DESAFIO BACK-END AYI

Elementos a incluir

- Servidor web
- Servidor de base de datos
- Front vanillaJS/React (opcional)
- API REST

Criterios Base de Datos :

La base de datos debe poder tener al menos 3 colecciones pudiendo sumar más si es necesario y las mismas pueden tener validaciones pero es opcional :

- Usuarios
- 2x Recursos a definir por contexto

Criterios Back End :

El servidor debe poder manejar varias rutas específicas desde cualquier cliente pero cualquier pedido enviado debe poder responder con una respuesta customizada. Las rutas a incluir son :

- **/login :**
    - descripción : recibe la información del usuario, valida contra la base de datos y envía una respuesta positiva junto con una cookie que establezca el correcto ingreso
    - metodo : POST
    - parámetros : nombre de usuario / email - contraseña

- **/signup:**
    - descripción : recibe la información del usuario, valida contra la base de datos y guarda un nuevo usuario en la misma. Envía una respuesta positiva junto con el id del nuevo usuario generado
    - metodo : POST
    - parámetros : nombre de usuario - email - contraseña

- **/logout:**
    - descripción : recibe la solicitud en la ruta y envía la destrucción de la cookie

del token en el front
    - metodo : POST
    - parámetros : ninguno

- **/recurso:**
    - descripción : cada uno debe poder tener un ABML estándar
    - metodos : GET - POST - PUT - DELETE
    - parametros : string - number - boolean

- Los recursos deben estar interconectados entre sí de alguna manera, por ejemplo con una relación simple de uno a muchos como :
    - ➢ - productos -> ventas
    - ➢ - usuarios -> mensajes
    - ➢ - usuarios -> pedidos
    - ➢ - alumnos -> exámenes
- Por lo menos una de las rutas para pedir uno de los recursos tiene que poder devolver ambos registros unidos (Ej.: El detalle de un producto con todas las ventas que tuvo, la información de un usuario junto con los mensajes que recibe, etc)
- Por lo menos una de las rutas para editar/crear un aspecto del recurso tiene que poder modificar ambos registros (Ej.: Si se crea una nueva venta, que se guarde el documento en la colección de ventas y además se aumente algún campo como “ordenes_totales” en el documento del producto de la colección productos)
- Pueden existir más recursos de ser necesarios pero estos no tienen que tener si o si una API REST, es decir no se tienen que implementar en ningun front
- De no estar implementado el Front End , las rutas tienen que poder alternativamente devolver json y html por lo menos en la lectura total de toda una colección
- Si se implementa un Front End , una de las rutas tiene que poder ser parametrizada por URL
- Todas las rutas tienen que poder devolver un mismo patrón de respuesta, independientemente de si fue exitoso el pedido o no (Ej.: Si para una respuesta envió un objeto como {error:false,message:””,data:{}} entonces en todas las respuestas debería poder enviar una respuesta similar)
- Todos los pedidos al servidor tienen que poder devolver una respuesta que no haya sido creada por express
- Los recursos que intenten ser modificados de alguna manera bajo los métodos post/put/patch/delete deben pasar a través del proceso de autenticación, es decir que primero obligatoriamente se tiene que pegar en la ruta de login para obtener la información necesaria para realizar los próximos pedidos (puede ser por cookie, por token, por algún sistema customizado por ustedes, etc)

Criterios Front End :

El front es totalmente opcional y si se llegara a implementar puede estar en html/js vanilla o bien se puede implementar un entorno para React con Webpack/Babel o usando create-react-app y un proxy