Taller de Programación 2

Examen Final (4 de diciembre, 2023)

Enunciado

Realizar un pequeño sistema que permita almacenar y consultar los resultados de una elección a nivel general y local correspondientes a cuatro distritos entre 3 candidatos conocidos. Debe poder satisfacer los siguientes casos de uso:

1.
    a. Carga de un voto, el cual incluye: distrito (llamados: “zona1”,”zona2”,”zona3”, “zona4”) y candidato (con opciones: “candidatoA”, candidatoB”, “candidatoC” ó “enblanco”). Realizarlo con el endpoint correspondiente que no exponga la información cargada.

    En caso de realizar la carga con datos consistentes la respuesta será un mensaje de texto: “voto cargado”. En caso contrario: “zona no correspondiente” ó “candidato no válido” ó ambos mensajes concatenados.

    b. Obtención de los votos por zona correspondientes a cada opción de candidato.

    El formato de respuesta será: { zonaX: { candidatoA: xx, candidatoB: yy, candidatoC: zz, enblanco: ww } }

    c. Obtención de los votos generales con los mismos lineamientos del punto b.

    El formato de respuesta será: { generales: { candidatoA: xx, candidatoB: yy, candidatoC: zz, enblanco: ww } }

2. Listado del resultado de las elecciones generales expresado en porcentaje para cada candidato, considerando que los votos en blanco se le asignan al candidato que más votos obtuvo. Devolver una colección con todos los candidatos y su resultado en el comicio.

    Ej: [{ candidatoA: x% }, { candidatoB: x% }, { candidatoC: x% }]

El servidor recibirá y responderá desde y hacia el frontend con los datos requeridos en formato JSON. 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.