SummaryResponse que contiene el resumen de los pedidos importados.
2. Obtener Pedido por UUID
    - Método HTTP: GET
    - Path: /orders/{uuid}
    - Descripción: Recupera un pedido específico por su UUID desde la API externa de Katas.
    - Parámetros:
        - uuid: Identificador único del pedido.
    - Respuesta: OrdersSingleResponse que contiene el pedido correspondiente al UUID proporcionado.
3. Obtener Todos los Pedidos de la base de datos
    - Método HTTP: GET
    - Path: /orders
    - Descripción: Recupera todos los pedidos almacenados en la base de datos interna.

    - Respuesta: OrdersListResponse que contiene una lista de todos los pedidos almacenados.

Anotaciones

- @RestController: Indica que la clase es un controlador REST que maneja las solicitudes HTTP.
- @Tag: Proporciona información adicional para la documentación de OpenAPI (Swagger), etiquetando este controlador como parte de la "API gestión datos de Pedidos".
- @Operation: Describe cada endpoint con un resumen y una descripción detallada que se incluirá en la documentación de la API generada por Swagger.

Dependencias

- spring-cloud-starter-openfeign: Facilita la integración con servicios RESTful externos como la API de Katas.
- feign-jackson: Provee soporte para la serialización y deserialización de objetos JSON utilizados en las comunicaciones REST con Feign.

OrdersFileController

Descripción:

OrdersFileController proporciona un endpoint RESTful para generar y descargar un archivo CSV que contiene datos de pedidos.

Utiliza un servicio (OrdersFileService) para realizar la generación del archivo CSV.

Endpoint

1. Generar y Descargar CSV de Pedidos
    - Método HTTP: GET
    - Path: /file/csv
    - Descripción: Genera un archivo CSV con los datos de pedidos y permite su descarga.
    - Respuesta: Un archivo CSV con los datos de pedidos listo para ser descargado.

Anotaciones

- @RestController: Indica que la clase es un controlador REST que maneja las solicitudes HTTP.

- @Tag: Proporciona información adicional para la documentación de OpenAPI (Swagger), etiquetando este controlador como parte de la "Descarga de Ficheros" para la gestión de archivos de pedidos.
- @Operation: Describe el endpoint con un resumen y una descripción detallada que se incluirá en la documentación de la API generada por Swagger.

Dependencias

- spring-core: Proporciona la funcionalidad básica de Spring Framework, esencial para la operación del controlador.
- spring-boot-starter: Incluye dependencias necesarias para la configuración de aplicaciones Spring Boot, facilitando la ejecución del controlador en un entorno Spring Boot.
- spring-web: Provee soporte para aplicaciones web de Spring, incluyendo clases y métodos de utilidad para la creación de controladores RESTful.

MÓDULO DOMAIN

PACKAGE SERVICE

OrdersServiceImpl

La clase OrdersServiceImpl implementa la interfaz OrdersService y proporciona la lógica de negocio para interactuar con pedidos desde una API externa y una base de datos local.

Descripción

OrdersServiceImpl es un servicio de backend que facilita la importación, gestión y recuperación de pedidos mediante la integración con una API externa (API de Katas) y un repositorio local de base de datos (Schema katas)..

Atributos

- ordersClientService: Servicio para interactuar con la API externa de pedidos.
- ordersRepositoryService: Servicio para interactuar con el repositorio de base de datos local de pedidos.
- ordersMethods: Utilidades y métodos auxiliares para la transformación y manipulación de pedidos.

Métodos

1. importAndSummarizeOrders(String page, String maxPerPage)
    - Descripción: Importa pedidos desde la API externa, los transforma en entidades de base de datos, los guarda y luego genera un resumen de los pedidos importados.
    - Parámetros:
        - page - Número de página para la consulta paginada de pedidos desde la API.
        - maxPerPage - Máximo número de pedidos por página en la consulta paginada.
    - Retorno: SummaryResponse que contiene el resumen de los pedidos importados.
2. getOrderByUUID(String uuid)
    - Descripción: Obtiene un pedido específico por su UUID desde la API externa y lo transforma en un DTO de pedido para la respuesta.
    - Parámetros:
        - uuid - UUID único del pedido a recuperar.
    - Retorno: OrdersSingleResponse que contiene el DTO del pedido recuperado.
3. getOrders()
    - Descripción: Recupera todos los pedidos almacenados en la base de datos local y los transforma en DTOs de pedidos para la respuesta.
    - Retorno: OrdersListResponse que contiene la lista de DTOs de pedidos.

Uso

- **Importación y Almacenamiento**: Utiliza ordersClientService para obtener pedidos desde la API externa, los convierte en entidades Orders y los guarda en la base de datos local mediante ordersRepositoryService.
- **Transformación de Datos**: Usa ordersMethods para convertir entre diferentes representaciones de pedidos (DTOs y entidades).
- **Generación de Resumen**: Calcula estadísticas y genera un resumen de los pedidos importados, que se devuelve en SummaryResponse

OrdersFileServiceImpl

La clase OrdersFileServiceImpl implementa la interfaz OrdersFileService y se encarga de generar archivos CSV a partir de datos de pedidos obtenidos de una base de datos, manejando excepciones específicas de archivo y procesamiento.

Descripción

OrdersFileServiceImpl es un servicio que facilita la generación de archivos CSV a partir de datos de pedidos almacenados en una base de datos. Utiliza métodos auxiliares y mapeadores para transformar los datos y gestionar errores durante el proceso de generación del archivo.

Atributos

- ordersRepositoryService: Servicio para acceder a los datos de pedidos en la base de datos.
- ordersFileMethods: Utilidades y métodos auxiliares para la generación de archivos CSV.
- ordersMapper: Mapeador utilizado para convertir entidades de pedidos en DTOs de pedidos.

Métodos

1. generateOrderFileCSV()
    - Descripción: Recupera todos los registros de pedidos desde la base de datos, los convierte en DTOs de pedidos utilizando un mapeador, y genera un archivo CSV a partir de estos datos.
    - Excepciones Manejadas:
        - FileException: Se lanza si no se encuentran registros de pedidos en la base de datos.
        - ProcessingException: Se lanza si ocurre un error inesperado durante la generación del archivo CSV.
    - Retorno: OrdersFileResponse que contiene los bytes del archivo CSV generado y los encabezados HTTP necesarios.

Uso

- **Obtención de Datos**: Utiliza ordersRepositoryService para recuperar todos los registros de pedidos almacenados en la base de datos.
- **Transformación de Datos**: Utiliza ordersMapper para convertir entidades de pedidos en DTOs de pedidos.
- **Generación de Archivo CSV**: Utiliza ordersFileMethods para generar un archivo CSV a partir de los DTOs de pedidos y devuelve una respuesta encapsulada en OrdersFileResponse.

PACKAGE UTILS

OrdersFileMethods

La clase OrdersFileMethods es una utilidad para la generación de archivos CSV a partir de datos de pedidos representados por objetos OrdersDTO,

manejando también la configuración de encabezados HTTP necesarios para la descarga del archivo.

Descripción

OrdersFileMethods proporciona métodos para generar contenido CSV desde una lista de OrdersDTO, configurar encabezados HTTP para la descarga del archivo CSV y manejar excepciones relacionadas con la generación de archivos.

Atributos y Constantes

- CSV_HEADER: Encabezados estáticos para el archivo CSV generado, que describen las columnas del archivo.
- DATE_FORMAT: Formato de fecha utilizado para formatear las fechas en el archivo CSV generado.

Métodos Principales

1. generateOrderFileCSV(List<OrdersDTO> ordersDtoList, String fileName)
    - Descripción: Genera un archivo CSV completo a partir de una lista de OrdersDTO, configurando los encabezados HTTP y retornando un objeto OrdersFileResponse que encapsula los bytes del archivo y los encabezados.
    - Excepciones Manejadas:
        - IOException: Se lanza si ocurre un error de entrada/salida durante la generación del contenido CSV.
    - Retorno: OrdersFileResponse que contiene los bytes del archivo CSV generado y los encabezados HTTP necesarios para la descarga.
2. generateContentCSVFile(List<OrdersDTO> ordersDTOList)
    - Descripción: Genera el contenido del archivo CSV a partir de una lista de OrdersDTO, utilizando la librería OpenCSV para escribir los datos en formato CSV.
    - Excepciones Manejadas:
        - IOException: Se lanza si ocurre un error de entrada/salida durante la escritura del contenido CSV.
3. getCsvHttpHeaders(byte[] csvBytes, String filename)
    - Descripción: Configura los encabezados HTTP necesarios para la descarga de un archivo CSV, incluyendo el tipo de contenido, disposición de contenido y longitud del archivo.
    - Retorno: HttpHeaders configurados adecuadamente para el archivo CSV.
4. generateUniqueFileName(String filename)
    - Descripción: Genera un nombre de archivo único para evitar colisiones usando un número aleatorio.
    - Retorno: Nombre de archivo único con extensión ".csv".

Uso

- **Generación de Contenido CSV**: Utiliza generateContentCSVFile para generar el contenido del archivo CSV a partir de los datos de pedidos.
- **Configuración de Encabezados HTTP**: Utiliza getCsvHttpHeaders para establecer los encabezados HTTP necesarios para la descarga del archivo CSV.
- **Manejo de Excepciones**: Captura y maneja excepciones relacionadas con errores de entrada/salida durante la generación del archivo CSV.

OrdersMethods

La clase OrdersMethods proporciona métodos utilitarios para el procesamiento y manipulación de datos relacionados con pedidos, incluyendo la conversión entre diferentes tipos de DTO y entidades, así como la generación de resúmenes estadísticos basados en los datos de pedidos.

Descripción

OrdersMethods es una clase utilitaria diseñada para facilitar operaciones específicas relacionadas con pedidos, como la conversión de DTOs a entidades, la generación de resúmenes estadísticos y la manipulación de excepciones específicas del dominio.

Atributos y Constantes

- ordersMapper: Componente utilizado para mapear entre objetos OrdersDTO y entidades Orders.
- ordersValidations: Componente utilizado para validar objetos ContentClientDTO antes de su conversión a entidades Orders.

Métodos Principales

1. convertToOrders(PaginatedOrderClientDTO paginatedOrderClientDTO)
    - Descripción: Convierte objetos PaginatedOrderClientDTO provenientes de la API Katas a una lista de entidades Orders.
    - Excepciones Manejadas:
        - ProcessingException: Se lanza si ocurre un error durante la conversión o validación.
2. convertToOrdersDTO(List<Orders> orders)
    - Descripción: Convierte una lista de entidades Orders a una lista de objetos OrdersDTO.
    - Excepciones Manejadas: Ninguna explícita, pero asume validación previa de datos.
3. generateOrderSummary(List<OrdersDTO> ordersDTOList)
    - Descripción: Genera un resumen estadístico de los pedidos basado en campos como región, país, tipo de ítem, canal de ventas y prioridad de orden.
    - Excepciones Manejadas:

        - ProcessingException: Se lanza si la lista de pedidos está vacía o nula.
4. convertContentClientDTOToOrderDTO(ContentClientDTO contentClientDTO)
    - Descripción: Convierte un objeto ContentClientDTO a un objeto OrdersDTO.
    - Excepciones Manejadas: Ninguna explícita, pero asume validación previa de datos.

Métodos Privados

1. convertToEntityWithExceptionHandling(ContentClientDTO contentClientDTO)
    - Descripción: Convierte un objeto ContentClientDTO a una entidad Orders, manejando excepciones relacionadas con la validación de datos.
    - Excepciones Manejadas:
        - InvalidException: Se lanza si la validación de datos falla.
        - ProcessingException: Se lanza si ocurre un error inesperado durante el procesamiento.
2. generateSummary(List<OrdersDTO> ordersDTOList, Function<OrdersDTO, String> classifier, String logField)
    - Descripción: Genera un resumen agrupado de los pedidos basado en una función de clasificación y un campo de registro para propósitos de log.
    - Excepciones Manejadas: Ninguna explícita, pero asume operaciones seguras sobre los datos de entrada.

Uso

- Conversión de Datos: Utiliza métodos como convertToOrders y convertToOrdersDTO para manejar la conversión entre diferentes representaciones de datos de pedidos.
- Generación de Resúmenes: Utiliza generateOrderSummary para obtener estadísticas agrupadas de los pedidos según varios criterios.
- Manejo de Excepciones: Captura y maneja excepciones específicas del dominio, como InvalidException y ProcessingException, para asegurar la integridad y la consistencia de los datos procesados.

OrdersValidations

La clase OrdersValidations proporciona métodos utilitarios para validar diversos campos de datos utilizados en el servicio de órdenes. Se asegura de que los datos sean correctos y cumplan con los requisitos específicos antes de ser procesados.

Descripción

OrdersValidations es una clase utilitaria diseñada para realizar validaciones comunes en los datos de órdenes, tales como la validación de fechas, cadenas de texto y números. Las validaciones se aplican tanto a nivel de campo individual como a nivel de objeto completo.

Atributos y Constantes

- dateFormat: Formato de fecha utilizado para validar y parsear cadenas de fecha en el formato "dd/MM/yyyy".

Métodos Principales

1. validateParseDate(String dateString, String fieldName, String id)
    - Descripción: Valida y parsea una cadena de fecha en formato "dd/MM/yyyy" a un objeto Date.
    - Parámetros:
        - dateString: La cadena de fecha a validar.
        - fieldName: Nombre del campo de fecha para el mensaje de excepción.
        - id: Identificador asociado del objeto JSON.
    - Excepciones Manejadas:
        - InvalidDateFormatException: Se lanza si la cadena de fecha es nula, vacía o no tiene el formato esperado.
2. validateStringNotEmpty(String fieldValue, String fieldName, String id)
    - Descripción: Valida que una cadena de texto no esté vacía o sea nula.
    - Parámetros:
        - fieldValue: La cadena de texto a validar.
        - fieldName: Nombre del campo para el mensaje de excepción.
        - id: Identificador asociado del objeto JSON.
    - Excepciones Manejadas:
        - InvalidStringValueException: Se lanza si la cadena es nula o está vacía.
3. validatePositiveNumber(Number fieldValue, String fieldName, String id)
    - Descripción: Valida que un número sea positivo.
    - Parámetros:
        - fieldValue: El número a validar.
        - fieldName: Nombre del campo para el mensaje de excepción.
        - id: Identificador asociado del objeto JSON.
    - Excepciones Manejadas:
        - InvalidNumberValueException: Se lanza si el número es nulo o no es positivo.
4. validateContentClientDTO(ContentClientDTO contentClientDTO)
    - Descripción: Valida todos los campos de un objeto ContentClientDTO antes de mapearlo a la entidad Order.
    - Parámetros:

        - contentClientDTO: DTO que contiene la información de la orden.
    - Excepciones Manejadas:
        - ParseException: Se lanza si algún dato no pasa las validaciones.

Consideraciones

- **Formato de Fecha**: Asegúrese de que las cadenas de fecha cumplan con el formato "dd/MM/yyyy" antes de llamarlas.
- **Cadena No Vacía**: Verifique que las cadenas de texto no estén vacías o sean nulas antes de procesarlas.
- Número Positivo: Valide que los números sean positivos para evitar datos inválidos.
- **Manejo de Excepciones**: La clase lanza excepciones específicas para cada tipo de validación fallida, lo que facilita el manejo de errores en niveles superiores de la aplicación.

PACKAGE MAPPER

OrdersMapper

La interfaz OrdersMapper proporciona métodos de mapeo entre los objetos de transferencia de datos (DTO) y las entidades del dominio relacionadas con las órdenes.

Utiliza MapStruct para generar automáticamente las implementaciones de los métodos de mapeo.

Descripción

OrdersMapper es una interfaz que define métodos para convertir entre ContentClientDTO, Orders, y OrdersDTO.

Estos métodos aseguran que los datos se transformen correctamente entre las diferentes capas de la aplicación.

Anotaciones y MapStruct

- @Mapper(componentModel = "spring"): Indica que esta interfaz es un mapper de MapStruct y que debe ser gestionado por el contenedor de Spring.

Métodos Principales

1. contentClientDTOToOrders(ContentClientDTO contentClientDTO)
    - Descripción: Convierte un objeto ContentClientDTO en una entidad Orders.
    - Parámetros:
        - contentClientDTO: DTO que contiene la información de la orden.
    - Retorno:
        - Orders: Entidad mapeada desde el DTO.
    - Mapeos Específicos:
        - id a orderId
        - priority a orderPriority
        - date a orderDate con el formato dd/MM/yyyy
        - shipDate a shipDate con el formato dd/MM/yyyy
2. ordersToOrderDTO(Orders orders)
    - Descripción: Convierte una entidad Orders en un objeto OrdersDTO.
    - Parámetros:
        - orders: Entidad de órdenes.
    - Retorno:
        - OrdersDTO: DTO mapeado desde la entidad.
3. contentClientDTOToOrderDTO(ContentClientDTO contentClientDTO)
    - Descripción: Convierte un objeto ContentClientDTO en un OrdersDTO.
    - Parámetros:
        - contentClientDTO: DTO que contiene la información de la orden.
    - Retorno:
        - OrdersDTO: DTO mapeado desde el ContentClientDTO.
    - Mapeos Específicos:
        - id a orderId
        - priority a orderPriority
        - date a orderDate con el formato dd/MM/yyyy
        - shipDate a shipDate con el formato dd/MM/yyyy

Consideraciones

- **Formato de Fecha**: Asegúrese de que las fechas en los DTO se ajusten al formato dd/MM/yyyy para que los mapeos se realicen correctamente.
- **Gestión de Componentes de Spring**: La anotación @Mapper(componentModel = "spring") permite que el mapper sea gestionado por Spring, lo que facilita su inyección en otros componentes de la aplicación.

PACKAGE EXCEPCIONES

InvalidException

Descripción:

InvalidException es una excepción personalizada que extiende de RuntimeException.

Su uso está destinado para manejar diferentes tipos de errores de validación y procesamiento en la aplicación relacionados con formatos de fecha inválidos, valores de cadena no válidos y valores numéricos no válidos.

Constructores

1. InvalidDateFormatException
    - Descripción: Subclase de ProcessingException específica para errores relacionados con formatos de fecha inválidos.
    - Constructor:
        - InvalidDateFormatException(String message, Throwable cause): Construye una excepción con un mensaje descriptivo y la causa original del error.
2. InvalidStringValueException
    - Descripción: Subclase de ProcessingException específica para errores relacionados con valores de cadena no válidos.
    - Constructor:
        - InvalidStringValueException(String message): Construye una excepción con un mensaje descriptivo del error.
3. InvalidNumberValueException
    - Descripción: Subclase de ProcessingException específica para errores relacionados con valores numéricos no válidos.
    - Constructor:
        - InvalidNumberValueException(String message): Construye una excepción con un mensaje descriptivo del error.

ProcessingException

Descripción:

ProcessingException es una excepción propia que extiende de RuntimeException.

Su uso está destinado para representar errores generales durante el procesamiento de operaciones en la aplicación.

Proporciona constructores para manejar mensajes descriptivos de error y causas originales.

Constructores

1. ProcessingException(String message)
    - Descripción: Construye una excepción con un mensaje descriptivo del error.
    - Parámetros:
        - message: Mensaje detallado que describe la causa del error.
2. ProcessingException(String message, Throwable cause)
    - Descripción: Construye una excepción con un mensaje descriptivo del error y la causa original.
    - Parámetros:
        - message: Mensaje detallado que describe la causa del error.
        - cause: Causa original de la excepción, que puede ser útil para el rastreo y la depuración.

FileException

Descripción:

FileException extiende RuntimeException.

Su uso está destinado para representar errores relacionados con operaciones de ficheros en la aplicación.

Constructores

1. FileException(String message)
    - Descripción: Construye una nueva FileException con el mensaje detallado especificado.
    - Parámetros:
        - message - El mensaje detallado que se guarda para su posterior recuperación mediante el método getMessage().
2. FileException(String message, Throwable cause)
    - Descripción: Construye una nueva FileException con el mensaje detallado y la causa especificada.
    - Parámetros:
        - message - El mensaje detallado que se guarda para su posterior recuperación mediante el método getMessage().
        - cause - La causa de la excepción que se guarda para su posterior recuperación mediante el método getCause().

MÓDULO INFRASTRUCTURE

PACKAGE CLIENT

KatasClientFeign

La interfaz KatasClientFeign define un cliente Feign para comunicarse con la API de Katas. Utiliza Feign para simplificar las llamadas HTTP, proporcionando una manera declarativa de interactuar con los servicios web externos.

Descripción

KatasClientFeign es una interfaz anotada con @FeignClient, lo que permite que Spring Boot gestione la configuración y las llamadas a la API de Katas. Define métodos para obtener órdenes paginadas y órdenes específicas por UUID.

Anotaciones y Feign

- @FeignClient(name = "katas-client", url = "https://kata-espublicotech.g3stiona.com/v1/"): Define el nombre del cliente y la URL base de la API de Katas.

Métodos Principales

1. getClientOrders(String page, String maxPerPage)
    - Descripción: Obtiene una lista paginada de órdenes desde la API de Katas.
    - Método HTTP: GET
    - Ruta: https://kata-espublicotech.g3stiona.com/v1/orders
    - Parámetros:
        - page (query parameter): Número de página a recuperar. Valor por defecto es "1".
        - maxPerPage (query parameter): Número máximo de órdenes por página. Valor por defecto es "100".
    - Retorno:
        - PaginatedOrderClientDTO: DTO que contiene la lista paginada de órdenes.
2. getClientOrderByUUID(String uuid)
    - Descripción: Obtiene una orden específica por su UUID desde la API de Katas.
    - Método HTTP: GET
    - Ruta: https://kata-espublicotech.g3stiona.com/v1/orders/{uuid}
    - Parámetros:
        - uuid (path variable): UUID de la orden a recuperar.
    - Retorno:

    - ContentClientDTO: DTO que contiene la información de la orden específica.

Consideraciones

- **Parámetros por defecto**: Los parámetros page y maxPerPage tienen valores por defecto definidos en defaultValue. Esto asegura que las solicitudes tengan siempre un valor válido.
- **Anotaciones Feign**: Las anotaciones @GetMapping, @RequestParam, y @PathVariable se utilizan para definir las rutas y los parámetros de las solicitudes HTTP.

PACKAGE DATABASE

OrdersRepository

La interfaz OrdersRepository extiende JpaRepository y proporciona métodos CRUD para la entidad Orders. Utiliza Spring Data JPA para simplificar las interacciones con la base de datos.

Descripción

OrdersRepository es una interfaz marcada con @Repository, lo que permite que Spring Boot la trate como un bean de repositorio. Al extender JpaRepository, hereda una serie de métodos para operaciones comunes de persistencia.

Anotaciones y Herencia

- @Repository: Marca la interfaz como un bean de repositorio, habilitando el manejo de excepciones específico de Spring y otras características relacionadas.
- Extiende JpaRepository<Orders, Long>: Proporciona métodos CRUD y de paginación/sorting para la entidad Orders.

PACKAGE SERVICE

OrdersClientService

La clase OrdersClientService es un servicio de Spring que proporciona métodos para interactuar con el servicio externo Katas utilizando un cliente Feign. Maneja la comunicación con el servicio externo y gestiona posibles excepciones que puedan surgir durante las llamadas.

Anotaciones y Dependencias

- @Slf4j: Proporciona un logger para la clase.
- @Service: Marca la clase como un servicio de Spring, permitiendo su inyección en otros componentes.

Campos

- katasClientFeign: Cliente Feign para interactuar con el servicio externo Katas.

Constructor

- OrdersClientService(KatasClientFeign katasClientFeign): Constructor que inyecta el cliente Feign.

Métodos

1. getPagedOrdersClient(String page, String maxPerPage): Obtiene pedidos paginados del servicio externo.
    - Parámetros:
        - page: Número de página a obtener.
        - maxPerPage: Número máximo de elementos por página.
    - Retorno:
        - PaginatedOrderClientDTO: DTO paginado de órdenes obtenidos del servicio externo.
    - Excepciones:
        - ProcessingException: Lanzada si ocurre un error al obtener las órdenes paginadas, indicando que falló la llamada al servicio externo.
    - Implementación:
        - Llama al método getClientOrders del cliente Feign.
        - Maneja excepciones FeignException y otras excepciones, registrando el error y lanzando ProcessingException.
2. getOrderByUUIDClient(String uuid): Obtiene un pedido por su UUID.
    - Parámetros:
        - uuid: Identificador único del pedido.
    - Retorno:

        - ContentClientDTO: DTO que contiene los datos del pedido.
    - Implementación:
        - Llama al método getClientOrderByUUID del cliente Feign.
        - Registra la llamada y el resultado.

Consideraciones

- **Gestión de Errores**: Los métodos manejan excepciones y lanzan ProcessingException para errores de comunicación o cualquier otro fallo durante la obtención de datos.
- **Uso de Feign**: Utiliza Feign como cliente HTTP para realizar las llamadas al servicio externo de manera declarativa.

OrdersRepositoryService

La clase OrdersRepositoryService es un servicio de Spring que proporciona métodos para interactuar con la base de datos a través del repositorio OrdersRepository. Esta clase maneja las operaciones de persistencia y recuperación de datos de la tabla Orders.

Anotaciones y Dependencias

- @Slf4j: Proporciona un logger para la clase.
- @Service: Marca la clase como un servicio de Spring, permitiendo su inyección en otros componentes.

Campos

- ordersRepository: Repositorio JPA para realizar operaciones CRUD en la entidad Orders.

Constructor

- OrdersRepositoryService(OrdersRepository ordersRepository): Constructor que inyecta el repositorio OrdersRepository.

Métodos

1. saveAllOrders(List<Orders> ordersList): Guarda una lista de pedidos en la tabla Orders.
    - Parámetros:
        - ordersList: Lista de pedidos a guardar.
    - Retorno: Ninguno.
    - Excepciones:
        - ProcessingException: Lanzada si ocurre un error durante el proceso de guardado.
    - Implementación:

        - Intenta guardar la lista de pedidos utilizando el método saveAll del repositorio.
        - Registra el intento de guardado y el resultado.
        - Maneja cualquier excepción lanzada durante el proceso, registrándola y lanzando ProcessingException.
2. findById(Long id): Recupera un pedido por su ID de la base de datos.
    - Parámetros:
        - id: ID del pedido a buscar.
    - Retorno:
        - Optional<Orders>: El pedido encontrado, o un Optional vacío si no se encuentra.
    - Implementación:
        - Utiliza el método findById del repositorio para recuperar el pedido por su ID.
3. findAll(): Recupera todos los pedidos de la base de datos.
    - Parámetros: Ninguno.
    - Retorno:
        - List<Orders>: Listado de todos los pedidos.
    - Implementación:
        - Utiliza el método findAll del repositorio para recuperar todos los pedidos.

Consideraciones

- **Gestión de Errores**: El método saveAllOrders maneja excepciones y lanza ProcessingException para cualquier fallo durante el guardado.
- **Uso del Repositorio**: Utiliza métodos proporcionados por OrdersRepository (saveAll, findById, findAll) para realizar operaciones de persistencia y recuperación de datos.

Fichero Propiedades de Spring

application.yml

El fichero de propiedades en formato yml (application.yml) configura varios aspectos de una aplicación Spring, incluyendo el nombre de la aplicación, el puerto del servidor, la configuración de la base de datos, la configuración de JPA, la configuración de Feign Client y la configuración de logging.

Propiedades Configuradas

1. spring.application.name
    - Descripción: Define el nombre de la aplicación Spring.
    - Valor: orders
2. spring.server.port

    - Descripción: Configura el puerto en el cual la aplicación Spring se ejecutará.
    - Valor: 8080
3. spring.datasource.url
    - Descripción: URL de conexión a la base de datos MySQL.
    - Valor: jdbc:mysql://localhost:3306/katas
4. spring.datasource.username
    - Descripción: Nombre de usuario para la base de datos MySQL.
    - Valor: root
5. spring.datasource.password
    - Descripción: Contraseña del usuario para la base de datos MySQL.
    - Valor: root
6. spring.datasource.driver-class-name
    - Descripción: Clase del driver JDBC para MySQL.
    - Valor: com.mysql.cj.jdbc.Driver
7. spring.jpa.hibernate.ddl-auto
    - Descripción: Acción a realizar por Hibernate al inicio de la aplicación en relación a la estructura de la base de datos.
    - Valor: update
8. spring.jpa.show-sql
    - Descripción: Habilita la visualización de consultas SQL generadas por Hibernate.
    - Valor: true
9. spring.jpa.open-in-view
    - Descripción: Configura si Hibernate abrirá una sesión para cada solicitud.
    - Valor: false
10. feign.client.config.default.connectTimeout
    - Descripción: Tiempo máximo de espera para establecer una conexión Feign.
    - Valor: 5000 (en milisegundos)
11. feign.client.config.default.readTimeout
    - Descripción: Tiempo máximo de espera para leer datos de la conexión Feign.
    - Valor: 5000 (en milisegundos)
12. feign.client.config.default.loggerLevel
    - Descripción: Nivel de registro para el cliente Feign.
    - Valor: basic
13. logging.level.root
    - Descripción: Nivel de registro para el registro raíz de la aplicación.
    - Valor: INFO
14. logging.level.org.springframework.web
    - Descripción: Nivel de registro para el paquete org.springframework.web.
    - Valor: DEBUG