Taller de Programación 2

Examen Final (14 de diciembre, 2020)

Enunciado

Realizar un pequeño sistema que permita cargar y consultar gastos hechos en nuestra empresa. Debe poder satisfacer los siguientes casos de uso:

1. Carga de un gasto, la cual incluye: monto del gasto (siempre mayor a cero), descripción (128 caracteres), día, mes, y año en que se efectuó el mismo. En caso de recibir un gasto con un monto estrictamente superior a los $10.000, el mismo se deberá informar a la administración, enviando un mail a: ***gastos@gmail.com*** .
2. Listado de gastos por fecha, que recibe mes/año desde y mes/año hasta, y devuelve una colección con todos los gastos registrados durante ese período. El período debe ser de a lo sumo 12 meses. En caso contrario, la consulta se considera inválida.

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 el nuevo gasto cargado, mientras que en el punto dos, será una lista de gastos. 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, y factories (los que correspondan de acuerdo al sistema modelado).
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, persistir en el DAO usando memoria ram 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.