CHECKLIST

para corregir el proyecto Peli vs. Peli

Para que un proyecto apruebe, tiene que cumplir con todas las condiciones del checklist correspondiente.

Condiciones para entregar _

- Archivos mínimos entregados: En la carpeta entregada se incluye el script con la estructura de las tablas competencia y voto (no necesariamente deben llamarse así), el package.json con las dependencias utilizadas, el archivo servidor.js con las rutas definidas, al menos un controlador con las funcionalidades del proyecto y el archivo que contiene la conexión con la base de datos.

- Rutas definidas: Revisar que el archivo servidor.js tenga definidas las rutas indicadas en la consigna y que las funciones que se ejecutan al llamarlas se encuentren al menos definidas y con algún contenido en el controlador.

Condiciones para aprobar _

- Ejecución sin errores: El código del servidor se ejecuta sin errores al correrlo desde la consola (node servidor.js).

- Estructura de la base de datos: El script SQL entregado se ejecuta sin errores y crea correctamente las entidades competencia y voto.

- Validación al crear una competencia: Para crear una competencia se valida que existan al menos dos competencias con los filtros.

- Funcionalidad de mostrar una competencia: La ruta /competencia/:id/peliculas obtiene dos películas aleatorias teniendo en cuenta todos los filtros de la competencia (actores, directores, género).

- Se implementan todos los endpoints: Todos los endpoints deben estar implementados y funcionar correctamente con el frontend:
    - GET /generos
    - GET /directores
    - GET /actores
    - GET /competencia/:id/peliculas
    - POST /competencia/:id/voto
    - GET /competencia/:id/resultados
    - GET /competencia/:id
    - GET /competencias
    - POST /competencia
    - PUT /competencia/:id
    - DELETE /competencia/:id/votos
    - DELETE /competencia/:id)

- Conexión del backend con la base de datos: Se utiliza correctamente la configuración para conectar el backend con la base de datos.

- Códigos de error: Retorna los códigos 404, 422 y 500 en forma correcta ante cada error (inexistencia del recurso, falla en una regla de negocio y error del servidor inesperado, respectivamente).

- Dependencias utilizadas: El archivo package.json contiene todas las dependencias utilizadas en el proyecto.

- Ejecución correcta de consultas: Las consultas SQL se ejecutan correctamente y devuelven los resultados esperados.