Clase 5 - DA2 Teórico

13/09/2021

```
{
    data: { object},
    code: 2005,
    success: true,
}
```

(15 puntos)

WebApi y Angular

Debido a la situación sanitaria el Club Deportivo Bartola se encuentra realizando un sistema que le permitirá a sus deportistas de diferentes disciplinas reservar las instalaciones del club mediante una aplicación web. Esta debe permitir las siguientes operaciones:

1. Como basquetbolista quiero poder ver todas las instalaciones disponibles para mi disciplina.
2. Como futbolista quiero poder reservar una de las canchas de fútbol del club.
3. Como basquetbolista quiero poder modificar una de mis reservas.
4. Como futbolista quiero poder cancelar una de mis reservas reserva.
5. Cómo basquebolista quiero poder ver mis reservas en un rango de fechas.

Nota: tenga en cuenta que las instalaciones se muestran dependiendo la disciplina del usuario de la aplicación. Es decir, si un basquetbolista solicita ver las instalaciones disponibles, solo va a poder ver las disponibles para los basquetbolistas.

a) **(8 puntos)** Realice un diseño de cada uno de los endpoints que utilizaría para resolver el problema planteado, considerando: URIs, recursos, verbos HTTP, headers, datos de entrada y/o salida, y los correspondientes códigos de error de estos.

b) **(7 puntos)** Dada la operación 4, diseñe un servicio de angular para este endpoint.

Verbo:
URI: usuario/[id]?clave=[valor]
Query Params:
Route Params:
Headers:
Body:
Responses:

1)
Verbo: GET
URI-1: /basquetbol/intalaciones
URI-2: /instalaciones
Headers: token
Body: -
Responses:

- 200: retornar lista de instalaciones
  - [{ “idInstalacion”: 1, “nombreInstalacion”: “Cancha1”}]
  - IdInstalacion: int
  - NombreIntalacion: string
- 404: no hay instalaciones disponibles
- 401: indica que no envió token de autenticación
- 403: cuando quiero listar las instalaciones de una disciplina que no es la mia

401: Faltan credenciales
403: No estoy autorizado

2)
Verbo: POST
URI-1: /instalaciones/[idInstalacion]/reservas
URI-2: /reservas
Headers: token
Body: { “idInstalacion”: 3, “fecha”: “13/09/2021”, “hora”: “11:00” }

- IdInstalacion: int
- Fecha: string
- Hora: string

Responses:

- 200: retornar el id de la reserva (o también el objeto de la reserva que mando en el body)
  - { “idReserva”: 1, “fecha”.... }
    - IdReserva: int
    - IdInstalacion: int
    - Fecha: string
    - Hora: string;
- 404: La instalación existe
- 401: indica que no envió token de autenticación

- 403: No puedo hacer reservas a instalaciones que no son de mi disciplina
- 500: Ocurrió un error inesperado

3)
Verbo: PUT/PATCH
URI-1: /instalaciones/[idInstalacion]/reservas/[idReserva]
URI-2: /reservas/[idReserva]
URI-3 /reservas
Headers: token
Body: { “idReserva”: 1, “idInstalacion”: 3, “fecha”: “13/09/2021”, “hora”: “11:00” }
    - IdInstalacion: int
    - Fecha: string
    - Hora: string
    - IdReserva: int

En el caso del PATCH, en el body podría recibir solamente lo que quiero modificar:
Body: { “idRreserva”: 5, “fecha”: “15/09/2021” }
Responses:
- 200: Reserva exitosa
- 401: indica que no envió token de autenticación
- 403: la reserva no es mia
- 404: si la reserva no existe

4)
Verbo: DELETE
URI-1: /instalaciones/[idInstalacion]/reservas/[idReserva]
URI-2: /reservas/[idReserva]
Headers: token
Body: -
Responses:
- 200: reserva eliminada con éxito
- 401: indica que no envió token de autenticación
- 403: la reserva no es mia
- 404: si la reserva no existe

5)
Verbo: GET

URI: /reservas?desde=13/09/2021&hasta25/09/2021
Query Params:
- desde: string
- hasta: string

Headers: token
Body: -
Responses:
- 200: Lista las reservas en el rango de fechas
- 404: SI no hay reservas en ese rango de fechas
- 401: indica que no envió token de autenticación

Tabla de Body
Nombre - Tipo - Descripción - Requerido

Maneja de sesiones:

Tabla sesiones:
Columnas:
- Identificador de usuario
- Token (clave única que lo identifica)

sesiones

idUsuario	Token autogenerado
1	token1
4	token2

Tabla Usuarios:
IdUsuario
Nombre de usuario
Contraseña

———————————————————————————

Modelo 4 + 1

Cuando queremos entender un sistema o una arquitectura grande y/o complejo. Es interesante poder dividir este sistema en un conjunto de partes diferentes, pero relaciones. El modelo 4 + 1 nos permite organizar, entender y documentar la arquitectura de un sistema siguiendo la linea de dividirla en partes más pequeñas (denominadas “vistas”).

Las vistas:
- Vista lógica o de diseño
- Vista de implementación o vista de desarrollo o vista de componentes
- Vista de procesos
- Vista fisica o de despliegue
- Y la vista +1, es la vista de los casos de uso, que es la que une las 4 vista anteriores. Muestra la funcionalidad del sistema desde el punto de vista del usuario final.

Vista lógica
Describen cómo se soportan los requerimientos funcionales del sistema, enfocado en el diseño y la colaboración entre sus elementos.
- Diagramas de clase
- Diagramas de objetos (nuevo)
- Diagramas de estructura compuesta (nuevo)
- Diagramas de estado (nuevo)
- Diagramas de interacción

Vista de implementación (vista de desarrollo o vista de componentes)
Muestra el sistema desde una perspectiva del programador. esta encfocada en las diferentes artefactos o componentes del software.
- Diagramas de componentes (nuevo)
- Diagramas de paquetes

Vista de procesos
Describe aspectos de sincronización y concurrencia . Nos muestran los procesos dinámicos que conectan las diferentes partes del sistema

- Diagrama de actividad

Vista de fisica o de despliegue
Describir la relación entre el software y hardware. Enfocada en Infraestructura .

- Diagrama de despliegue

Vista casos de uso
Describe el comportamiento del sistema, basado en los principales casos de uso visto desde la perspectiva de los usuarios, analistas, etc.