Taller de Programación 2

Examen Final (19 de diciembre, 2022)

Enunciado

Se desea realizar un sistema de control de flujo de personas en un aeropuerto muy concurrido, en la etapa de control se seguridad, por la que deben pasar todos los pasajeros. Usualmente se producen colas de muchas personas, con grandes demoras. Para llevar a cabo el control, se escanea el pasaporte y tarjeta de embarque en 3 puntos: cuando el pasajero recién se suma a la fila, en un punto intermedio y cuando está próximo al puesto de seguridad.

1. Implementar los endpoints de los puntos de control, considerando que el agente que escanea la documentación “entrega” al sistema (mediante un dispositivo con conexión http):
    a. nombre, apellido (cadenas de caracteres, no nulas) y pasaporte (numérico, no nulo)
    b. fecha de nacimiento
    c. hora del vuelo
2. En el momento de incorporación a la fila, el sistema debe verificar con un servicio externo si el pasaporte está observado como irregular o no, para poder notificar la situación. La URL externa es: https://aeropuerto.deno.dev/verif?num=<nro pasaporte>.
   Formato de la respuesta:
   {
   num: "12345678",
   irregular: true
   }
3. Si en algún punto de escaneo el vuelo está próximo a salir (1h), se debe disparar una notificación a un sistema externo (puede estar representado por/encapsulado en una clase), para que esa persona tenga prioridad.

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. Considerar la inclusión de algún test funcional (usando axios, por ejemplo) para verificar el correcto funcionamiento de lo que se está desarrollando.

7. 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**.