Upc-pre-ea - examen parcial INGENIERÍA DE SOFTWARE

Ingeniería de Software (Universidad Peruana de Ciencias Aplicadas)

CC236 – INGENIERÍA DE SOFTWARE
EXAMEN PARCIAL
2024-2

Profesor: **Mendoza Puerta, Henry Antonio**
Hora: **11: 00 AM - 13:50 PM**
Duración: **170 minutos**

Indicaciones

1. Duración: El examen consta de 5 preguntas y tendrá 170 minutos para resolverlo.
2. Entrega: Debe subir y enviar su archivo solución 15 o 10 minutos antes de las 13:50 PM, que es cuando se cierra la actividad de entrega del examen parcial. Es su responsabilidad cumplir con la entrega a tiempo. No se revisarán archivos enviados fuera de tiempo.

3. **Proyecto base y endpoints en Postman:** Para iniciar con su desarrollo, debe descargar tanto el proyecto base como el archivo JSON de Postman que contiene todos los endpoints necesarios para el examen. Ambos archivos pueden ser descargados desde el siguiente enlace: Enlace de descarga del proyecto base y endpoints en Postman.

Instrucciones para la Entrega del Examen Parcial

Sigue estos pasos 15 o 10 minutos antes de las 13:50 PM para asegurar una entrega correcta:

1. Revisa tu código: Verifica que tu proyecto esté completo y que sea el archivo correcto.
2. Comprime tu proyecto: Asegúrate de comprimir todo tu proyecto en formato zip o rar.
3. Renombra el archivo: Usa la siguiente nomenclatura para nombrar tu archivo: upc_pre_XXXX, donde XXXX corresponde a tu código de estudiante. Ejemplo: upc_pre_20234567.
4. Sube el archivo: Ve a la Actividad de Entrega de Examen Parcial, que se encuentra en la sección Examen Parcial del aula virtual.
5. Adjunta tu archivo y envíalo: Asegúrate de hacer clic en enviar para que se registre tu entrega correctamente.

Rúbrica de Calificación

Criterios de Calificación	Sobresaliente	Necesita Mejora	Deficiente
Implementación del registro de membresía	Cumple con todas las reglas de negocio y validaciones solicitadas, y se evidencia el correcto funcionamiento mediante Postman.	Cumple parcialmente con las reglas de negocio y validaciones solicitadas. Se evidencia el funcionamiento mediante Postman.	No cumple con las reglas de negocio o no se evidencia el funcionamiento mediante Postman.
	5 ptos	2.5 ptos	0 ptos
Implementación del registro de visita	Cumple con las reglas de negocio, verifica límites según el tipo de membresía, y se lanza la excepción adecuada si se excede el límite de visitas.	Cumple parcialmente las reglas de negocio. No se verifican completamente los límites de visitas o las excepciones no funcionan correctamente	No verifica los límites de visitas o no se lanza la excepción correspondiente. O no se evidencia funcionamiento mediante Postman.
	5 ptos	2.5 ptos	0 ptos
Listar membresías por tipo	Cumple con la implementación y permite filtrar correctamente las membresías activas por tipo, mostrando los detalles necesarios.	Cumple parcialmente con la implementación, pero la filtración o los detalles mostrados no son correctos o completos.	No cumple con la implementación o no permite filtrar membresías por tipo. O no se evidencia funcionamiento mediante Postman.
	3 ptos	1.5 ptos	0 ptos
Generar reporte de estadísticas	Genera correctamente el reporte de estadísticas usando una función de PostgreSQL y se evidencia el funcionamiento mediante Postman.	El reporte se genera parcialmente o no incluye todas las estadísticas requeridas. Se evidencia el funcionamiento mediante Postman.	No genera el reporte correctamente o no incluye los datos solicitados. O no se evidencia funcionamiento mediante Postman
	4 ptos	2 ptos	0 ptos
Eliminar una membresía	Permite eliminar una membresía correctamente utilizando el ID. Si la membresía no existe, lanza una excepción indicando el error.	Cumple parcialmente con la eliminación de la membresía. Las excepciones no se manejan correctamente o el funcionamiento no es consistente.	No permite eliminar una membresía correctamente o no lanza la excepción adecuada cuando el ID no existe. O no se evidencia funcionamiento mediante Postman.
	3 ptos	1.5 ptos	0 ptos

Enunciado (20 ptos)

Se le ha asignado el desarrollo de un sistema para gestionar las membresías en un gimnasio. Este sistema permitirá a los administradores del gimnasio registrar nuevas membresías, llevar un control de las visitas de los miembros, y generar reportes de estadísticas. La implementación será mediante una API REST, utilizando **Spring Boot**, **Spring Data JPA**, **PostgreSQL,Validation, ModelMapper y Lombok** para la gestión de datos. Además, se espera que el sistema gestione diferentes tipos de membresías, como **Básica**, **Premium**, y **VIP**, cada una con sus propias características en cuanto a los límites de visitas mensuales.

Su objetivo es implementar un sistema funcional que permita a los administradores del gimnasio realizar las siguientes acciones: registrar nuevas membresías, actualizar el número de visitas de los miembros, listar membresías por tipo, generar un reporte de estadísticas y eliminar membresías. Se proporcionarán a continuación los requisitos funcionales detallados.

Detalle de Clase y Tabla

A continuación, se proporciona el detalle de la clase Membership que será utilizada para representar las membresías de los miembros del gimnasio. También se describe la estructura de la tabla memberships en la base de datos.

Atributo	Tipo de Dato	Columna	Descripción
id	Long	id	Identificador único de la membresía. Se genera automáticamente.
memberName	String	member_name	Nombre del miembro asociado a la membresía.
membershipType	MembershipType	membership_type	Tipo de membresía: **Básica**, **Premium**, o **VIP**. Se enumera como cadena de texto. (Enum)..
monthsSubscribed	int	months_subscribed	Número total de meses contratados por el miembro
monthsRemaining	int	months_remaining	Número de meses restantes de la membresía.
membershipPrice	Double	membership_price	Precio total de la membresía.
monthlyVisits	int	monthly_visits	Número de visitas mensuales realizadas por el miembro. Inicialmente se establece en 0. No nulo.
description	String	description	Descripción de la membresía. No debe exceder los 255 caracteres. No nulo.

Endpoints

El base_url para todos los endpoints será: http://localhost:8080/api/v1.

Método HTTP	Endpoint	Descripción
POST	/memberships	Registrar una nueva membresía
PUT	/memberships/{membershipId}/visit	Registrar una visita de un miembro
GET	/memberships?membershipType={type}	Listar membresías por tipo
GET	/memberships/statistics	Generar reporte de estadísticas
DELETE	/memberships/{membershipId}	Eliminar una membresía por su ID

Requisitos Funcionales

1. **Pregunta 1: Registrar una nueva membresía**
   Como administrador del gimnasio, **Quiero** registrar una nueva membresía para un miembro,
   Para gestionar su acceso al gimnasio y mantener un control sobre su plan de visitas.

   Criterios de Aceptación:
   - Se debe permitir el registro de una membresía con los siguientes datos: nombre del miembro, tipo de membresía (Básica, Premium, VIP), cantidad de meses suscritos y precio.
   - No se debe permitir registrar una membresía si ya existe una con el mismo nombre y tipo de membresía.
   - El sistema debe almacenar el número de meses suscritos y las visitas mensuales iniciales deben ser 0.

   Resultado Esperado:
   La membresía se registra correctamente y está disponible para futuras consultas.

Ejemplo de JSON Request	Ejemplo de JSON Response

{
"memberName": "John Doe",
"membershipType": "BASICA",
"monthsSubscribed": 12,
"membershipPrice": 300.00,
"description": "Membresía básica para acceso a todas las áreas comunes."
}
	{
	"id": 1,
	"memberName": "John Doe",
	"membershipType": "BASICA",
	"monthsSubscribed": 12,
	"monthsRemaining": 12,
	"membershipPrice": 300.00,
	"monthlyVisits": 0,
	"description": "Membresía básica para acceso a todas las áreas comunes."
	}

Ejemplo de JSON Response (Error - Membresía Duplicada)

{
"timestamp": "2024-09-29T15:10:05.234",
"message": "Ya existe una membresía con el mismo nombre y tipo.",
"details": "Error: Membresía duplicada"
}

2. **Pregunta 2: Registrar la visita de un miembro**
   Como miembro del gimnasio, **Quiero** registrar mi visita cuando accedo al gimnasio,
   Para llevar un control de las visitas mensuales que he realizado.

   Criterios de Aceptación:
   - Se debe incrementar el contador de visitas mensuales cada vez que un miembro ingresa al gimnasio.
   - Si el miembro tiene una membresía Básica o Premium, se debe validar que no haya excedido el límite de visitas (10 visitas para Básica y 20 visitas para Premium).
   - Los miembros con membresía VIP pueden realizar visitas ilimitadas.
   - Si el miembro excede su límite de visitas, el sistema debe lanzar una excepción indicando el error.

   Resultado Esperado:
   La visita se registra correctamente y, si se excede el límite, se notifica el error al miembro.

Ejemplo de JSON Response (Respuesta Exitosa)	Ejemplo de JSON Response (Error de límite excedido)

{
"id": 1,
"memberName": "John Doe",
"membershipType": "BASICA",
"monthsSubscribed": 12,
"monthsRemaining": 10,
"membershipPrice": 300.00,
"monthlyVisits": 1,
"description": "Membresía básica para acceso a todas las áreas comunes."
}
	{
	"timestamp": "2024-09-29T15:00:00",
	"message": "Has alcanzado el límite de 10 visitas mensuales para la membresía Básica.",
	"details": "VisitLimitExceededException"
	}

3. **Pregunta 3: Listar membresías por tipo**
   Como administrador del gimnasio,**Quiero** listar todas las membresías según el tipo de membresía,
   Para tener una visión clara de los miembros activos según el tipo de plan que tienen.

   Criterios de Aceptación:
   - El sistema debe permitir listar todas las membresías activas filtradas por el tipo de membresía (Básica, Premium, VIP).
   - Se deben mostrar los detalles del miembro, el número de visitas mensuales y los meses restantes.

   Resultado Esperado:
   Se genera una lista de todas las membresías activas según el tipo de membresía solicitado.

Ejemplo de JSON Response (Respuesta Exitosa)


```
[
  {
    "id": 1,
    "memberName": "John Doe",
    "membershipType": "BASICA",
    "monthsSubscribed": 12,
    "monthsRemaining": 8,
    "membershipPrice": 300.00,
    "monthlyVisits": 1,
    "description": "Membresía básica para acceso a todas las áreas comunes."
  },
  {
    "id": 2,
    "memberName": "Jane Smith",
    "membershipType": "BASICA",
    "monthsSubscribed": 6,
    "monthsRemaining": 4,
    "membershipPrice": 150.00,
    "monthlyVisits": 4,
    "description": "Membresía básica con acceso limitado a clases grupales."
  }
]
```


4. **Pregunta 4: Generar reporte de estadísticas**
   Como administrador del gimnasio, **Quiero** generar un reporte de estadísticas de las membresías,
   Para analizar el rendimiento del gimnasio en términos de número de miembros y sus visitas mensuales.

   Criterios de Aceptación:
   - El sistema debe generar un reporte que incluya el número total de miembros por tipo de membresía, así como el total de visitas mensuales para cada tipo de membresía.
   - El reporte debe generarse utilizando una función de PostgreSQL para consolidar los datos de las membresías.

   Resultado Esperado:
   El reporte se genera correctamente y muestra los totales y promedios por tipo de membresía.

Ejemplo de JSON Response (Respuesta Exitosa)


```
[
  {
    "membershipType": "BASICA",
    "totalMembers": 25,
    "totalMonthlyVisits": 10
  },
  {
    "membershipType": "PREMIUM",
    "totalMembers": 15,
    "totalMonthlyVisits": 20
  },
  {
    "membershipType": "VIP",
    "totalMembers": 10,
    "totalMonthlyVisits": 150
  }
]
```

5. **Pregunta 5: Eliminar una membresía**
   Como administrador del gimnasio, **Quiero** poder eliminar una membresía específica,
   Para eliminar los registros de miembros que ya no están activos en el gimnasio.

   Criterios de Aceptación:
   - El sistema debe permitir eliminar una membresía utilizando el ID de la membresía.
   - Si el ID no existe, se debe lanzar una excepción informando que la membresía no fue encontrada.

   Resultado Esperado:
   - La membresía se elimina correctamente del sistema o, en caso de error, se muestra un mensaje indicando que no existe.

Ejemplo de JSON Response (Respuesta Exitosa)
HTTP/1.1 204 No Content

Ejemplo de JSON Response (Cuando la membresía no existe)

{
"timestamp": "2024-09-29T14:20:01.123",
"message": "No se encontró la membresía con ID 5.",
"details": "Error: Membresía no encontrada"
}