



Información del Caso de Prueba 01

ID: TCP-REG-001

Versión: 1.0
Fecha: 2 Diciembre 2024
Autor: Equipo QA
Prioridad: Alta

1. Descripción General

Este caso de prueba verifica el proceso completo de registro de usuarios en el sistema de chat, incluyendo validaciones de campos, almacenamiento en base de datos y envío de correo de confirmación.

2. Precondiciones

- Ambiente de pruebas configurado y operativo
- Base de datos PostgreSQL inicializada y accesible
- Servidor SMTP configurado para envío de correos
- Selenium WebDriver instalado y configurado

3. Datos de Prueba

```
test_data = {  
    "nombre": "Juan Pérez",  
    "password": "Test2024#",  
    "confirm_password": "Test2024#" }  
}
```

```
invalid_test_data =  
    [{"password": "123", "expected_error": "Contraseña debe tener al menos 8 caracteres"},  
    {"confirm_password": "diferente", "expected_error": "Las contraseñas no coinciden"} ]
```

4. Script de Automatización

```
from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected_conditions as EC  
import pymongo  
import unittest  
  
class RegistroUsuarioTest(unittest.TestCase):  
    def setUp(self):  
        self.driver = webdriver.Chrome()  
        self.driver.get("<https://chat-system.test/registro>")  
        self.mongo_client = pymongo.MongoClient("mongodb://localhost:27017/")  
        self.db = self.mongo_client["chat_db"]  
  
    def test_registro_exitoso(self):  
        # Completar formulario  
        self.driver.find_element(By.ID, "nombre").send_keys(test_data["nombre"])
```

```

self.driver.find_element(By.ID, "email").send_keys(test_data["email"])
self.driver.find_element(By.ID, "password").send_keys(test_data["password"])
self.driver.find_element(By.ID,
"confirm_password").send_keys(test_data["confirm_password"])

# Enviar formulario
self.driver.find_element(By.ID, "submit-btn").click()

# Verificar mensaje de éxito
success_message = WebDriverWait(self.driver, 10).until(
    EC.presence_of_element_located((By.CLASS_NAME, "success-message"))
)
self.assertTrue(success_message.is_displayed())

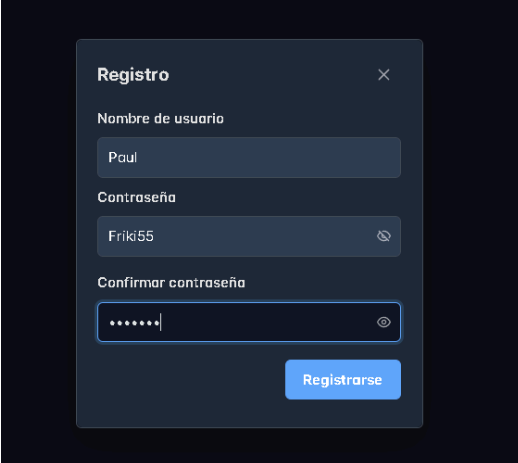
# Verificar registro en BD
user = self.db.users.find_one({"email": test_data["email"]})
self.assertIsNotNone(user)

def tearDown(self):
    self.driver.quit()
    self.mongo_client.close()

```

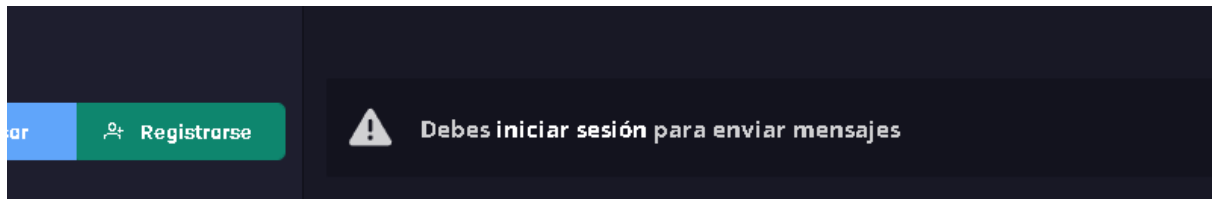
5. Pasos de Prueba Manual

1. Acceder a la URL de registro: <http://localhost:4200/>
2. Verificar que todos los campos del formulario estén presentes:
 - Campo de nombre
 - Campo de contraseña
 - Campo de confirmación de contraseña

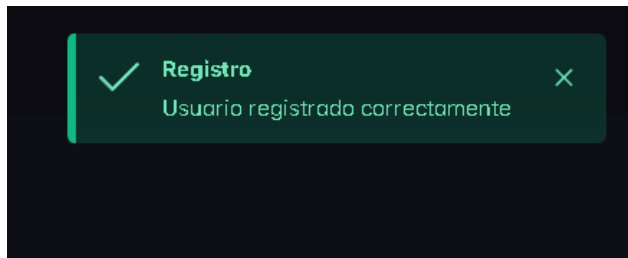


The image shows a registration form titled "Registro" with a close button (X) in the top right corner. The form contains three input fields: "Nombre de usuario" with the value "Paul", "Contraseña" with the value "Friki55", and "Confirmar contraseña" with masked characters "*****". There is a blue button labeled "Registrarse" at the bottom right of the form.

3. Ingresar los datos de prueba especificados
4. Hacer clic en el botón "Registrar"



5. Verificar mensaje de confirmación en pantalla



6. Validaciones

Campo	Regla de Validación	Mensaje de Error
Nombre	Mínimo 3 caracteres	"Nombre debe tener al menos 3 caracteres"
Contraseña	Mínimo 8 caracteres, incluir números y símbolos	"Contraseña debe cumplir requisitos de seguridad"

7. Resultados Esperados

- Usuario creado en base de datos con datos correctos

	Id [PK] bigint	user_preferences_id bigint	username character varying (255)
1	1	1	admin
2	2	2	Paul
3	3	3	Andres

- Contraseña almacenada con hash bcrypt

Showing rows: 1

password	
text	
\$2a\$10\$1AZ0VWcjRVuD5J/zTivMSOU1KuQ3tWmDW8RAJFcybHfdCz0z...	
\$2a\$10\$WTGBMq2cRocH9QTBfaF0BeVrm1vnV2K1UIQbo1U0eedjIQIpou...	
\$2a\$10\$TyKbizuqsEAL4EQIX5wmN.xMiAzd6oJoVxqe8CunkrAxuAZRNI5...	

8. Monitoreo y Logs

ActivityStateConfigurationLogsSystem

Sessions

☐ Active sessions only

Search

			PID	User	Application	Client	Backend start	Transaction start	State	Wait event
✖	■	>	2424	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:49 -05		idle	Client: ClientRead
✖	■	>	4308	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:10 -05		idle	Client: ClientRead
✖	■	>	5592	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:57 -05		idle	Client: ClientRead
✖	■	>	8716	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:27 -05		idle	Client: ClientRead
✖	■	>	16320	postgres	pgAdmin 4 - DB:hablen...	:::1	2024-12-02 22:47:17 -05	2024-12-02 23:50:58 -05	active	
✖	■	>	16652	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:01 -05		idle	Client: ClientRead
✖	■	>	17412	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:25 -05		idle	Client: ClientRead
✖	■	>	17484	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:00 -05		idle	Client: ClientRead
✖	■	>	18572	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:48:00 -05		idle	Client: ClientRead
✖	■	>	19140	postgres	pgAdmin 4 - CONN:504...	:::1	2024-12-02 23:48:06 -05		idle	Client: ClientRead
✖	■	>	20028	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:49 -05		idle	Client: ClientRead
✖	■	>	20376	postgres	PostgreSQL JDBC Driver	127.0.0.1	2024-12-02 23:47:58 -05		idle	Client: ClientRead

Locks

9. Criterios de Aceptación

Criterio	Métrica	Estado
Tiempo de registro	< 3 segundos	[]
Validaciones de campos	100% funcionales	[]
Envío de correo	< 5 segundos	[]
Seguridad de contraseña	Hash bcrypt verificado	[]

Notas de Implementación

- Ejecutar pruebas en ambiente aislado
- Limpiar datos de prueba después de cada ejecución