



**INGENIERÍA DE SOFTWARE**  
**IASI0729 - DESARROLLO DE APLICACIONES OPEN SOURCE**  
**EXAMEN FINAL**  
**202520**

**NRC:** 7369

**Profesores:** Bautista Ubillús, Efraín Ricardo  
Castro Veramendi, Rafael Oswaldo  
Espinoza Bravo, Wilder Julio  
Flores Moroco, Juan Antonio  
Mori Paiva, Hugo Allan  
Robles Fernández, Iván  
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 eb<nrc>u<código-estudiante>.zip (por ejemplo, eb7377u201621873.zip), conteniendo el proyecto de software, en la Actividad para el Examen final.
  3. Debe crear su proyecto de solución y evidenciar la autoría.
  4. Puede utilizar como referencia los materiales publicados en el aula virtual, los sitios web de documentación de frameworks, lenguaje de programación utilizados, así como ejemplos de clase (solo como referencia, no como fuente de duplicado o copia).
- 

**Enunciado:**

**Caso RoadZen**

En un futuro impulsado por la movilidad eléctrica y la conducción autónoma, la gestión y optimización de flotas de vehículos eléctricos (VE) se ha convertido en un desafío crucial. RoadZen (<https://www.roadzen.ai/>) se ha posicionado como líder en el desarrollo de plataformas de telemetría y diagnóstico inteligente para flotas de vehículos eléctricos (VE). Tras años de desarrollo en hardware y software de vehículos conectados, RoadZen lanzó su primera generación de Smart Vehicle Telemetry Modules (SVTM) —dispositivos que miden en tiempo real el estado de la batería, el rendimiento del motor, la posición GPS y el estado general de cada vehículo.

Ahora, como parte de su estrategia de expansión, RoadZen está preparando el lanzamiento de FleetSense Pro, su nueva plataforma de gestión de flotas, y requiere una API backend robusta que permita gestionar tanto los dispositivos (VE) como los registros de telemetría que generan.

## Pregunta 1 (20 puntos)

Usted se integra al backend software developer team, a cargo de la creación de un RESTful API que brinde soporte a las operaciones de la plataforma RoadZen. El ecosistema requiere que el módulo de RoadZen Vehicle & Monitoring cuente con Endpoints en el RESTful API, para el manejo de la información de: **TelemetryRecords**, conformada por los atributos `id`, `vehicleVin`, `batteryLevelPct`, `speedKmh`, `engineStatus`, `recordedAt`; y **Vehicles**, conformada por los atributos `id`, `vin`, `fleetId`, `preferredMaxSpeedKmh`.

La tabla asociada a **Vehicles** debe contener la siguiente información inicial:

id	vin	fleetId	preferredMaxSpeedKmh
1	RZ-TES-M201	101	100.0
2	RZ-NIO-X342	102	120.0
3	RZ-BYD-S550	101	90.0
4	RZ-BMW-E678	103	110.0

El valor de **engineStatus** corresponde con una enumeración **EngineStatus** con los posibles valores:

Id	Name
0	ON
1	OFF
2	ERROR

### Reglas de negocio:

- Establece que la información que se desea preservar de los **Vehicles** incluye: **id** (Long, Primary Key, Autogenerado), **vin** (Vehicle Identification Number, obligatorio, formato válido: 2 letras + guion + 3 letras + guion + 4 caracteres alfanuméricos, ej. "RZ-TES-M201"), **fleetId** (Long, obligatorio, positivo) y **preferredMaxSpeedKmh** (Double, obligatorio, entre 80.0 y 130.0).
- Establece que la información que se desea preservar de los **TelemetryRecords** incluye: **id** (Long, Primary Key, Autogenerado), **vehicleVin** (VIN, obligatorio, debe corresponder a un Vehicle existente), **batteryLevelPct** (Double, obligatorio, entre 0.0 y 100.0), **speedKmh** (Double, obligatorio,  $\geq 0.0$ ), **engineStatus** (EngineStatus, obligatorio) y **recordedAt** (LocalDateTime, obligatorio, no nulo, no puede ser posterior a la fecha actual).
- **VIN** es un *value object* que valida el formato especificado.
- La información de **Vehicles** debe poblarse automáticamente al iniciar la aplicación mediante el evento **ApplicationReadyEvent**.
- Al recibir un request, el campo **recordedAt** se proporciona como String con el formato "yyyy-MM-dd HH:mm:ss".
- Los valores de **engineStatus** se envían como String en el request.
- Al registrar un **TelemetryRecord** con datos válidos, se debe Validar que **vehicleVin** exista en Vehicles, y de Emitir un evento de integración **TelemetryRecordRegisteredEvent**, que incluya **vehicleVin** y **speedKmh**.
- El event handler de **TelemetryRecordRegisteredEvent** debe actualizar **preferredMaxSpeedKmh** del Vehicle correspondiente solo si el nuevo **speedKmh** supera el valor actual.
- **TelemetryRecord** y **Vehicle** pertenecen a bounded contexts distintos: **TelemetryRecord** es para **telemetry**, **Vehicle** es para **fleet-management**. Por tanto, se requiere un Anti-Corruption Layer (ACL) para acceder a la información del otro contexto.
- Ambas entidades son Aggregate Roots y deben incluir atributos de auditoría: **createdAt** (fecha de creación) y **updatedAt** (fecha de última actualización). Estos deben gestionarse automáticamente por el sistema y no mostrarse en los responses.

## Endpoints requeridos:

/api/v1/telemetry-records

Solo operación POST: registrar un nuevo registro. Respuesta: status 201 Created con el objeto registrado (incluyendo **id** generado), sin atributos de auditoría.

/api/v1/vehicles

Solo operación GET: listar todos los vehículos. Respuesta: lista de vehículos sin atributos de auditoría.

## Technical constraints

1. Elabore la solución con Java 25, Spring Boot Framework 3.5.8 y Spring Data JPA como ORM.
2. Nombre del proyecto: **eb7377u<código-estudiante>** (ej. eb7377u201621873).
3. Persistencia en PostgreSQL, esquema **roadzen**.
4. Paquetes raíz: **ia.roadzen.platform.u<código-estudiante>**.
5. Bounded contexts: **telemetry**: contiene TelemetryRecord y EngineStatus y **fleet-management**: contiene Vehicle.
6. Use un bounded context **shared** para elementos comunes (siguiendo el estilo del proyecto *learning center platform* visto en clase).
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. Utilice el patrón Assembler para el Object Mapping en la sección *transform* en interfaces layer.
10. 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.
11. Utilice minúsculas para los nombres de URL y términos compuestos separados por guión medio (-) para todos los endpoints.
12. Utilice la biblioteca Lombok para el manejo de métodos constructores y de acceso en las clases de Java.
13. Utilice records en vez de clases para almacenamiento de valores inmutables.
14. Para **telemetry record** y **vehicle**, incluya atributos de auditoría createdAt y updatedAt con valores poblados de forma automática por Spring Boot al momento de la creación.
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**, proporcionando información a nivel general, de las operaciones, los requests y los posibles responses.
17. Por defecto, todos los textos de mensajes deben ser en inglés.
18. El API debe soportar los mensajes de error o validación en el response en los language Inglés (EN, EN-US) o Español (ES, ES-PE), según indique el parámetro **Accept-Language** en el header del request.
19. Considere la gestión de **excepciones** en la aplicación, proporcionando Responses adecuados.
20. Incluya en el archivo **README.md**, la información en inglés de la aplicación, descripción y su información como author.
21. Empaquete su solución como un archivo **.zip**. (**único formato válido**) con el nombre **eb<NRC>u<código-estudiante>.zip** (por ejemplo, **eb7377u201621873.zip**).
22. 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
<b>C01. Building y ejecución</b>	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.	
	<b>2.0 puntos</b>	<b>1.0 punto</b>	<b>0.5</b>	<b>0 puntos</b>	
<b>C02. First Endpoint</b>	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 <i>nombrada según especificaciones</i> en base de datos relacional según enunciado 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.	
	<b>4.0 puntos</b>	<b>3.0 puntos</b>	<b>1.5 puntos</b>	<b>0 puntos</b>	
<b>C03. Second Items Endpoint</b>	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 <i>nombrada según especificaciones</i> en base de datos relacional según enunciado 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.	
	<b>4.0 puntos</b>	<b>3.0 puntos</b>	<b>1.5 puntos</b>	<b>0 puntos</b>	
<b>C04. Business Rules</b>	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.	
	<b>4.0 puntos</b>	<b>3.0 puntos</b>	<b>1.5 puntos</b>	<b>0 puntos</b>	
<b>C05. Code Organization</b>	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.	
	<b>2.0 punto</b>	<b>1.0 puntos</b>	<b>0.5</b>	<b>0 puntos</b>	
<b>C06. Code Quality</b>	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.	
	<b>3.0 puntos</b>	<b>2.0 punto</b>	<b>1.0</b>	<b>0 puntos</b>	
<b>C07. Naming Standards</b>	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.	
	<b>1.0 puntos</b>	<b>0.5 punto</b>	<b>0.25 puntos</b>	<b>0 puntos</b>	
<b>Total</b>	<b>20 puntos</b>	<b>13.5 puntos</b>	<b>6.5 puntos</b>	<b>0 puntos</b>	

Lima, 12 de diciembre del 2025

## 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/25/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/>

Get Date and Time From a Datetime String in Java

<https://www.baeldung.com/java-date-time-from-string>

Spring Data JPA Auditing:

<https://docs.spring.io/spring-data/jpa/reference/auditing.html#auditing.annotations>

How to validate MAC address using Regular Expression

<https://www.geeksforgeeks.org/dsa/how-to-validate-mac-address-using-regular-expression/>