DESARROLLO DE APLICACIONES OPEN SOURCE (SI729)
EXAMEN FINAL
2023-1

Sección: SW51, SW53, WS51, WS52, WX51, WX52
Profesores: Flores Moroco, Juan Antonio
Mori Paiva, Hugo Allan
Navarrete Vilca, Elio Jefferrson
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** con nombre *upc-pre-202301-si729-<sección>-eb-u<código-estudiante>.zip*, conteniendo el proyecto de software, en la Actividad para el Examen final.
3. Cada examen cuenta con un equipo académico, el cual estará conectado **durante el examen**.
4. 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 WS51, WX52: García Rojas, Fidel Eugenio al correo pcsifgar@upc.edu.pe
   Secciones SW51, WS52: Prialé De La Peña, Mónica Rosario al correo pcsimpri@upc.edu.pe
   Secciones SW53, WX51: Rojas Malasquez, Royer Edelwer al correo pcisrroj@upc.edu.pe

5. Los profesores en mención, solo recibirán correos provenientes de las cuentas UPC, de ninguna manera se recibirán correos de cuentas públicas.
6. 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.
7. Los problemas técnicos se recibirán como máximo 15 minutos culminado el examen.

Enunciado:

Caso ISA.

En el corazón de toda cocina profesional se encuentra el frigorífico. Cualquier dueño de negocio que dependa de alimentos frescos será muy consciente de la importancia de la refrigeración comercial para su negocio. Para tales empresas, los alimentos frescos son la principal fuente de ingresos, por lo que es crucial que sus aparatos de refrigeración comercial funcionen de manera confiable y efectiva.

Mantener temperaturas óptimas garantiza que los dueños de negocios puedan ofrecer a sus clientes alimentos frescos. Si la temperatura llega a estar por encima o por debajo del rango óptimo, se producirá el deterioro y la descomposición de los alimentos. Los productos alimenticios en mal estado, el incumplimiento de las normas alimentarias y la avería de los equipos contribuyen a la pérdida de ingresos, al daño a la reputación y, a menudo, generan gastos adicionales, incluso multas.

Desde 1963 ISA (https://www.isaitaly.com/) produce en su región y desde allí provee vitrinas refrigeradas y muebles refrigerados para lugares públicos a nivel mundial. ISA actúa en el mercado a través de tres marcas: ISA, COF, TASSELLI e HIZONE con un volumen de ventas superior a los 120 millones de euros exportando, en 107 países, productos de calidad con un alto índice de tecnología e innovación, también en términos de sostenibilidad mediante el uso refrigerantes naturales.

ISA es hoy en día uno de los jugadores más importantes del mundo en el campo del diseño de vitrinas y armarios frigoríficos para heladería y pastelería. Más allá de numerosos y prestigiosos clientes, que hacen uso diario de los equipos de ISA, la empresa colabora desde hace años con importantes marcas internacionales, como: Ahold, Auchan, Autogrill, Billa, Bindi, Coldstone Creamery, Coca Cola, Conad, Coop, Cremonini , Brioche Doree, Haagen Dazs, Nestlé, Sammontana, Sturbucks y Unilever.

ISA opera en un mercado competitivo donde los clientes exigen un producto confiable y características adicionales que se suman a la experiencia del usuario. Para ISA, la conectividad IoT proporciona la base para un conjunto más amplio de nuevas características y beneficios. A continuación, exploramos algunos de los beneficios clave que IoT genera para ISA.

ISA Connect es una plataforma que incluye un conjunto de productos relacionados para facilitar el mantenimiento predictivo y el monitoreo continuo.

Para ISA, es importante que sus soluciones ofrezcan:

- Monitoreo remoto del funcionamiento de la vitrina 24/7 durante todo el año.
- Mantenimiento remoto de las funciones principales de la vitrina.
- Estadísticas operativas y análisis de datos.
- Locales integrados y conectados: La tecnología ISA Connect se aplica a la amplia gama de productos ISA, desde heladerías hasta supermercados, desde los escaparates hasta las cocinas comerciales.

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 ISA.

El ecosistema de ecodrive.com requiere que el ISA Connect Web Application, el Embedded IoT System y el ISA Connect Mobile Application cuenten con **Endpoints** en el RESTful API, para el manejo de la información de:

Products, conformadas por los atributos id (Long, Primary Key, Autogenerado), brand (String, Obligatorio), model (String, Obligatorio), serialNumber (String, Obligatorio), monitoringLevel (Integer, Obligatorio, Valores posibles 1 = Essential Monitoring, 2 = Advance Monitoring)

Snapshots, conformadas por los atributos id (Long, Primary Key, Autogenerado), snapshotId (String, Obligatorio, proporcionado como parte del snapshot), productSerialNumber (String, Obligatorio, Serial Number del product que genera el record), temperature (Double, obligatorio, Valor en Grados centígrados), energy (Double, Obligatorio para products con Advance Monitoring Level, Valor de consumo en Watts), leakage (Integer, Obligatorio para products con Advance Monitoring Level, Valores posibles 0 = No Leakage, 1 = Leakage).

Como reglas de negocio, ISA:

- No permite que se registre dos **products** con el mismo valor de *serialNumber*.
- No permite que se registre dos **snapshots** con el mismo valor de *snapshotId*.
- Establece que la información que se desea preservar de los ***Products,*** incluye **id** (Long, Primary Key, Autogenerado), **brand** (String, Obligatorio), **model** (String, Obligatorio), **serialNumber** (String, Obligatorio), **monitoringLevel** (Integer, Obligatorio, Valores posibles 1 = Essential Monitoring, 2 = Advance Monitoring)
- Establece que la información de los ***Snapshots***, incluye **id** (Long, Primary Key, Autogenerado), **snapshotId** (String, Obligatorio, no vacío), **productSerialNumber** (String, Obligatorio, no vacío), **temperature** (Double, obligatorio, no vacío), **energy** (Double, Obligatorio, no vacío, requerido para products con *Advance Monitoring Level*), **leakage** (Integer, Obligatorio, no vacío, requerido para products con *Advance Monitoring Level*, valores posibles 0 y 1).
- Especifica que el atributo **monitoringLevel** es de tipo *MonitoringLevel*, que es un enumeration conformado por: ESSENTIAL_MONITORING, ADVANCE_MONITORING.
- Considera que siendo **monitoringLevel** de tipo enumeration, al momento de ingresar o actualizar vía el Endpoint, el valor de **monitoringLevel** se debe recibir en el request como **String**, siendo los únicos posibles valores: “ESSENTIAL_MONITORING”, “ADVANCE_MONITORING”.
- Cuando se responde una consulta de **product** por **id**, en el response, para el caso del valor de **monitoringLevel,** se debe retornar en el response su valor como String según corresponda, sea “ESSENTIAL_MONITORING” o “ADVANCE_MONITORING”.
- No es válido recibir requests de **Snapshots** que incluyan valores correspondientes a Advance Monitoring para **Products** cuyo valor de monitoringLevel es ESSENTIAL_MONITORING. Esto debe generar un response status de 400 y el mensaje “Snapshot Data not compatible with product current Monitoring Level”.

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

/api/v1/products
/api/v1/products/{productId}/snapshots

Products Endpoint (/api/v1/products)
Debe implementar **solo dos** operaciones en el RESTful API: agregar un **product** y consultar por **id**. Los valores de **id** son autogenerados al momento de almacenar la información.

Product Snapshots Endpoint (/api/v1/products/{productId}/snapshots)
Debe implementar **solo dos** operaciones sobre los **snapshots** en el RESTful API: agregar un **snapshot** y consultar **snapshots** para un **productId**. Los valores de **id** son autogenerados al momento de almacenar la información.

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

Technical constraints

1. Elabore la solución con Java 17 / 19 / 20 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 / PostgreSQL), en un esquema **isa**.
4. Los packages de su solución deben tener como nombre raíz **com.isa.platform.u<código-estudiante>** (por ejemplo, com.*isa.platform.u201621873*).
5. Considere que el concepto **Product** pertenece al bounded context **inventory** y el concepto Snapshot pertenece al bounded context **monitoring**.
6. Aplique buenas prácticas de Arquitectura de Software, enfoque Domain-Driven, principios y patrones de diseño, 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).
7. El puerto de escucha del API en **localhost** debe ser el puerto **8080**.
8. Utilice minúsculas para los nombres de URL para todos los endpoints.
9. Utilice la biblioteca Lombok para el manejo de métodos constructores y de acceso en las clases POJO[1].
10. Utilice la biblioteca ModelMapper para el Object Mapping.
11. Incluya documentación de Endpoints con OpenAPI.
12. Considere la gestión de excepciones en la aplicación.
13. Empaquete su solución como un archivo **.zip**. (único formato válido) con el nombre ***upc-pre-202301-si729-<sección>-eb-u<código-estudiante>.zip*** (por ejemplo, *upc-pre-202301-si729-sw51-eb-u201621873.zip*).
14. Suba su archivo de solución en la Actividad indicada para el Examen final.

NO forma parte del alcance del proyecto:

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

[1] POJO: Plain Old Java Object. Se refiere a aquellas clases que en su mayoría no contienen lógica, solo atributos y métodos de acceso a los mismos (get y set).

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. Products 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 endpoints 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. Product Snapshots 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 endpoints 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, 04 de Julio del 2023

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/

ModelMapper: http://modelmapper.org/

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

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