

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Universidad Nacional de Colombia - Sede Bogotá
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas e Industrial

Asignatura:
Ingeniería de Software I

Proyecto - Informe Testing

Natalia Carolina Bautista Sanchez
Daniel Santiago Cocinero Jimenez
Baruj Vladimir Ramirez Escalante
Juan Daniel Ramirez Mojica

Profesor:
Oscar Eduardo Alvarez Rodriguez

2024 - II
Bogotá D.C.

Introducción

Este proyecto consiste en el desarrollo de una plataforma digital para la lectura colectiva y la discusión de libros. La idea surge a partir de la observación de una oportunidad de negocio basada en la afición de un integrante del equipo por la lectura, identificando la necesidad de un espacio virtual que facilite la creación y gestión de clubes de lectura en línea.

El objetivo es crear un Producto Mínimo Viable (MVP) con funcionalidades esenciales: registro de usuarios, gestión de clubes, debates organizados y votaciones para seleccionar libros. Se priorizarán la seguridad, el rendimiento y un diseño intuitivo.

Se utilizará Java, JavaScript, Spring Boot y bases de datos no relacionales. Además, el proyecto tiene potencial de impacto cultural y oportunidades de financiación con entidades y editoriales.

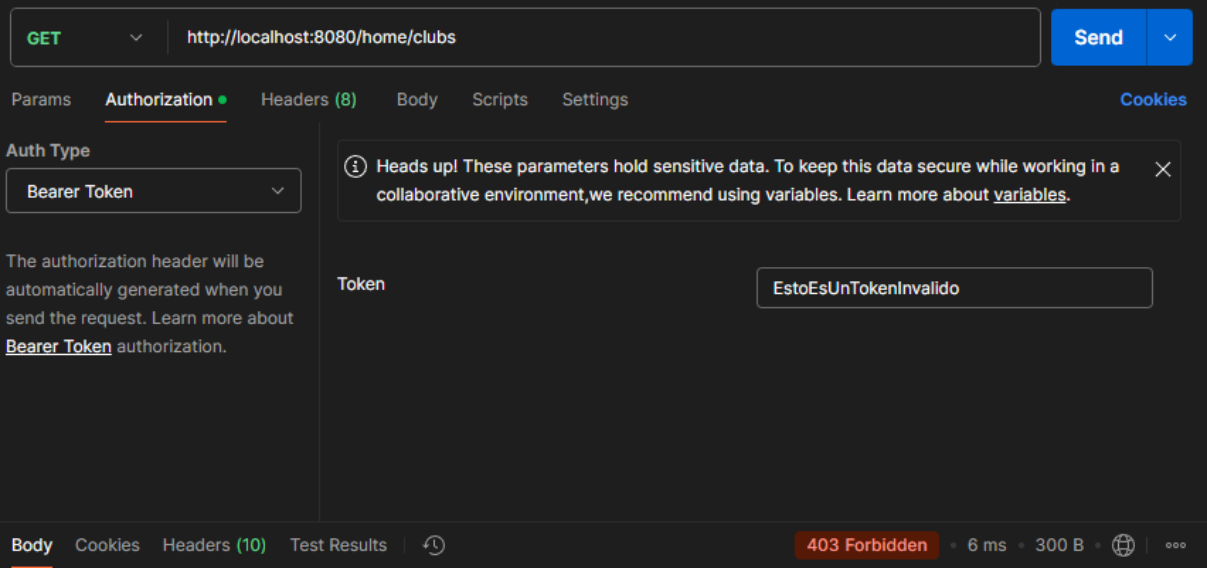
Resumen de los tests

Integrante	Tipo de prueba	Descripción del componente probado	Herramienta o framework usado	Código del test	Resultado de la ejecución
Daniel Santiago Cocinero Jimenez	Prueba de integración	Verificación del proceso de registro de usuarios en la plataforma.	Spring Boot	<pre>@Test new * void testRegister() throws Exception { String userJson = "{\"username\":\"testuser\", \"password\":" mockMvc.perform(post(uriTemplate: "/auth/register") .contentType(MediaType.APPLICATION_JSON) .content(userJson)) .andExpect(status().isOk()); }</pre>	<p>Prueba exitosa, se registró el usuario en la plataforma.</p> <div><div>✓ AuthControllerTests (org.worm.bo1sec 72 ms</div><div>✓ testRegister()1sec 72 ms</div></div> <p>MongoDB:</p> <p>_id: ObjectId('67c120b646e1a75f502cd1d6')</p> <p>username : "testuser"</p> <p>email : "testuser@example.com"</p> <p>password : "\$2a\$10\$nSWNgpD2CBf2EYwYEV7hC.u7hDA9UjkcoeG61p5ryUwLYcu3N4K0."</p> <p>role : "USER"</p> <p>_class : "org.worm.bookhunt.user.User"</p>
Daniel Santiago Cocinero Jimenez	Prueba de integración	Verificación del proceso de registro de usuarios en la plataforma.	Spring Boot	<pre>@Test new * void testRegisterWithEmptyParams() throws Exception { String emptyUserJson = "{\"username\":\"\", \"email\":\"\" mockMvc.perform(post(uriTemplate: "/auth/register") .contentType(MediaType.APPLICATION_JSON) .content(emptyUserJson)) .andExpect(status().isBadRequest()); }</pre>	<p>Prueba exitosa, Usuario vacío no válido para registro.</p> <div><div>✓ AuthControllerTests (org.worm.bookh 857 ms</div><div>✓ testRegisterWithEmptyParams()857 ms</div></div>

Daniel Santiago Cocinero Jimenez	Prueba de integración	Verificación del proceso de registro de usuarios en la plataforma.	Spring Boot	<pre> @Test new * void testRegisterDuplicateUser() throws Exception { String userJson = "{\"username\":\"testuser1\", \"password\":\"password\"}"; // Register the first user mockMvc.perform(post(uriTemplate: "/auth/register") .contentType(MediaType.APPLICATION_JSON) .content(userJson)) .andExpect(status().isOk()); // Attempt to register the same user again mockMvc.perform(post(uriTemplate: "/auth/register") .contentType(MediaType.APPLICATION_JSON) .content(userJson)) .andExpect(status().isConflict()); } </pre>	<p>Se detectó un error: el sistema permite registrar usuarios duplicados.</p>  <p>MongoDB:</p> <pre> _id: ObjectId('67c121fea8866c22ee40feae') username: "testuser1" email: "testuser1@example.com" password: "\$2a\$10\$6HzCEn9ndG9KvsK1z.ZCfuJaSyk4ATuFqgn0OH5hkP.wbITnuJcJq" role: "USER" _class: "org.worm.bookhunt.user.User" </pre> <hr/> <pre> _id: ObjectId('67c121fea8866c22ee40feaf') username: "testuser1" email: "testuser1@example.com" password: "\$2a\$10\$J3bKIUGFBaWYppVlZaxdu6V/Pp7v/iPx4qaIpou9kVaGVTxVhHJ6" role: "USER" _class: "org.worm.bookhunt.user.User" </pre>
Daniel Santiago Cocinero Jimenez	Prueba de integración	Verificación del proceso de login de usuarios en la plataforma.	Spring Boot	<pre> @Test new * void testLogin() throws Exception { String loginJson = "{\"username\":\"testuser\", \"password\":\"password\"}"; mockMvc.perform(post(uriTemplate: "/auth/login") .contentType(MediaType.APPLICATION_JSON) .content(loginJson)) .andExpect(status().isOk()); } </pre>	<p>Prueba exitosa, el usuario puede realizar el login.</p> 
Daniel Santiago Cocinero Jimenez	Prueba de integración	Verificación del proceso de login de usuarios en la plataforma.	Spring Boot	<pre> @Test new * void testLoginWithEmptyParams() throws Exception { String emptyLoginJson = "{\"username\":\"\", \"password\":\"\"}"; mockMvc.perform(post(uriTemplate: "/auth/login") .contentType(MediaType.APPLICATION_JSON) .content(emptyLoginJson)) .andExpect(status().isForbidden()); } </pre>	<p>Prueba exitosa, se obtiene el error al recibir los parámetros en blanco.</p> 

Juan Daniel Ramírez Mojica	Prueba de integración	Verificación del funcionamiento al salir de un club.	Spring Boot	<pre> @Test new * void testRemoveUserFromClub() throws Exception { String token = "Bearer eyJhbGciOiJIUzI4NCJ9.eyJzdWIiOiJ0ZXN0dXNlcjE1NTU1NTU1NSIsImhhdCI6MTYwMzQwNzQzODU4fQ.QMKIpeHrEtct6mSABJeYlKeTfuWEkgYasNt1mj3jML_a1ufQq4Ydz9vRAhfnWke5"; String clubId = "67c0fc8b11d9dd000959316d"; String userId = "testuser15555555"; mockMvc.perform(MockMvcRequestBuilders.delete(uriTemplate: "/clubs/{clubId}/remove", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isOk()); } </pre>	<p>Prueba exitosa, los usuarios pueden salir de los clubes sin problemas.</p> <div> <div>✓ ClubControllerTests (org.worm.bookhur 814 ms)</div> <div>✓ testRemoveUserFromClub() 814 ms</div> </div>
Juan Daniel Ramírez Mojica	Prueba de integración	Verificación del funcionamiento al salir de un club.	Spring Boot	<pre> @Test new * void testRemoveUserFromClubUserNotInClub() throws Exception { String token = "Bearer eyJhbGciOiJIUzI4NCJ9.eyJzdWIiOiJ0ZXN0dXNlcjE1NTU1NTU1NSIsImhhdCI6MTYwMzQwNzQzODU4fQ.QMKIpeHrEtct6mSABJeYlKeTfuWEkgYasNt1mj3jML_a1ufQq4Ydz9vRAhfnWke5"; String clubId = "67c0fc8b11d9dd000959316d"; String userId = "testuser15555555"; // First remove attempt mockMvc.perform(MockMvcRequestBuilders.delete(uriTemplate: "/clubs/{clubId}/remove", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isOk()); // Second remove attempt mockMvc.perform(MockMvcRequestBuilders.delete(uriTemplate: "/clubs/{clubId}/remove", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isNotFound()); } </pre>	<p>Prueba fallida, se esperaba que dijera que el usuario no se encontraba en el club.</p> <div> <div>✗ ClubControllerTests (org.worm.bookh 985 ms)</div> <div>✗ testRemoveUserFromClubUserNotInClub() 985 ms</div> </div> <p><code>.andExpect(status().isNotFound());</code></p> <p>java.lang.AssertionError: Status expected:<404> but was:<200></p>
Baruj Vladimir Ramírez Escalante	Prueba de integración	Validación del proceso de unión a un club.	Spring Boot	<pre> @Test new * void testJoinClub() throws Exception { String token = "Bearer eyJhbGciOiJIUzI4NCJ9.eyJzdWIiOiJ0ZXN0dXNlcjE1NTU1NTU1NSIsImhhdCI6MTYwMzQwNzQzODU4fQ.QMKIpeHrEtct6mSABJeYlKeTfuWEkgYasNt1mj3jML_a1ufQq4Ydz9vRAhfnWke5"; String clubId = "67c0fc8b11d9dd000959316d"; String userId = "testuser15555555"; mockMvc.perform(MockMvcRequestBuilders.post(uriTemplate: "/clubs/{clubId}/join", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isOk()); } </pre>	<p>Prueba exitosa, los usuarios pueden unirse a clubes correctamente.</p> <div> <div>✓ ClubControllerTests (org.worm.bookh 849 ms)</div> <div>✓ testJoinClub() 849 ms</div> </div>

Baruj Vladimir Ramirez Escalante	Prueba de integración	Validación del proceso de unión a un club.	Spring Boot	<pre>@Test new * void testJoinClubUserAlreadyInClub() throws Exception { String token = "Bearer eyJhbGciOiJIUzI1NiIsInR5cGE6YWV0Ij0ZXN0dXNlcjE1NTU1NTU1NSIsImhhdCI6 "oxNzQwNzQzODU4fQ.QMKIpeHrEtct6mSABJeYLKeTfuWEkgYasNt1mj3jML_aiufQq4Ydz9vRAhfnW String clubId = "67c0fc8b11d9dd000959316d"; String userId = "testuser155555555"; // First join attempt mockMvc.perform(MockMvcRequestBuilders.post(uriTemplate: "/clubs/{clubId}/join", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isOk()); // Second join attempt mockMvc.perform(MockMvcRequestBuilders.post(uriTemplate: "/clubs/{clubId}/join", clubId) .param(name: "userId", userId) .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isConflict()); }</pre>	<p>Se detectó un error: el sistema permite que el mismo usuario se una dos veces a el mismo club.</p> <div><div>✖ ClubControllerTests (org.worm.bookhunt.club)</div><div>✖ testJoinClubUserAlreadyInClub() 958 ms</div></div> <div><pre>.andExpect(status().isConflict());</pre><div>java.lang.AssertionError: Status expected:<409> but was:<200></div></div> <p>En la base de datos podemos observar que realmente no se agregan duplicados debido a la construcción de la colección pero la lógica del programa si lo permite.</p> <div><pre>_id: ObjectId('67c0fc8b11d9dd000959316d') name: "Terror Club" description: "This is the default club created at startup." members: Array (1) 0: "testuser155555555" _class: "org.worm.bookhunt.club.Club"</pre></div>
Natalia Carolina Bautista Sanchez	Prueba de integración	Comprobación de que el Home Controller muestra correctamente la lista de clubes.	Spring Boot	<pre>@Test new * void testGetUserClubs() throws Exception { String token = "Bearer eyJhbGciOiJIUzI1NiIsInR5cGE6YWV0Ij0ZXN0dXNlcjE1NTU1NTU1NSIsImhhdCI6 "oxNzQwNzQzODU4fQ.QMKIpeHrEtct6mSABJeYLKeTfuWEkgYasNt1mj3jML_ mockMvc.perform(MockMvcRequestBuilders.get(uriTemplate: "/home/clubs") .header(name: "Authorization", token) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isOk()) .andExpect(jsonPath(expression: "\$", hasSize(1))); }</pre>	<p>Prueba exitosa, la lista de clubes se muestra correctamente en la página de inicio.</p> <div><div>✔ HomeControllerTests (org.worm.bookhui</div><div>✔ testGetUserClubs() 997 ms</div></div>

<p>Natalia Carolina Bautista Sanchez</p>	<p>Prueba de integración</p>	<p>Comprobación de que el Home Controller muestra correctamente la lista de clubes.</p>	<p>Spring Boot</p>	<pre> @Test new * void testGetUserClubsWithInvalidToken() throws Exception { String invalidToken = "Bearer eyJhbGciOiJIUzI1NiIsInR5cGU6IjY9LWZQ1MjQ5fQ.T2YI2o1wNachBDnwoYP6tD_Ho709wfy7YCNp_Hha9v-N9CU9s" mockMvc.perform(MockMvcRequestBuilders.get(uriTemplate: "/home/clubs") .header(name: "Authorization", invalidToken) .contentType(MediaType.APPLICATION_JSON)) .andExpect(status().isForbidden()); } </pre>	<p>Se detectó un error: Aunque al hacer una petición de este tipo en Postman el código de estado si es correcto. Ejemplo:</p>  <p>Debido al funcionamiento de las pruebas integradas de Spring Boot, el error que salta es relacionado con que el token está mal hecho o pertenece a un usuario inexistente y no se logra probar de forma fiable la respuesta a esta petición.</p> <pre> mockMvc.perform(MockMvcRequestBuilders.get(uriTemplate: "/home/clubs")) </pre> <p>io.jsonwebtoken.security.SignatureException: JWT signature does not match locally computed signature. JWT validity cannot be asserted and should not be trusted.</p> <p>! HomeControllerTests (org.worm.bookhur 777 ms)</p> <p>! testGetUserClubsWithInvalidToken() 777 ms</p>
--	------------------------------	--	--------------------	--	--

Lecciones aprendidas y dificultades

- Aunque nos apoyamos en la IA para la generación de los test, muchas veces tocó realizar cambios y ajustes al código generado, ya que los test salían defectuosos o poco útiles.
- Durante la ejecución de las pruebas, se identificó un problema crítico en el registro de usuarios, donde el sistema permitía la creación de cuentas duplicadas. Este hallazgo refuerza la importancia de implementar y ejecutar pruebas, ya que permiten detectar errores que podrían pasar desapercibidos en un desarrollo sin una estrategia de testing adecuada.
- La implementación de tests nos facilitó la validación de diversas funcionalidades de la aplicación, permitiendo detectar y corregir errores con mayor rapidez. Esto nos permitió observar como una estrategia de testing bien aplicada no solo ayuda a mejorar la calidad del software, sino que también hace más eficiente el proceso de desarrollo al permitir iteraciones y ajustes continuos.
- Al implementar los tests, nos dimos cuenta de que el testing facilita la interacción con la base de datos, permitiéndonos crear usuarios y clubes de manera sencilla. Esto resultó bastante útil, ya que anteriormente habíamos enfrentado dificultades en estas tareas.
- Identificamos la necesidad de profundizar en el funcionamiento de las pruebas en Spring Boot, ya que en un caso específico obtuvimos un comportamiento distinto al realizar la misma petición a través de Postman.