DESARROLLO DE APLICACIONES OPEN SOURCE (SI729)
EXAMEN FINAL
2024-2

Sección: WX55

Profesores: Bautista Ubillús, Efraín Ricardo
Flores Moroco, Juan Antonio
Mori Paiva, Hugo Allan
Sánchez Seña, Alberto Wilmer
Velásquez Núñez, Ángel Augusto

Duración: 170 minutos

Indicaciones:

1. El examen consta de 1 pregunta, y tendrá **170 minutos** para resolverlas.
2. La pregunta es de tipo Proyecto de Software y la entrega de su respuesta es a través de envío de archivo empaquetado **.zip** (único formato válido) con nombre *upc-pre-202402-si729-<sección>-eb-u<código-estudiante>.zip*, conteniendo el proyecto de software, en la Actividad para el Examen final.
3. Puede utilizar como referencia los materiales publicados en el aula virtual, los sitios referenciados en el enunciado, así como sitios web de documentación de frameworks o tecnologías referenciadas.
4. Cada examen cuenta con un equipo académico, el cual estará conectado **durante el examen**.
5. El alumno debe dedicar los primeros 15 minutos a revisar las preguntas del examen y de presentarse alguna duda enviar un correo al(los) profesor(es)

   Secciones SV54, SW51, SW53, SW54, SW56: Mayta Guillermo, Jorge Luis al correo pcsijmay@upc.edu.pe
   Secciones SW57, WS51, WS53, WX51, WX55: Rojas Malasquez, Royer Edelwer al correo pcisrroj@upc.edu.pe

6. Los profesores en mención, solo recibirán correos provenientes de las cuentas UPC, de ninguna manera se recibirán correos de cuentas públicas.
7. Ante problemas técnicos, debe de forma obligatoria adjuntar evidencias del mismo, como capturas de pantalla, videos, fotos, etc. Siendo requisito fundamental que, en cada evidencia se pueda apreciar claramente la fecha y hora del sistema operativo del computador donde el alumno está rindiendo el examen.
8. No se recibirán problemas técnicos una vez culminado el examen.

Enunciado:

Caso Kinaxis.

Kinaxis (https://www.kinaxis.com/) es una empresa global que ofrece software y servicios basados en la nube para la gestión de la cadena de suministro.

Las soluciones de software de Kinaxis ayudan con una variedad de tareas de gestión de la cadena de suministro, entre ellas: gestión de inventario, cumplimiento de pedidos, planificación de la capacidad, programación maestra y planificación de ventas y operaciones (S&OP).

Pregunta 1 (20 p.).

Usted se integra al backend software developer team, a cargo de la creación de un **RESTful API** que brinde soporte a las operaciones de Kinaxis. El ecosistema de Kinaxis Platform requiere que el Kinaxis Supply Chain Management and Warehouse Management cuente con **Endpoints** en el RESTful API, para el manejo de la información de:
InventoryItem, conformada por los atributos id, kinaxisSku, status, minimumQuantity, availableQuantity, reservedQuantity, pendingSupplyQuantity.
OrderItem, conformadas por los atributos id, orderId, kinaxisSku, requestedQuantity, status, orderedAt.

Cada Inventory Item debe tener un status. Los status de Inventory Item son los siguientes:

Id	Name
0	AVAILABLE
1	SUPPLY_NEEDED
2	AUDITING
3	DISABLED

Cada Order Item debe tener un status. Los status de Order Item son los siguientes:

Id	Name
0	READY_FOR_DISPATCH
1	IN_PROCESS
2	DISPATCHING
3	COMPLETED

Como reglas de negocio, Kinaxis:

- Establece que la información que se desea preservar de los Inventory Items, incluye **id** (Long, Primary Key, Autogenerado), **kinaxisSku**[1] (identificador del negocio, embedded type Obligatorio, No vacío, solo puede contener un valor UUID válido), **status** (InventoryItemStatus Enumeration, Obligatorio), **minimumQuantity** (Double, Obligatorio, No nulo), **availableQuantity** (Double, Obligatorio, No nulo), **reservedQuantity** (Double, No Obligatorio), **pendingSupplyQuantity** (Double, No Obligatorio).
- Establece que la información de los **Order Items**, incluye **id** (Long, Primary Key, Autogenerado), **orderId** (Long, No nulo, mayor que cero), **kinaxisSku** (identificador del negocio, embedded type Obligatorio, No vacío, solo puede contener un valor UUID válido), **requestedQuantity** (Double, Obligatorio, No nulo, mayor que cero), **status** (OrderItemStatus Enumeration, Obligatorio), **orderedAt** (Date, Obligatorio, No nulo).
- Especifica que el atributo **kinaxisSku** debe ser embedded y solo puede contener un valor UUID válido, que debe proporcionarse (no autogenerarse) al momento de registrarse el inventory item.
- Especifica que **kinaxisSku** se considera un identificador interno del negocio. No se debe registrar en la base de datos dos **inventory items** con el mismo valor de **kinaxisSku**.
- Especifica que al momento de registrar un **inventory item**, el valor de **minimumQuantity** debe ser mayor que 5.
- Especifica que al momento de registrar un **inventory item**, el valor de **avaliableQuantity** debe ser por lo menos el doble del valor de **minimumQuantity**.
- Especifica que al momento de registrar un **inventory item**, no se debe incluir en el request valores para **status**, **reservedQuantity** ni para **pendingSupplyQuantity**. Se debe asignar por defecto *AVAILABLE* a **status** y cero tanto a *reservedQuantity* como a *pendingSupplyQuantity*.
- Indica que al momento de registrar un **order item**, no se debe incluir en el request valor para **status**.

[1] **kinaxisSku** contiene un SKU que es un identificador típico del dominio de control de inventarios. SKU son las siglas de Stock Keeping Unit. Para Kinaxis, el Kinaxis SKU Identifier es un identificador único que genera para el control de inventario de productos.

- Especifica que al momento de registrar un **order item**, el valor de **kinaxisSku** debe corresponder con el valor de **kinaxisSku** de un **inventory item** previamente registrado.
- Especifica que no se debe permitir registrar dos **order items** con la misma combinación de **orderId** y **kinaxisSku**.
- Al momento de registrar un **order item**, el valor de **requestedQuantity** debe ser mayor que cero. Si el valor de **requestedQuantity** puede ser atendido con el valor de **availableQuantity** para el **inventory item** correspondiente, entonces en el **inventory item** se resta **requestedQuantity** del valor de **availableQuantity** y se suma el valor de **requestedQuantity** al valor actual de **reservedQuantity** en dicho **inventory item**. Con ello el **order item** recibe el **status** *READY_FOR_DISPATCH*. Si el valor de **requestedQuantity** no puede ser atendido de forma total, en **inventory item** se resta lo que esté disponible en **availableQuantity** para adicionarlo a **reservedQuantity** y el saldo que falte atender debe sumarse al valor actual de **pendingSupplyQuantity**. En ese caso el **status** del **order item** recibe el valor de *IN_PROCESS*.
- Requiere que el valor de **orderedAt** no sea mayor que la fecha actual.
- Cuando el valor de **availableQuantity** de un **inventory item** pase a estar por debajo del valor de **minimumQuantity**, entonces se debe emitir un evento **MinimumQuantityThresholdReachedEvent**, conteniendo el **kinaxisSku** y **requestedSupplyQuantity** (cuyo valor se calcula como la suma de **minimumQuantity** y **pendingSupplyQuantity**) para el **inventory item**.
- Al emitirse el **evento MinimumQuantityThresholdReachedEvent**, el proceso a realizarse en el *Event Handler* debe consistir en mostrar un mensaje de info vía *Logger*[2] en consola indicando "*SCM: A supply order is needed for the product with SKU X with at least Y units*" donde *X* es el kinaxisSku e *Y* es el valor de **requestedSupplyQuantity**. Además, el **status** del **inventory item** debe pasar a *SUPPLY_NEEDED*.
- Especifica que tanto *Inventory Item* como *Order Item* pertenecen a diferentes bounded contexts, por lo que se necesita un *Anti-Corruption Layer* (*ACL*) para que el bounded context que lo requiera, acceda a capacidades que exponga el otro bounded context como parte de la implementación de una característica requerida.
- Especifica que tanto *Inventory Item* como *Order Item* son Aggregate Roots en sus respectivos bounded contexts, por lo que requiere contar con atributos de auditoría para registrar **createdAt** (fecha y hora de creación de registro) y **updatedAt** (fecha y hora de última actualización de registro), de uso interno y poblados de forma automática por el sistema al momento de las operaciones de creación o actualización.

Nota: *Para efectos del caso el concepto Order no se considera como Aggregate y está fuera del alcance, por lo que se considera a Order Item como Aggregate Root*.

Durante la etapa de desarrollo, le asignan trabajar en específico sobre dos Endpoints:

/api/v1/inventory-items
/api/v1/orders/{orderId}/items

Incluya como parte del desarrollo la implementación de todas las reglas de negocio.

Inventory Items Endpoint (/api/v1/inventory-items)
Debe implementar **solo una** operación en el RESTful API: agregar (POST) un **inventory item**. Los valores de **id** no se solicita, son autogenerados al momento de almacenar la información. Los valores de **kinaxisSku** que se proporciona como String deben poder convertirse a valores válidos de tipo UUID versión 4[3]. Al agregar se debe retornar en el response el status 201 (created). Incluya en el response un objeto con el *id* generado para el *inventory id* y sus demás atributos, incluyendo el *kinaxisSku* como String. El valor de status se debe incluir como String en el response. No incluya en el response los atributos de auditoría.

Order Items Endpoint (/api/v1/orders/{orderId}/items)

[2] **Nota**: Vea ejemplos de uso de Logger en el proyecto *learning center platform* revisado en clase.
[3] **Nota:** Para obtener valores UUID versión 4 válidos, puede utilizar la herramienta en línea ***Online UUID Generator*** (ver sección de referencias).

Debe implementar **solo una** operación sobre los **order items** en el RESTful API: agregar (POST) un **order item**. Los valores de **id** no se solicita, son autogenerados al momento de almacenar la información. El valor de orderId se especifica en el path y no debe ser parte del body del request. El valor de **kinaxisSku** debe aceptarse como String. Al agregar se debe retornar en el response el status 201 (created). Al agregar se debe retornar en el response el objeto para el order item, con el *id* generado para el mismo y sus demás atributos, incluyendo el *kinaxisSku* como String. El valor de status se debe incluir como String en el response. No incluya en el response los atributos de auditoría.

Technical constraints

1. Elabore la solución con Java 22 y Spring Boot Framework 3.X.
2. Cree su proyecto de software con el nombre ***si729ebu<código-estudiante>*** (por ejemplo, *si729ebu201621873*).
3. La información debe ser persistente en una base de datos relacional (MySQL), en un esquema **kinaxis**.
4. Los packages de su solución deben tener como nombre raíz **com.kinaxis.platform.u<código-estudiante>** (por ejemplo, com.*kinaxis.platform.u201621873*).
5. Considere que el concepto **Inventory Item** pertenece al bounded context **wms** y el concepto **Order Item** pertenece al bounded context **commerce**.
6. Considere el bounded context **shared** para elementos base comunes/reutilizables que puedan ser aprovechados o extendidos por elementos en otros bounded contexts.
7. Aplique buenas prácticas de Arquitectura de Software, enfoque de **Domain-Driven Design**, separación en bounded contexts, **layered architecture** (domain, application, interfaces, infrastructure), patrones de strategic y tactical Domain-Driven Design, patrón CQRS, patrón Anti-Corruption Layer (ACL), principios y patrones de diseño de software orientado a objetos, convenciones de nomenclatura en **inglés**, así como buenas prácticas de nomenclatura en Java (entre ellas Upper-Camel-Case para Clases, Lower-Camel-Case para atributos y métodos) y buenas prácticas para nomenclatura de objetos de Base de Datos (entre ellas snake case, tablas en plural, sin mnemónicos).
8. Aplique reglas de object-relational mapping, implementando en shared un physical naming strategy que establezca de forma automática las convenciones de snake case para objetos de base de datos, así como plural para nombres de tablas.
9. Considere el bounded context **shared** para elementos base comunes/reutilizables que puedan ser aprovechados o extendidos por elementos en otros bounded contexts. El contenido del bounded context shared debe seguir las especificaciones del bounded context *shared* del proyecto de ejemplo *learning center platform* revisado en clase.
10. Utilice minúsculas para los nombres de URL y términos compuestos separados por guión medio (-) para todos los endpoints.
11. Utilice la biblioteca Lombok para el manejo de métodos constructores y de acceso en las clases de Java.
12. Utilice records en vez de clases para almacenamiento de valores inmutables.
13. Para **inventory item** y **order item**, incluya atributos de auditoría createdAt y updatedAt con valores poblados de forma automática por Spring Boot al momento de la creación.
14. Utilice el patrón Assembler para el Object Mapping la sección *transform* en interfaces layer.
15. Documente su código con **JavaDoc** (ver referencias), colocando en inglés información de propósito para principales objetos de programación, así como propósito, parámetros y valor retornado en clases y métodos relevantes. Incluya como parte de la documentación sus nombres y apellidos como valor para @author.
16. Incluya documentación de Endpoints con **OpenAPI**.
17. Considere la gestión de excepciones en la aplicación.
18. Empaquete su solución como un archivo **.zip**. **(único formato válido)** con el nombre ***upc-pre-202402-si729-<sección>-eb-u<código-estudiante>.zip*** (por ejemplo, *upc-pre-202402-si729-sw51-eb-u201621873.zip*).
19. Suba su archivo de solución en la Actividad indicada para el Examen final.

NO forma parte del alcance del proyecto:

1. Soporte de CORS.
2. Security.
3. Testing.

Rúbrica de calificación

Criterio de Calificación	Sobresaliente (S)	Esperado (E)	Necesita Mejorar (M)	Insuficiente (I)	Calificación
C01. Building y ejecución	Al abrir el proyecto y ordenar la ejecución, ésta se inicia sin problemas. El API es accesible en la ruta indicada.	La aplicación no llega a iniciar y ejecutarse, sin embargo el proceso de building llega a concluir.	Al cargar el proyecto el proceso de building presenta errores y no llega a concluir.	No elabora solución.	
	2.0 puntos	**1.0 punto**	**0.5**	**0 puntos**	
C02. Examiners Endpoint	El RESTful API expone el endpoint especificado en el enunciado. Se evidencia la funcionalidad de las operaciones solicitadas, proporcionando en cada caso los valores esperados y respondiendo adecuadamente ante las excepciones. Se evidencia la persistencia de los objetos solicitados, cumpliendo la estructura según enunciado, en una tabla nombrada según especificaciones, en base de datos relacional según enunciado en el esquema indicado. Se incluye documentación del Endpoint con OpenAPI.	El RESTful API expone el endpoint especificado en el enunciado. Se evidencia parcialmente la funcionalidad de las operaciones solicitadas, proporcionando en algunos casos los valores esperados y respondiendo de forma parcialmente adecuada ante las excepciones, o se evidencia parcialmente la persistencia en base de datos relacional, o se evidencia parcialmente la persistencia de los objetos solicitados con estructura según enunciado, en una tabla *nombrada según especificaciones* en base de datos relacional según enunciado en el esquema *indicado*.	La aplicación implementa y expone el endpoint especificado en el enunciado, pero no cumple con la ruta especificada o no se puede registrar elementos.	La aplicación no implementa o expone el endpoint solicitado.	
	4.0 puntos	**3.0 puntos**	**1.5 puntos**	**0 puntos**	
C03. Mental State Exams Endpoint	El RESTful API expone el endpoint especificado en el enunciado. Se evidencia la funcionalidad de las operaciones solicitadas, proporcionando en cada caso los valores esperados y respondiendo adecuadamente ante las excepciones. Se evidencia la persistencia de los objetos solicitados, cumpliendo la estructura según enunciado, en una tabla nombrada según especificaciones, en base de datos relacional según enunciado en el esquema indicado. Se incluye documentación del Endpoint con OpenAPI.	El RESTful API expone el endpoint especificado en el enunciado. Se evidencia parcialmente la funcionalidad de las operaciones solicitadas, proporcionando en algunos casos los valores esperados y respondiendo de forma parcialmente adecuada ante las excepciones, o se evidencia parcialmente la persistencia en base de datos relacional, o se evidencia parcialmente la persistencia de los objetos solicitados con estructura según enunciado, en una tabla *nombrada según especificaciones* en base de datos relacional según enunciado en el esquema *indicado*.	La aplicación implementa y expone el endpoint especificado en el enunciado, pero no cumple con la ruta especificada o no se puede registrar elementos.	La aplicación no implementa o expone el endpoint solicitado.	
	4.0 puntos	**3.0 puntos**	**1.5 puntos**	**0 puntos**	
C04. Business Rules	El desarrollo incluye la implementación de reglas de negocio, cubriendo de forma completa las condiciones y escenarios establecidos, siendo éstas ejecutables, con adecuado manejo de excepciones, implementando éstas en las capas más adecuadas, aplicando convenciones y buenas prácticas.	El desarrollo incluye la implementación de la mayoría de reglas de negocio, cubriendo de forma parcial las condiciones y escenarios establecidos, siendo éstas ejecutables, implementándolas en las capas adecuadas en la mayoría de casos, ó aplicando parcialmente convenciones y buenas prácticas.	El desarrollo incluye la implementación de algunas de las reglas de negocio, incumpliendo la mayoría de condiciones y escenarios establecidos, siendo éstas ejecutables, implementándolas en las capas adecuadas en muy pocos casos, ó con poca evidencia de aplicar convenciones y buenas prácticas.	No implementa reglas de negocio, o no cubre escenarios más allá de operaciones CRUD básicas, o éstas no son ejecutables.	
	4.0 puntos	**3.0 puntos**	**1.5 puntos**	**0 puntos**	
C05. Code Organization	El desarrollador organiza el código y los elementos de backend de la solución, aplicando buenas prácticas de Java, Spring Boot Framework y Domain-Driven Design, agrupando los elementos de la solución según convenciones, manteniendo organización de paquetes y carpetas recomendadas por el fabricante y buenas prácticas de la industria de software.	El desarrollador aplica en la mayoría de casos para el backend convenciones, recomendaciones y buenas prácticas de Java, Spring Boot Framework y Domain-Driven Design.	El desarrollador aplica en algunos casos para el backend convenciones, recomendaciones y buenas prácticas de Java, Spring Boot Framework y Domain-Driven Design.	No se evidencia un criterio de organización para los elementos de la solución.	
	2.0 punto	**1.0 puntos**	**0.5**	**0 puntos**	
C06. Code Quality	Utiliza para el backend el lenguaje de programación Java La codificación tiene un estilo claro, indentando los bloques de código según los estándares de programación correspondientes al lenguaje, aplicando una lógica consistente en los métodos, condicionales sin escenarios no contemplados, uso adecuado de reutilización de código para evitar redundancia. Aplica patrones de arquitectura y patrones de diseño. Distribuye el código en los niveles correspondientes, asignando lógica de persistencia, lógica de negocio, lógica de control, lógica de mapping y transferencia a las interfaces y clases que corresponden. Cumple de forma completa con los technical constraints.	Utiliza para el backend el lenguaje de programación Java. La codificación es funcional, aplica en la mayoría de casos los estándares de indentación de bloques de código, ó existen algunas ineficiencias en la codificación: redundancia ó inconsistencias en la lógica de programación. Aplica parcialmente patrones de arquitectura y patrones de diseño, o existe en algunas partes una distribución de la lógica en los niveles incorrectos. Cumple con la mayoría de technical constraints.	Utiliza para el backend el lenguaje de programación Java. La codificación es funcional, pero solo aplica algunos de los estándares de indentación de bloques de código, ó existen muchas ineficiencias en la codificación: redundancia ó inconsistencias en la lógica de programación. Aplica algunos patrones de arquitectura y patrones de diseño, o existe en muchos casos una distribución de la lógica en los niveles incorrectos. Cumple con solo algunos de los technical constraints.	No utiliza el lenguaje de programación Java para el backend, ó la codificación es funcional pero no se evidencia aplicación de estándares ó criterios de eficiencia en la codificación, con ausencia de comentarios, ó no aplica patrones de arquitectura ni patrones de diseño, o la codificación no es funcional.	
	3.0 puntos	**2.0 punto**	**1.0**	**0 puntos**	
C07. Naming Standards	El desarrollador aplica en todos los nombres de objetos de programación y base de datos como paquetes, componentes, interfaces, clases, objetos, variables, constantes, métodos, tablas, columas la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, tablas, columnas, así como los recursos.	El desarrollador aplica en la mayoría de casos la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, así como los recursos.	El desarrollador aplica en muy pocos casos la nomenclatura en inglés y la nomenclatura estándar para identificadores de clases, objetos, miembros de programación, así como los recursos.	El desarrollador no aplica nomenclatura en inglés para los objetos de programación ó recursos.	
	1.0 puntos	**0.5 punto**	**0.25 puntos**	**0 puntos**	
Total	**20 puntos**	**13.5 puntos**	**6.5 puntos**	**0 puntos**	

Lima, 25 de Noviembre del 2024

Anexos

Anexos A. Referencias

Comprimir y descomprimir archivos:
https://support.microsoft.com/es-es/windows/comprimir-y-descomprimir-archivos-8d28fa72-f2f9-712f-67df-f80cf89fd4e5

REST API Tutorial:
https://restfulapi.net/

Project Lombok:
https://projectlombok.org/

springdoc-openapi:
https://springdoc.org/

Spring Data JPA - Reference Documentation:
https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference

Class UUID:
https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/util/UUID.html

Online UUID Generator:
https://www.uuidgenerator.net/version4

How to Write Doc Comments for the Javadoc Tool:
https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html

Formats for date and dateTime in JSON payloads
https://www.geeksforgeeks.org/formats-for-date-and-datetime-in-json-payloads/

Spring Data JPA Auditing:
https://docs.spring.io/spring-data/jpa/reference/auditing.html#auditing.annotations