# **Pruebas Login**

## Introducción

Bienvenido a nuestra plantilla de gestión de usuarios. Esta solución simple y eficaz incluye funciones para el inicio de sesión, registro de usuarios, asignación de roles, recuperación de contraseñas y una vista de bienvenida. Ideal para quienes buscan una base sólida y fácil de implementar para manejar el acceso y la administración de usuarios.

**Contenido**

[**Pruebas Login** 1](#_Toc176533288)

[Introducción 2](#_Toc176533289)

[CoreApplication 5](#_Toc176533290)

[Verificación de Compilación correctamente 5](#_Toc176533291)

[ApplicationConfigTest 6](#_Toc176533292)

[Verificación de la Configuración de la Base de Datos 6](#_Toc176533293)

[Verificación de inicio de sesión exitoso (AuthController) 7](#_Toc176533294)

[Verificación de usuario no encontrado durante el inicio de sesión (AuthController) 7](#_Toc176533295)

[Verificación de credenciales incorrectas en inicio de sesión (AuthController) 8](#_Toc176533296)

[Verificación de actualización de token (AuthController) 8](#_Toc176533297)

[Verificación de registro de usuario exitoso (AuthController) 9](#_Toc176533298)

[Verificación de registro con nombre de usuario existente (AuthController) 9](#_Toc176533299)

[Verificación de registro con email existente (AuthController) 10](#_Toc176533300)

[Verificación de obtención de usuario por nombre o nombre de usuario (UserService) 11](#_Toc176533301)

[Verificación de obtención de usuario por token de contraseña (UserService) 11](#_Toc176533302)

[Verificación de obtención de usuario por nombre de usuario (UserService) 12](#_Toc176533303)

[Verificación de existencia de nombre de usuario (UserService) 12](#_Toc176533304)

[Verificación de existencia de email (UserService) 13](#_Toc176533305)

[Verificación de guardado de usuario (UserService) 13](#_Toc176533306)

[Verificación de carga de usuario por nombre de usuario (UserDetailsServiceImpl) 14](#_Toc176533307)

[Verificación de recuperación de rol por nombre (RolService) 15](#_Toc176533308)

[Verificación de recuperación de rol por nombre no encontrado (RolService) 15](#_Toc176533309)

[Verificación de guardado de rol (RolService) 16](#_Toc176533310)

[Verificación de envío de correo cuando el usuario no se encuentra (EmailPasswordController) 17](#_Toc176533311)

[Verificación de envío de correo exitoso (EmailPasswordController) 17](#_Toc176533312)

[Verificación de cambios de contraseña con errores de validación (EmailPasswordController) 18](#_Toc176533313)

[Verificación de cambio de contraseña cuando las contraseñas no coinciden (EmailPasswordController) 18](#_Toc176533314)

[Verificación de cambio de contraseña cuando el usuario no se encuentra (EmailPasswordController) 19](#_Toc176533315)

[Verificación de cambio de contraseña exitoso (EmailPasswordController) 19](#_Toc176533316)

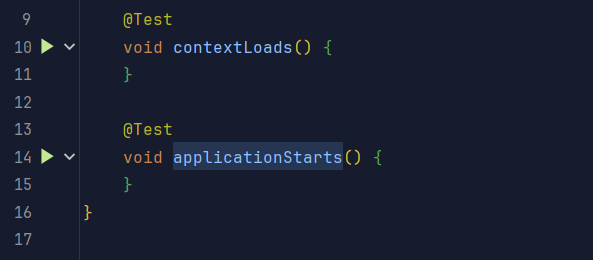
## CoreApplication

### Verificación de Compilación correctamente

Esta prueba asegura que la aplicación arranca correctamente y que el contexto de Spring se carga sin errores.

**Pruebas Incluidas:**

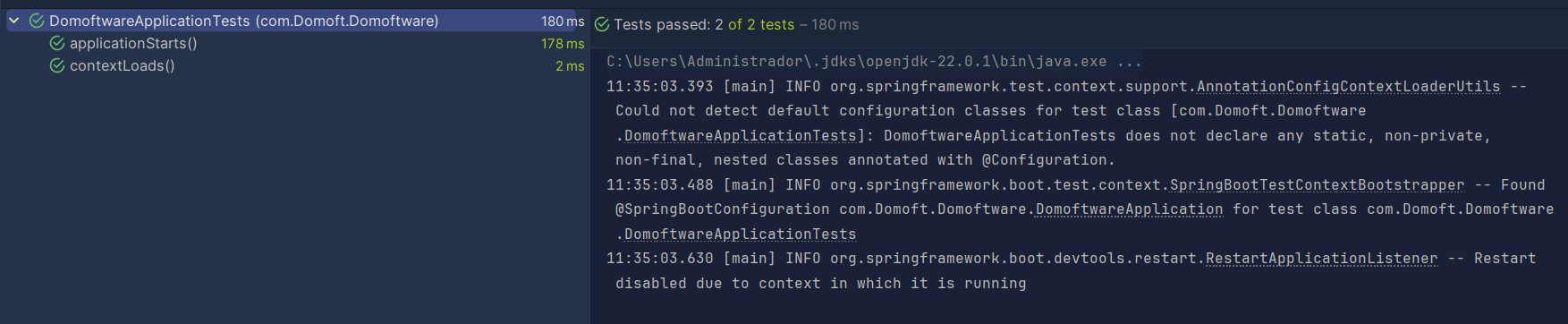
1. **Carga del Contexto (contextLoads()):** Verifica que el contexto de Spring se inicializa correctamente sin errores.
2. **Arranque de la Aplicación (applicationStarts()):** Confirma que el método main de la aplicación se ejecuta sin excepciones, garantizando un arranque exitoso.

****

**Resultados de la Prueba**

1. Ambas pruebas deben completarse sin excepciones. El mensaje de éxito es: "La aplicación se ha inicializado correctamente. El contexto de Spring se ha cargado sin errores y la aplicación ha arrancado sin problemas."

**Salida de la Ejecución de la Prueba**

****

## ApplicationConfigTest

### Verificación de la Configuración de la Base de Datos

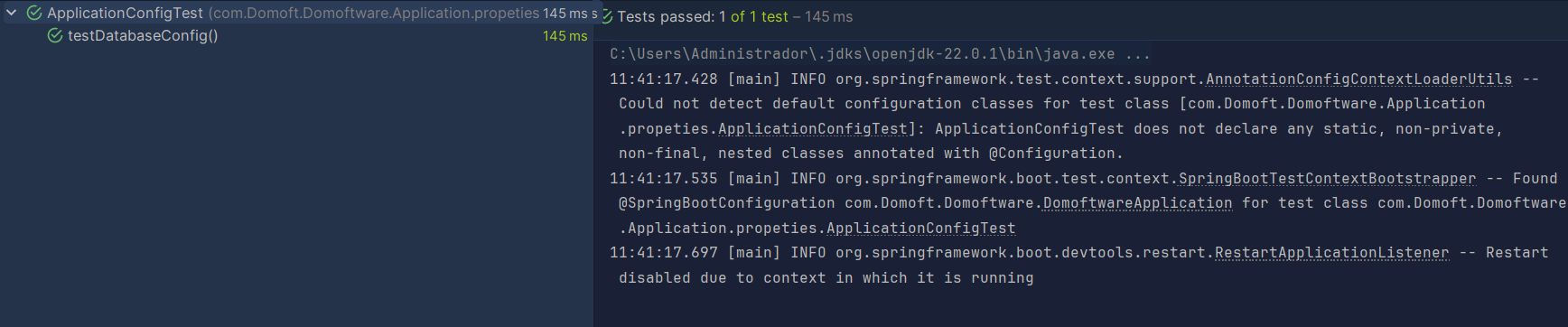
Esta prueba valida que los parámetros de configuración de la base de datos se carguen correctamente desde el archivo de propiedades. Verifica la URL, el nombre de usuario y la contraseña de la base de datos.

****

**Resultados de la Prueba**   
  
La prueba pasa si los valores de spring.datasource.url, spring.datasource.username, y spring.datasource.password coinciden con:

* URL: "jdbc:mysql://localhost:3306/login?useSSL=false&serverTimezone=UTC&useLegacyDateTimeCode=false"
* Usuario: "root"
* Contraseña: "root"

**Salida de la Ejecución de la Prueba**

****

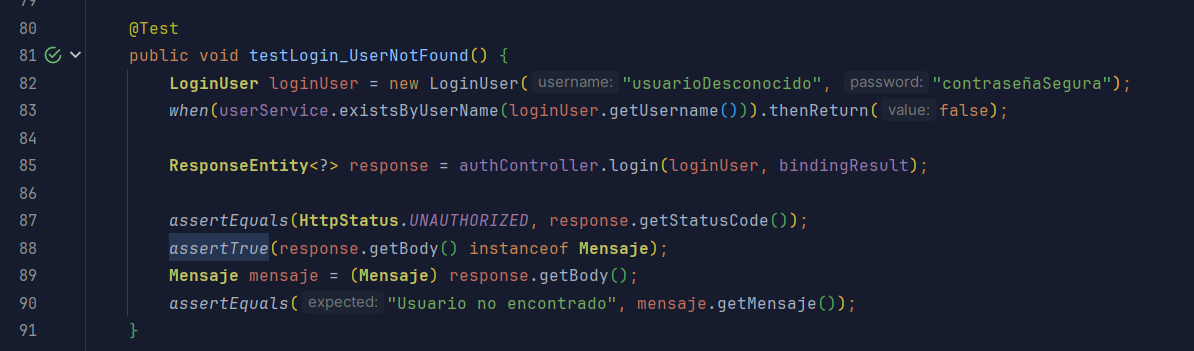
### Verificación de inicio de sesión exitoso (AuthController)

La prueba `testLogin\_Success` valida el comportamiento del sistema cuando un usuario proporciona credenciales correctas. Se simula la autenticación con un token válido y se comprueba que el controlador responde con un token JWT generado exitosamente, junto con el código de estado HTTP `200 OK`. Esto asegura que el sistema maneja correctamente un inicio de sesión exitoso y devuelve la información correcta al usuario.



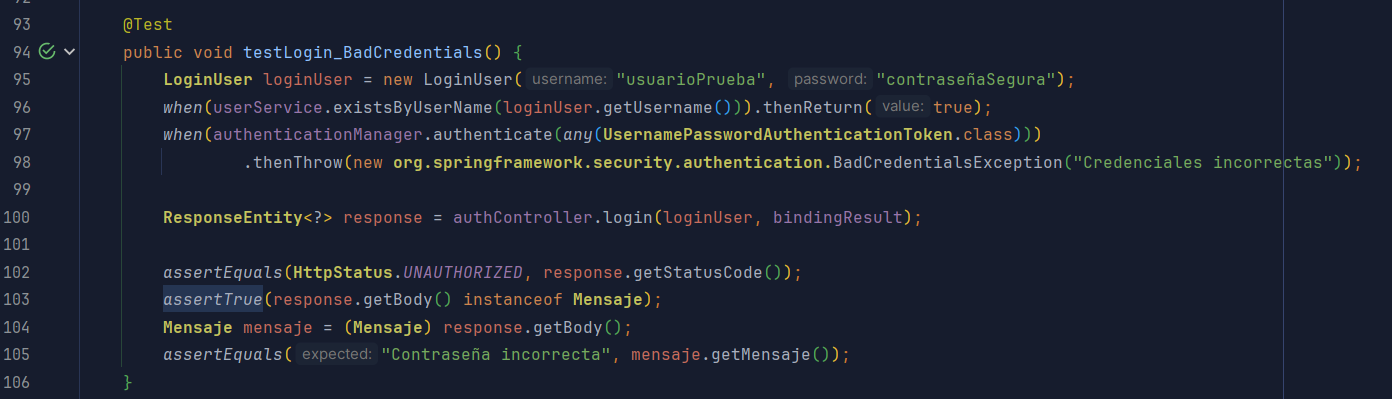
### Verificación de usuario no encontrado durante el inicio de sesión (AuthController)

La prueba `testLogin\_UserNotFound` verifica cómo el sistema responde cuando un usuario intenta iniciar sesión con un nombre de usuario que no existe en la base de datos. El test asegura que el sistema responda con un código de estado HTTP `401 Unauthorized` y un mensaje de error indicando que el usuario no fue encontrado. Esto valida que el controlador maneja adecuadamente los intentos de inicio de sesión con nombres de usuario no registrados.



### Verificación de credenciales incorrectas en inicio de sesión (AuthController)

La prueba `testLogin\_BadCredentials` comprueba que el sistema maneje correctamente el caso en que un usuario proporciona una contraseña incorrecta. Se simula una excepción de credenciales incorrectas y se asegura que el sistema responda con un código de estado HTTP `401 Unauthorized` y un mensaje de error adecuado. Esto valida que el controlador maneja adecuadamente los errores de autenticación por contraseñas incorrectas.



### Verificación de actualización de token (AuthController)

La prueba `testRefreshToken` valida que el sistema pueda generar un nuevo token JWT a partir de un token antiguo. Se simula la solicitud de actualización de token y se verifica que el controlador responda con un nuevo token y un código de estado HTTP `200 OK`. Esto asegura que la funcionalidad de actualización de token está implementada correctamente y el sistema devuelve el token nuevo esperado. 

### Verificación de registro de usuario exitoso (AuthController)

La prueba `testRegister\_Success` valida el comportamiento del sistema cuando un nuevo usuario se registra con éxito. Se simula la creación de un nuevo usuario y se comprueba que el controlador responda con un código de estado HTTP `201 Created` y un mensaje de éxito. Esto asegura que el controlador maneja correctamente el registro de nuevos usuarios y devuelve una respuesta adecuada al éxito del registro.



### Verificación de registro con nombre de usuario existente (AuthController)

La prueba `testRegister\_UsernameExists` verifica cómo el sistema maneja el intento de registro de un nuevo usuario con un nombre de usuario que ya existe. Se simula una solicitud de registro con un nombre de usuario ya registrado y se asegura que el controlador responda con un código de estado HTTP `400 Bad Request` y un mensaje indicando que el nombre de usuario ya existe. Esto valida que el controlador maneja adecuadamente los conflictos de nombre de usuario durante el registro.

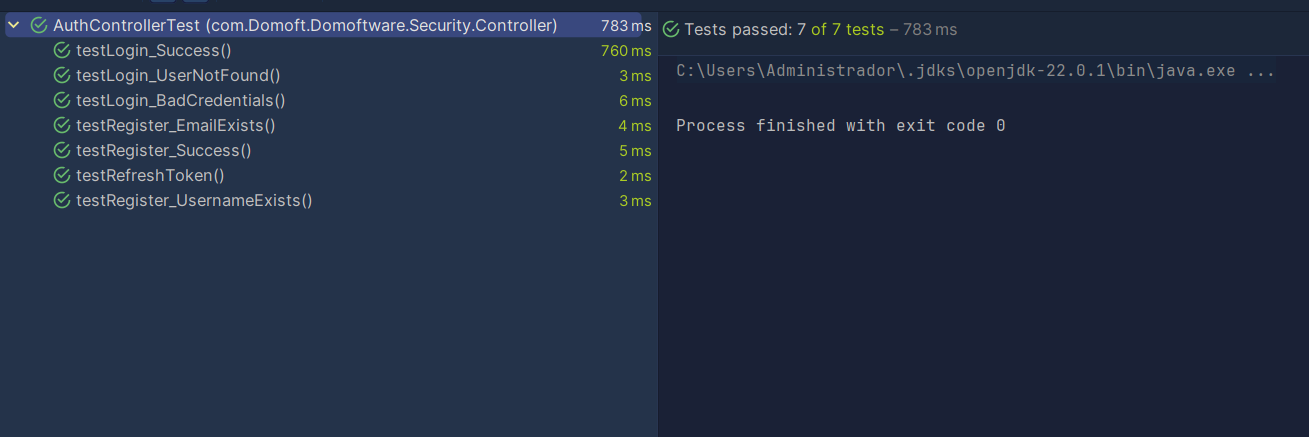


### Verificación de registro con email existente (AuthController)

La prueba `testRegister\_EmailExists` valida cómo el sistema maneja el intento de registro de un nuevo usuario con un email que ya está en uso. Se simula una solicitud de registro con un email ya registrado y se verifica que el controlador responda con un código de estado HTTP `400 Bad Request` y un mensaje indicando que el email ya existe. Esto asegura que el controlador maneja adecuadamente los conflictos de email durante el registro.

****

**Resultado de pruebas:**

****

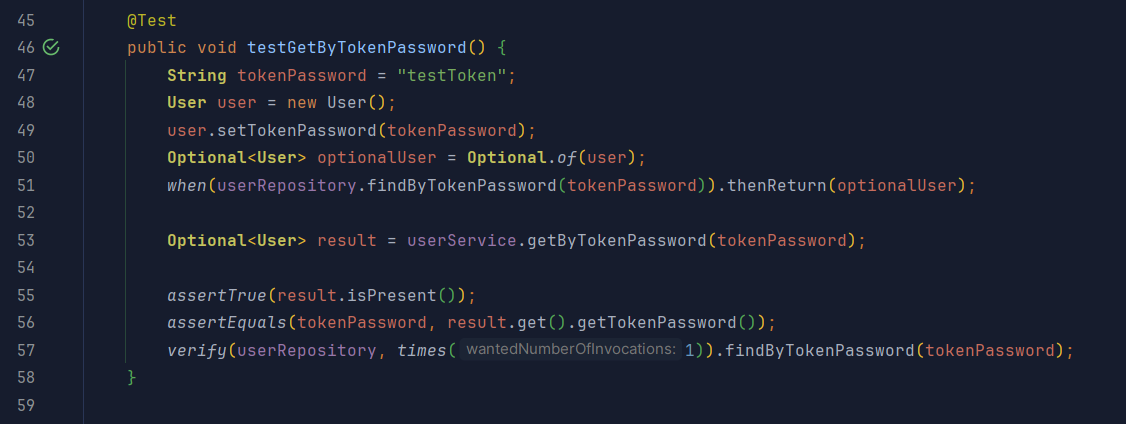
### Verificación de obtención de usuario por nombre o nombre de usuario (UserService)

La prueba `testGetBynameOrUsername` valida que el método `getBynameOrUsername` del `UserService` funcione correctamente al buscar un usuario por nombre o nombre de usuario. Se simula la respuesta del repositorio con un usuario que coincide con el nombre proporcionado y se verifica que el resultado devuelto contenga el usuario esperado. Además, se confirma que el método del repositorio fue llamado una vez.



### Verificación de obtención de usuario por token de contraseña (UserService)

La prueba `testGetByTokenPassword` verifica que el método `getByTokenPassword` del `UserService` retorna un usuario basado en el token de contraseña proporcionado. Se simula el comportamiento del repositorio para devolver un usuario con el token especificado y se asegura que el método del servicio devuelva el usuario correcto. También se comprueba que el repositorio fue invocado una vez.



### Verificación de obtención de usuario por nombre de usuario (UserService)

La prueba `testGetByUserName` valida que el método `getByUserName` del `UserService` devuelve el usuario correcto basado en el nombre de usuario proporcionado. Se simula una respuesta del repositorio con un usuario que tiene el nombre de usuario especificado y se verifica que el resultado es el esperado. Se confirma que el repositorio fue llamado una vez.



### Verificación de existencia de nombre de usuario (UserService)

La prueba `testExistsByUserName` asegura que el método `existsByUserName` del `UserService` correctamente determina si un nombre de usuario existe en la base de datos. Se simula que el repositorio indica que el nombre de usuario existe y se verifica que el método del servicio retorne el valor esperado. También se confirma que el repositorio fue llamado una vez.



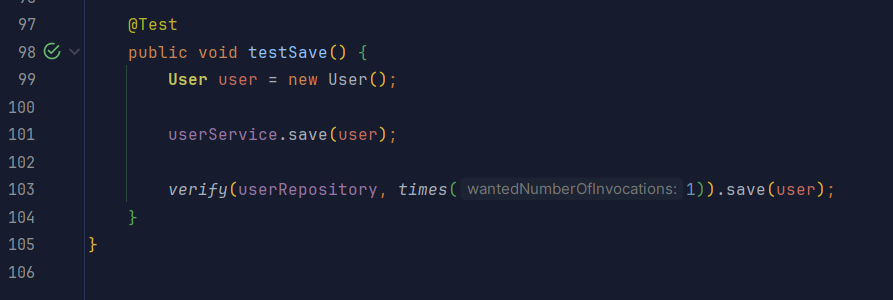
### Verificación de existencia de email (UserService)

La prueba `testExistsByEmail` valida que el método `existsByEmail` del `UserService` determina correctamente si un email ya está registrado. Se simula una respuesta positiva del repositorio y se asegura que el método del servicio devuelva el resultado esperado. También se verifica que el repositorio fue llamado una vez.

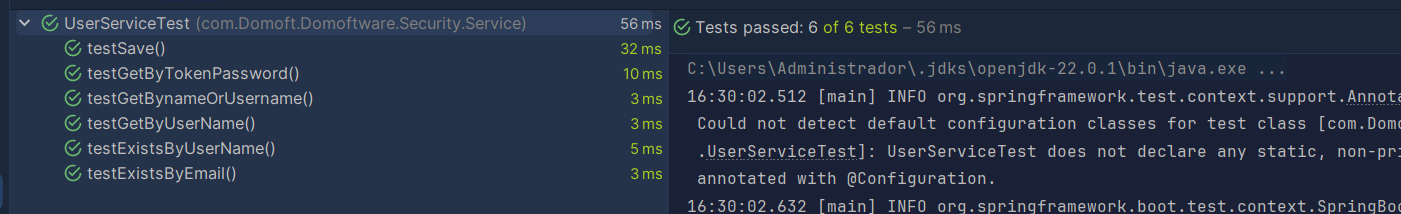


### Verificación de guardado de usuario (UserService)

La prueba `testSave` comprueba que el método `save` del `UserService` llama correctamente al método `save` del repositorio para guardar un usuario. Se simula la llamada al repositorio y se verifica que el método `save` del repositorio sea invocado una vez con el usuario proporcionado.

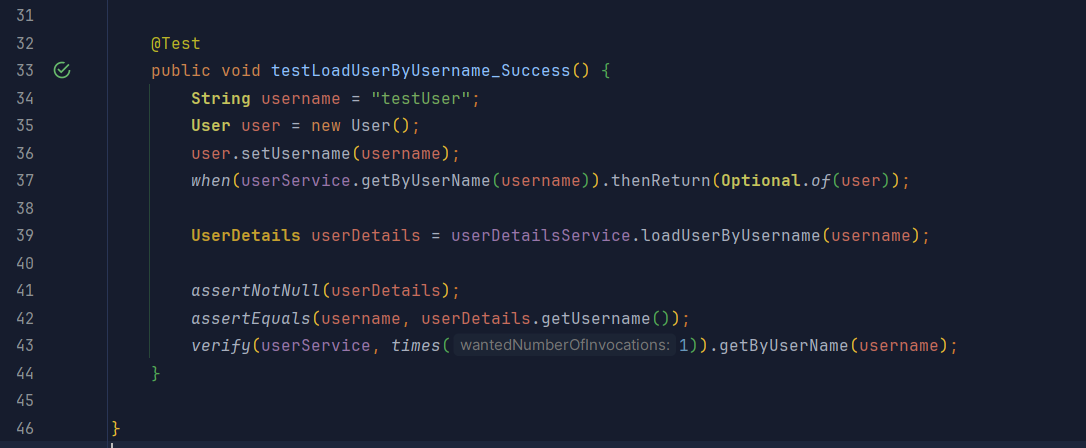


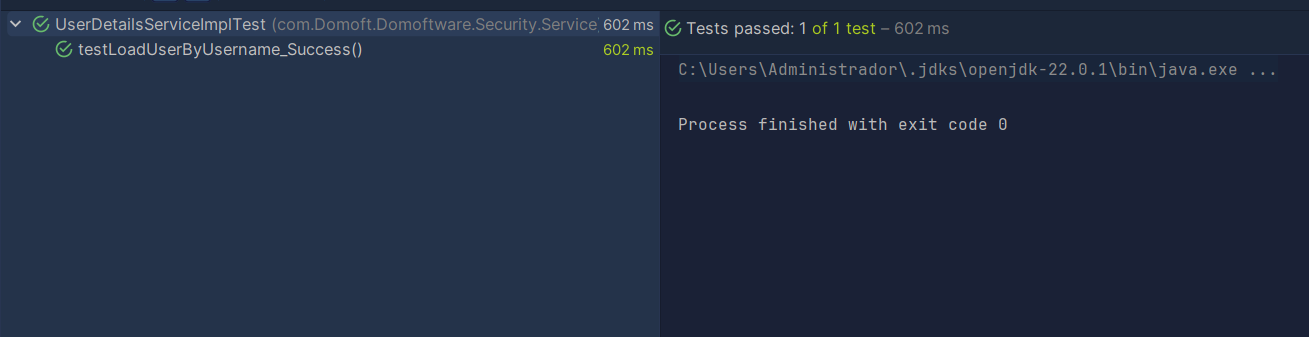
**Resultado de pruebas:**

****

### Verificación de carga de usuario por nombre de usuario (UserDetailsServiceImpl)

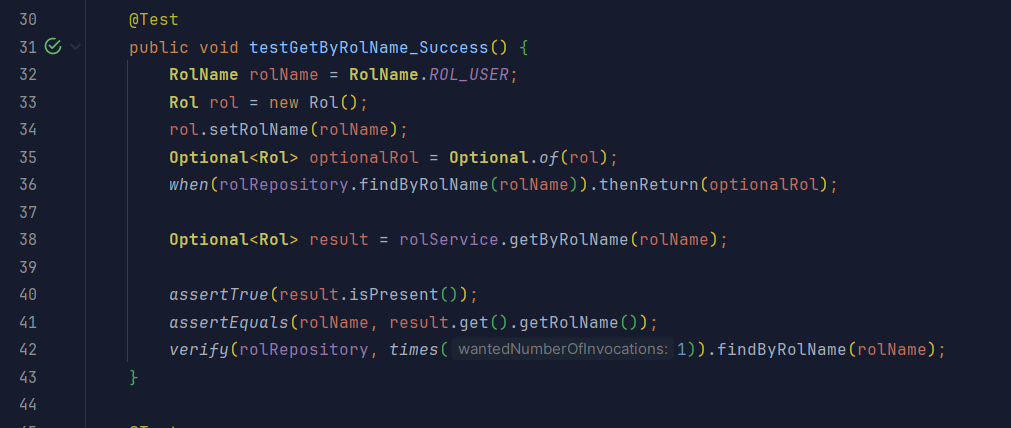
La prueba `testLoadUserByUsername\_Success` valida que el método `loadUserByUsername` del `UserDetailsServiceImpl` funcione correctamente al cargar un usuario basado en su nombre de usuario. Se simula que el `UserService` devuelve un usuario que coincide con el nombre de usuario proporcionado. Se verifica que el objeto `UserDetails` no sea nulo y que el nombre de usuario coincida con el esperado. Además, se confirma que el método `getByUserName` del `UserService` fue llamado una vez.

**  
  
Resultado de pruebas:**

****

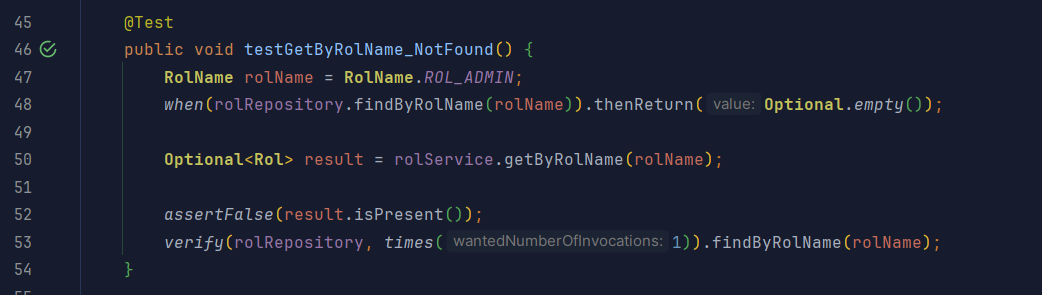
### Verificación de recuperación de rol por nombre (RolService)

La prueba `testGetByRolName\_Success` valida que el método `getByRolName` del `RolService` funcione correctamente al recuperar un rol basado en el nombre de rol proporcionado. Se simula que el `RolRepository` devuelve un rol con el nombre especificado. Se verifica que el resultado no esté vacío y que el nombre del rol coincida con el esperado. Además, se confirma que el método `findByRolName` del `RolRepository` haya sido llamado una vez.



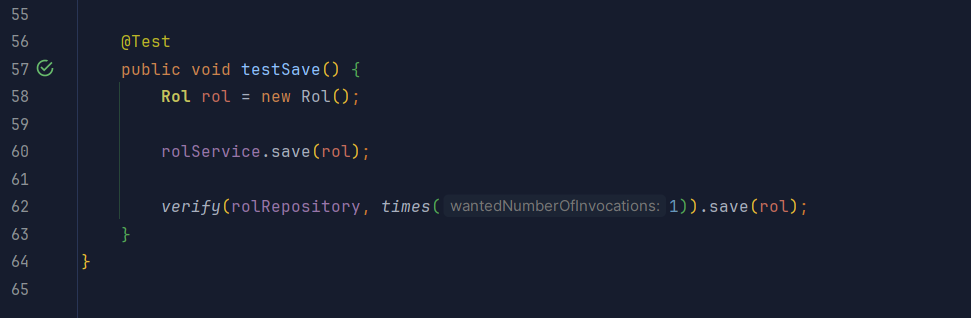
### Verificación de recuperación de rol por nombre no encontrado (RolService)

La prueba `testGetByRolName\_NotFound` valida que el método `getByRolName` del `RolService` maneje correctamente el caso en que no se encuentra un rol con el nombre de rol proporcionado. Se simula que el `RolRepository` devuelve un `Optional.empty()`. Se verifica que el resultado esté vacío y que el método `findByRolName` del `RolRepository` haya sido llamado una vez.

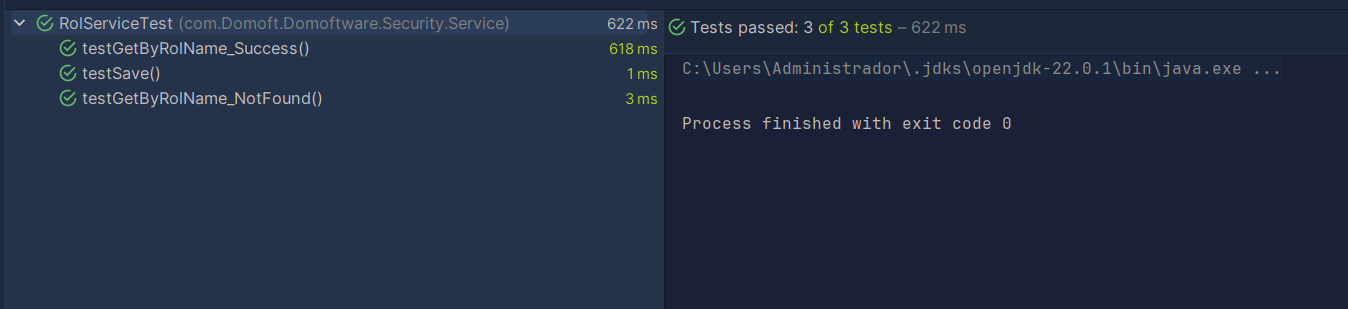


### Verificación de guardado de rol (RolService)

La prueba `testSave` valida que el método `save` del `RolService` funcione correctamente al guardar un rol en la base de datos. Se simula que el `RolRepository` guarda el rol sin errores. Se confirma que el método `save` del `RolRepository` haya sido llamado una vez con el rol proporcionado.

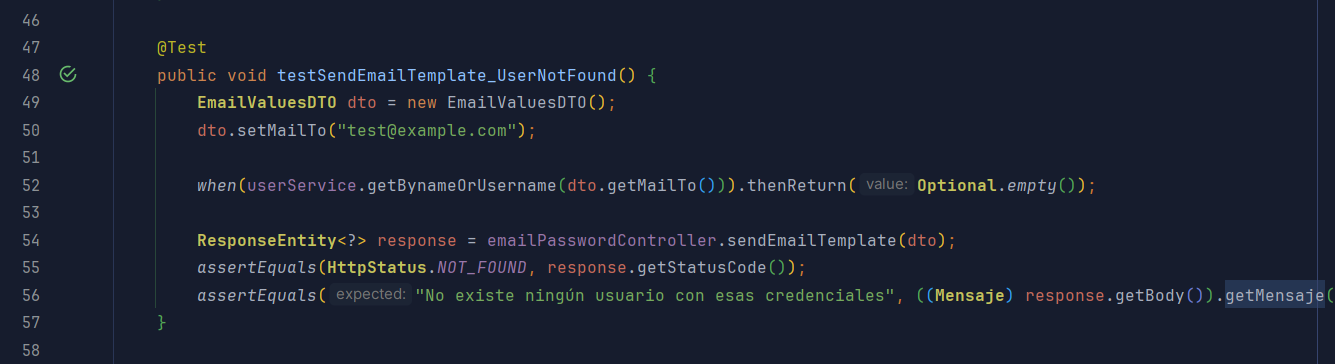
****

**Resultado de pruebas:**

****

### Verificación de envío de correo cuando el usuario no se encuentra (EmailPasswordController)

La prueba `testSendEmailTemplate\_UserNotFound` valida que el método `sendEmailTemplate` del `EmailPasswordController` maneje correctamente el caso en que no se encuentra un usuario con el correo electrónico proporcionado. Se simula que el `UserService` no encuentra ningún usuario con el correo electrónico dado. Se verifica que la respuesta tenga el estado HTTP `NOT\_FOUND` y que el mensaje de respuesta indique que no se encontró ningún usuario con las credenciales proporcionadas.



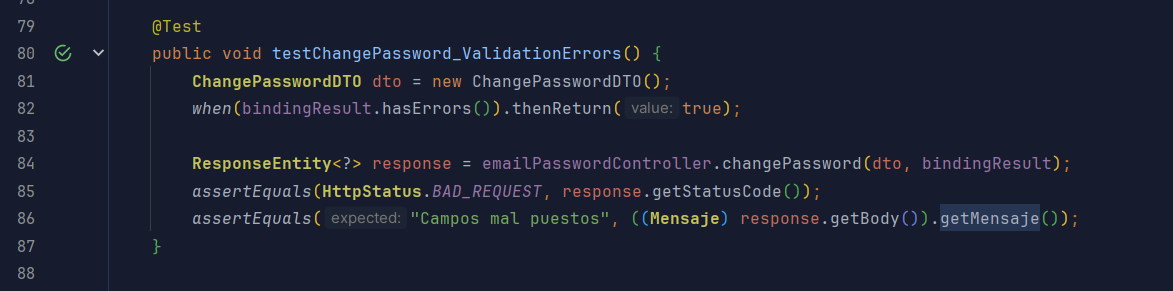
### Verificación de envío de correo exitoso (EmailPasswordController)

La prueba `testSendEmailTemplate\_Success` valida que el método `sendEmailTemplate` del `EmailPasswordController` funcione correctamente cuando se encuentra un usuario con el correo electrónico proporcionado. Se simula que el `UserService` encuentra un usuario con el correo electrónico dado y que el `EmailService` envía el correo electrónico. Se verifica que la respuesta tenga el estado HTTP `OK` y que el mensaje de respuesta indique que el correo electrónico ha sido enviado. Además, se confirma que el `EmailService` ha sido llamado para enviar el correo electrónico.



### Verificación de cambios de contraseña con errores de validación (EmailPasswordController)

La prueba `testChangePassword\_ValidationErrors` valida que el método `changePassword` del `EmailPasswordController` maneje correctamente los casos en que hay errores de validación en los campos del `ChangePasswordDTO`. Se simula que el `BindingResult` reporta errores de validación. Se verifica que la respuesta tenga el estado HTTP `BAD\_REQUEST` y que el mensaje de respuesta indique que los campos están mal puestos.



### Verificación de cambio de contraseña cuando las contraseñas no coinciden (EmailPasswordController)

La prueba `testChangePassword\_PasswordsDoNotMatch` valida que el método `changePassword` del `EmailPasswordController` maneje correctamente el caso en que las contraseñas proporcionadas no coinciden. Se configura el `ChangePasswordDTO` con contraseñas que no coinciden y se verifica que la respuesta tenga el estado HTTP `BAD\_REQUEST` y que el mensaje de respuesta indique que las contraseñas no coinciden.



### Verificación de cambio de contraseña cuando el usuario no se encuentra (EmailPasswordController)

La prueba `testChangePassword\_UserNotFound` valida que el método `changePassword` del `EmailPasswordController` maneje correctamente el caso en que no se encuentra un usuario con el token de contraseña proporcionado. Se simula que el `UserService` no encuentra ningún usuario con el token dado. Se verifica que la respuesta tenga el estado HTTP `NOT\_FOUND` y que el mensaje de respuesta indique que no se encontró ningún usuario con las credenciales proporcionadas.

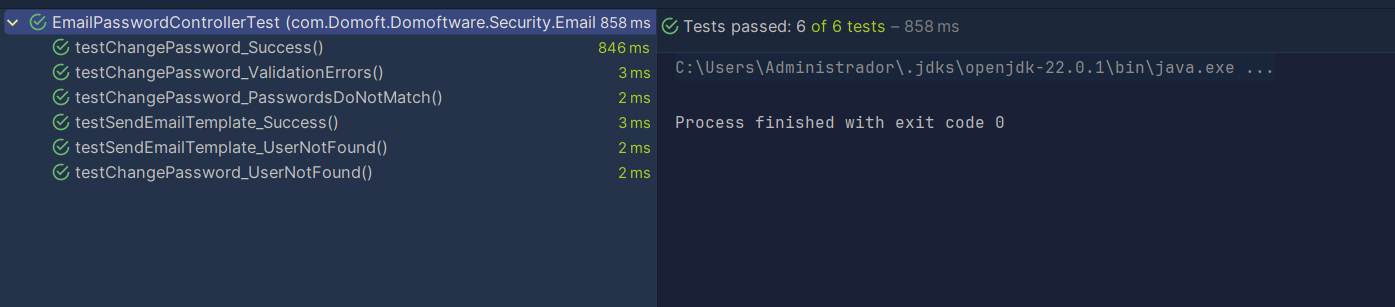


### Verificación de cambio de contraseña exitoso (EmailPasswordController)

La prueba `testChangePassword\_Success` valida que el método `changePassword` del `EmailPasswordController` funcione correctamente cuando el token de contraseña es válido y las contraseñas coinciden. Se simula que el `UserService` encuentra un usuario con el token dado y que el `PasswordEncoder` codifica la nueva contraseña correctamente. Se verifica que la respuesta tenga el estado HTTP `OK` y que el mensaje de respuesta indique que la contraseña ha sido actualizada. Además, se confirma que el `UserService` ha sido llamado para guardar el usuario con la nueva contraseña.



**Resultado de pruebas:**

****