



DESARROLLO DE APLICACIONES OPEN SOURCE (SI729)
PRÁCTICA CALIFICADA 2
2023-2

Sección: SW53
Profesores: Navarrete Vilca, Elio Jefferrson
Duración: 100 minutos
Indicaciones:

1. La práctica calificada consta de 1 pregunta, y tendrá 100 minutos para resolverla.
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-202302-si729-<sección>-pc2-u<código-estudiante>.zip*, conteniendo el proyecto de software, en la Actividad PC2 (Práctica calificada 2).

Enunciado:

Caso sportnet.com.

Su cliente, sportnet, desea contar con una plataforma para brindar soporte a los procesos relacionados con las experiencias de reserva de eventos deportivos, teniendo como segmentos a los deportes de mayor demanda en las principales ciudades del mundo.

Pregunta 1 (20 p.).

Usted forma parte del equipo de desarrollo de backend encargado de construir una RESTful API para una nueva red social enfocada en deportistas, **sportnet.com**.

El objetivo de la plataforma es conectar a personas interesadas en diversas actividades físicas y deportes. El ecosistema de sportnet.com necesita que la aplicación backend provea **Endpoints** para el manejo de información de **eventos deportivos (SportEvent)**, que consta de los atributos: eventName (String), sportType (String), organizerId (Long), location (String), capacity (Integer).

Las reglas de negocio de sportnet.com son las siguientes:

- No se debe permitir la creación de dos eventos con el mismo nombre (**eventName**) y tipo de deporte (**sportType**) en la misma ubicación (**location**).
- Cada evento debe tener un identificador único de organizador (**organizerId**) y no pueden existir dos eventos con el mismo **organizerId** en el mismo día.
- La información requerida para cada evento incluye: **id** (Long, obligatorio, autogenerated, clave primaria), **eventName** (String, obligatorio, no vacío), **sportType** (String, obligatorio, no vacío), **organizerId** (Long, obligatorio, no vacío), **location** (String, obligatorio), **capacity** (Integer, obligatorio, positivo).

Se le asigna el desarrollo de un Endpoint específico:

/api/v1/sport-events

SportEvents Endpoint (/api/v1/sport-events)

Debe implementar **solo una** operación en el RESTful API: agregar un **SportEvent** (POST). Los valores de **id** no se ingresan al momento de agregar, pues son autogenerados al momento de almacenar la información. Ante una adición satisfactoria, retorne el HTTP Status 201 y en el body del response el resource incluyendo la información del evento deportivo y su id generado. En caso de error incluya el HTTP Status correcto según el error, así como el mensaje de error en el body del response.

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

Technical constraints

1. Elabore la solución con Java 20 o 21, Spring Boot Framework 3 como development framework y Spring Data JPA como ORM.
2. Cree su proyecto de software con el nombre **si729pc2u<código-estudiante>** (por ejemplo, **si729pc2u201621873**).
3. La información debe ser persistente en una base de datos relacional (MySQL), en un esquema **sportnet**.
4. Los packages de su solución deben tener como nombre raíz **com.sportnet.platform.u<código-estudiante>** (por ejemplo, **com.sportnet.platform.u201621873**).
5. Considere que el concepto **SportEvent** pertenece al bounded context **booking**.
6. Aplique buenas prácticas de Arquitectura de Software, enfoque de Domain-Driven Design, layered architecture, 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. Considere el bounded context **shared** para elementos base comunes/reutilizables que puedan ser aprovechados o extendidos por elementos en otros bounded contexts.
8. Documente su código con **Javadoc**, colocando 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**.
9. Incluya documentación de **OpenAPI** con Swagger UI.
10. El puerto de escucha del API en **localhost** debe ser el puerto **8080**.
11. Utilice minúsculas y plural para los nombres de URL para todos los endpoints.
12. De ser necesario, puede utilizar la biblioteca Lombok para el manejo de métodos de acceso en las clases de Java.
13. Considere la gestión de excepciones en la aplicación.
14. Empaquete su solución como un archivo **.zip**. (único formato válido) con el nombre **upc-pre-202302-si729-<sección>-pc2-u<código-estudiante>.zip** (por ejemplo, **upc-pre-202302-si729-sw53-pc2-u201621873.zip**).
15. Suba su archivo de solución en la Actividad indicada para la Práctica calificada 2.

NO forma parte del alcance del proyecto:

1. Auditoría
2. Soporte de CORS.
3. Security.
4. 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. 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 indicada en el esquema indicado.	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 <i>nombrada según especificaciones</i> en base de datos relacional indicada en el esquema <i>indicado</i> .	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.	
	6.0 puntos	4.0 puntos	2.0 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.	
	3.0 punto	2.0 puntos	1.0	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, columnas 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.	
	2.0 puntos	1.0 punto	0.5 puntos	0 puntos	
Total	20 puntos	13.0 puntos	6.5 puntos	0 puntos	

Lima, 08 de noviembre 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/>

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

Spring Boot OpenAPI documentation: <https://springdoc.org/>