Taller de Programación 2

Examen Final (10 de febrero, 2025)

Enunciado

Realizar un pequeño sistema que permita almacenar y consultar las temperaturas registradas en un centro de investigaciones en el polo sur. Debe poder satisfacer los siguientes casos de uso:

1.
    a. Carga de una temperatura, la cual incluye: magnitud (numérico), y unidad (puede ser “celsius”, “farenheit”, o “kelvin”).
    b. En caso de registrar una temperatura menor a: 0 kelvin ó -273 celsius ó -460 farenheit, informar al administrador del servicio enviando un mail a: ***administrador@gmail.com*** ya que esto sería un hecho de suma importancia para la comunidad científica! (no hace falta implementar la funcionalidad real de envío de email, sólo dejar la clase planteada con los métodos correspondientes y la llamada en donde corresponda).
2. Listado de temperaturas registradas, que recibe un rango (min, max) y devuelve un colección con todas las temperaturas registradas que se encuentren incluídas en dicho rango.

El servidor recibirá y responderá desde y hacia el frontend con los datos requeridos en formato JSON. En el caso del punto 1, la respuesta será la nueva temperatura cargada, mientras que en el punto 2, será una lista de temperaturas. En caso de suceder algún inconveniente, se espera que el servidor responda 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.

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 que propone esta propuesta, especialmente a la hora de elegir las rutas de acceso al sistema.
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, servicio de envío de mails, factories, y cualquier otro patrón de diseño visto que colaboren con el modelado del sistema.
3. Prestar atención al sentido de las dependencias entre los componentes, recordando que las capas más cercanas al negocio no deben estar acopladas a las capas más externas (usualmente de infraestructura). Con esto en mente, *importar módulos o inyectar dependencias según corresponda*.
4. La *validación de datos* es una parte importante del negocio, por lo tanto, observar cómo y dónde realizarla.
5. *No es necesario utilizar una conexión a base de datos* real, trabajar con un DAO/repositorio, persistiendo en la memoria del servidor.
6. Recordar el rol de las factories, que nos permiten desacoplarnos de 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.