Informe de Tests Unitarios al back

Resumen

Este informe proporciona un análisis detallado de los tests unitarios implementados en el código proporcionado. Los tests están diseñados para verificar la serialización y deserialización de dos clases, `TokenResponse` y `UserResponse`, así como para verificar la correcta construcción de la clase `User` utilizando un `UserRepresentation`.

1. `TokenResponse` Tests

a. `testSerialization`

Descripción: Este test verifica que la serialización de un objeto `TokenResponse` no incluya ciertas propiedades en el JSON resultante.

Verificaciones:

- Se espera que el JSON resultante contenga la propiedad `"access_token":"testAccessToken"`.
- Se espera que el JSON resultante no contenga las propiedades: `"expires_in":"3600"`, `"refresh_expires_in":"7200"`, `"refresh_token":"testRefreshToken"`, `"token_type":"Bearer"`, `"not_before_policy":"0"`, `"session_state":"sessionState"`, `"scope":"read write"`.

Resultado: El test parece estar diseñado incorrectamente, ya que las verificaciones de que ciertas propiedades no estén presentes parecen no alinearse con la expectativa general de la serialización completa de un objeto. Es probable que se haya omitido alguna configuración de `ObjectMapper`.

b. `testDeserialization`

Descripción: Este test verifica que la deserialización de un JSON a un objeto `TokenResponse` establezca correctamente las propiedades.

Verificaciones:

- Se espera que la propiedad `accessToken` del objeto resultante sea `"testAccessToken"`.
- Se espera que las demás propiedades sean `null`.

Resultado: El test está correctamente diseñado y verifica que la deserialización maneje adecuadamente la falta de ciertas propiedades en el JSON.

2. `UserResponse` Tests

a. `testSerialization`

Descripción: Este test verifica la correcta serialización de un objeto `UserResponse` a JSON.

Verificaciones:

- Se espera que el JSON resultante contenga las propiedades con los valores esperados: `"id":"123"`, `"first_name":"John"`, `"last_name":"Doe"`, `"email":"john.doe@example.com"`, `"username":"johndoe"`.

Resultado: El test está correctamente diseñado y asegura que la serialización del objeto `UserResponse` se realice según lo esperado.

b. `testDeserialization`

Descripción: Este test verifica la correcta deserialización de un JSON a un objeto `UserResponse`.

Verificaciones:

- Se espera que las propiedades del objeto resultante sean: `id = "123"`, `firstName = "John"`, `lastName = "Doe"`, `email = "john.doe@example.com"`, `username = "johndoe"`.

Resultado: El test está correctamente diseñado y asegura que la deserialización se realice correctamente.

c. `testConstructorWithUser`

Descripción: Este test verifica la correcta construcción de un objeto `UserResponse` a partir de un objeto `User`.

Verificaciones:

- Se espera que las propiedades del objeto resultante sean: `id = "123"`, `firstName = "John"`, `lastName = "Doe"`, `email = "john.doe@example.com"`, `username = "johndoe"`.

Resultado: El test está correctamente diseñado y asegura que la construcción del objeto `UserResponse` desde un objeto `User` se realice correctamente.

3. `User` Test

a. `testConstructorWithUserRepresentation`

Descripción: Este test verifica la correcta construcción de un objeto `User` a partir de un objeto `UserRepresentation`.

Verificaciones:

- Se espera que las propiedades del objeto resultante sean: `id = "123"`, `firstName = "John"`, `lastName = "Doe"`, `email = "john.doe@example.com"`, `username = "johndoe"`.
- Se espera que la propiedad `password` sea `null`.

Resultado: El test está correctamente diseñado y asegura que la construcción del objeto `User` desde un `UserRepresentation` se realice correctamente y que la propiedad `password` no sea establecida por el constructor.

Recomendaciones

1. **Revisión del Test de Serialización en `TokenResponse`**: Revisar y corregir la configuración del `ObjectMapper` o las expectativas del test `testSerialization` para asegurar que refleje el comportamiento esperado.
2. **Consistencia en Verificaciones**: Asegurar que las verificaciones reflejen consistentemente las expectativas del comportamiento de serialización y deserialización para todos los tests.

Conclusión

La mayoría de los tests están bien diseñados y cumplen con los objetivos de verificar la correcta serialización, deserialización y construcción de objetos. La única excepción notable es el test `testSerialization` de `TokenResponse`, que requiere una revisión y ajuste para alinearse con las expectativas de comportamiento.