Taller de Programación 2

Examen Final llamado Julio (16 de Julio, 2025)

Enunciado

Se desea realizar un sistema que reciba y procese coordenadas de posición que envían aeronaves cercanas a la torre de control. Los datos del avión se componen del número de vuelo: con tres letras y tres números (ej. AAB001) y los valores de posición (xa, ya, za) en metros respecto a la torre de control que se considera punto de origen (xt=0, yt=0, zt=0), donde z representa la coordenada de altura. El objetivo del sistema es poder almacenar los datos recibidos y si corresponden al mismo vuelo actualizar sólo la posición. Además se pide implementar un sistema de alerta para aviones que estén a menos de 500 metros uno del otro. La fórmula para el cálculo de distancia será la siguiente:

$d = \sqrt{(xa1 - xa2)^2 + (ya1 - ya2)^2 + (za1 - za2)^2}$ donde xa1, ya1 y za1 son las coordenadas de un avión y xa2, ya2 y za2 las de otro.

Para cumplir con esta implementación necesitamos que la aplicación backend pueda:

- Almacenar/actualizar y listar los datos recibidos de las aeronaves.
- Al momento del ingreso de una nueva coordenada revisar distancias con respecto a todo lo almacenado para dar alerta.

1. Implementar los endpoints que permitan:
    - **a.** Ingresar y persistir los datos de un avión. El formato enviado será:

      {

      id: "AAB001", // id del número de vuelo
      xa: 7500, // en metros (coordenada x)
      ya: 6200, // en metros (coordenada y)
      za: 1000 // en metros (coordenada z)

      }

      Realizar las validaciones necesarias para recibir un valor alfanumérico de 6 dígitos en id y valores numéricos en las coordenadas. En caso de NO cumplir con los valores estipulados para cada campo se debe retornar el mensaje: "datos no válidos" en el formato que se pide más abajo.

    - **b.** Listar los datos completos de todos los vuelos.

    - **c.** Al momento del ingreso del vuelo, si este es correcto, realizar la comprobación de distancia respecto a los demás almacenados, retornando en caso de peligro de colisión un array con los ids de vuelo de todos los aviones comprometidos, caso contrario devolver un array vacío. Verificar que para realizar el cálculo haya más de un vuelo ingresado y en caso de estar actualizando uno existente no me marque colisión con el mismo. Para cualquier caso, los datos del vuelo tienen que ser persistidos.

En caso de ser necesario, el servidor recibirá desde el cliente los datos requeridos en formato JSON. En caso de inconvenientes, el servidor responderá con un objeto con un campo **'errorMsg'** informando el motivo de la falla. Todas las respuestas deberán estar correctamente adosadas con su código de estado correspondiente, según el resultado de la operación. En el caso de realizar notificaciones, modularizar adecuadamente. La notificación en sí puede simularse con un mensaje por terminal.

Aclaraciones sobre el desarrollo esperado:

1. El proyecto debe incluir únicamente el backend del sistema, utilizando Node.js + express. El formato del servidor es de tipo RESTful. Tener en cuenta los lineamientos de dicho formato, especialmente a la hora de elegir los nombres de las rutas de acceso al sistema, los verbos que accionan sobre ellas, y cómo pasar datos adicionales a la consulta.
2. El sistema debe estar correctamente separado en capas y componentes, y esta separación debe estar claramente puesta de manifiesto en la estructura de carpetas y archivos. Entre los componentes que esperamos que estén presentes encontramos: router/controlador, casos de uso, modelo/s, DAO/s, repositorios, servicios de terceros, factories, commands (los que correspondan de acuerdo al sistema modelado y se hayan visto en cursada).
3. La *validación de datos* es una parte importante del negocio, por lo tanto, observar cómo y dónde realizarla.
4. *No es necesario utilizar una conexión a base de datos* real; es posible persistir en el DAO/Repositorio usando memoria del servidor.
5. Recordar el rol de las factories, que nos permiten desacoplar las dependencias de nuestros componentes a la hora necesitar una instancia de los mismos. Recordar esto especialmente a la hora de decidir cómo obtener los casos de uso para invocarlos desde la capa de ruteo.
6. Pueden reutilizar código de proyectos realizados durante el cuatrimestre, siempre y cuando el código se utilice y realmente aporte al desarrollo de las funcionalidades pedidas. **La inclusión de código innecesario o fuera de lugar será penalizada**.