Instituto Politécnico Nacional
Escuela Superior de Cómputo (ESCOM)

Ingenieria de software

Practica 5. Plan de pruebas

Torres Ruiz Axel Alejandro
Boleta: 2022630178
Grupo: 7CV3

Junio de 2025

Contents

1 Estrategia General de Pruebas

El proyecto emplea un enfoque de múltiples capas para garantizar la calidad de código y la estabilidad de la aplicación. Incluye:

1.1 Pruebas Unitarias

- **Service Layer:** Pruebas aisladas de la lógica de negocio en servicios, usando Mockito para mockear repositorios.
- **Controller Layer (MVC & REST):** Pruebas de controladores Spring MVC y REST, simulando peticiones HTTP con MockMvc y validando códigos de estado y payloads JSON.

1.2 Pruebas de Integración

- **Component Interaction:** Verifica la colaboración entre controllers, services y repositories con el contexto Spring completo.
- **Authentication & Security:** Testea flujos de registro, login, logout y control de acceso por roles.
- **Full Application Context:** Carga básica del contexto Spring Boot con @SpringBootTest.

1.3 Pruebas de Integración Especializadas

- `SecurityTestIT.java`: Verificaciones de configuraciones de seguridad.
- `StressTestIT.java`: Evaluación de rendimiento y estabilidad bajo carga.

1.4 Características Clave

- **Aislamiento:** Uso extenso de Mockito en pruebas unitarias.
- **Ecosistema Spring:** Spring Test, Spring Boot Test, Spring Security Test.
- **Automatización:** JUnit 5 para ejecución en CI/CD.
- **Cobertura:** Desde lógica de negocio hasta flujos completos y seguridad.

2 Categorías de Pruebas y Descripciones

2.1 2.1 Controller Tests

- **Ubicación:** `src/test/java/com/escom/papelio/controller/` (MVC)
- **Objetivo:** Validar peticiones web, interacción con servicios mockeados y vistas.
- **Herramientas:** JUnit 5, Mockito, Spring Test (MockMvc).
- **Clases:** AdminControllerTest, AuthControllerTest, SearchControllerTest, UserControllerTest.

2.2 2.2 REST Controller Tests

- **Ubicación:** `src/test/java/com/escom/papelio/controller/` (REST)
- **Objetivo:** Validar endpoints REST, JSON, status codes.
- **Herramientas:** JUnit 5, Mockito, MockMvc, JSONPath.
- **Clases:** AdminRESTControllerTest, SearchRestControllerTest, UserRestControllerTest.

2.3 2.3 Service Tests

- **Ubicación:** `src/test/java/com/escom/papelio/service/`
- **Objetivo:** Pruebas unitarias de lógica de negocio con dependencias mockeadas.
- **Herramientas:** JUnit 5, Mockito.
- **Clases:** AdminServiceTest, ArticleFavoriteServiceTest, ArticleViewHistoryServiceTest, SearchHistoryServiceTest, SemanticScholarServiceTest, UserServiceTest.

2.4 2.4 Integration Tests

- **Ubicación:** `src/test/java/com/escom/papelio/integration/`
- **Objetivo:** Interacción de múltiples componentes con contexto real.
- **Herramientas:** JUnit 5, Spring Boot Test, MockMvc, Spring Security Test.
- **Clases:** AuthenticationFlowIntegrationTest, SearchIntegrationTest, SecurityTestIT, StressTestIT.

2.5 2.5 Application Context Tests

- **Ubicación:** `src/test/java/com/escom/papelio/`
- **Objetivo:** Verificar carga del contexto con papelioApplicationTests.java.

3 Herramientas y Tecnologías de Prueba

- JUnit 5
- Mockito
- Spring Test (MockMvc, @SpringBootTest, @AutoConfigureMockMvc)
- Spring Security Test
- Hamcrest Matchers
- JSONPath
- Jackson ObjectMapper

- Maven Surefire/Failsafe
- SQL Scripts con @Sql

4 Recursos del Plan de Pruebas

- Datos de prueba en métodos @BeforeEach y SQL de ejemplo en src/test/resources/sql/
- Configuración en application-test.properties para base en memoria

5 Resultados de StressTestIT

Logs ejecutados el 2 de mayo de 2025, incrementando de 10 a 200 usuarios concurrentes en pasos de 10, 15 peticiones por hilo.

Table 1: Estadísticas de rendimiento por número de hilos

Hilos	**Regs**	**Logins**	**Dash**	**Avg (ms)**	**Max (ms)**
10	150	150	150	119	508
20	450	450	450	160	633
30	900	900	900	212	913
40	1500	1500	1500	266	1308
50	2250	2250	2250	319	1308
60	3150	3150	3150	375	1662
70	4200	4200	4200	431	1821
80	5400	5400	5400	487	2747
100	8250	8250	8250	596	3396
110	9900	9900	9900	651	3510
120	11700	11700	11700	744	7451
130	13650	13650	13650	828	7451
140	15750	15750	15750	880	7451
150	18000	18000	18000	923	7451
160	20400	20400	20400	970	7451
170	22950	22950	22950	1034	9030
180	25650	25650	25650	1079	9030
190	28500	28500	28500	1124	9030
200	31500	31500	31500	1166	9030

6 Conclusiones y Aprendizajes sobre QA

La experiencia de diseñar e implementar un plan de pruebas integral permitió comprender en profundidad la importancia del aseguramiento de la calidad (QA) en el desarrollo de software. QA no es una etapa aislada, sino una práctica continua que permea todo el ciclo de vida del desarrollo, desde la definición de requisitos hasta la entrega final del producto.

Uno de los principales aprendizajes es que las pruebas unitarias son fundamentales para detectar errores tempranos y facilitar el mantenimiento del código. Permiten validar la lógica de negocio en aislamiento y, gracias a herramientas como Mockito, es posible simular comportamientos complejos sin necesidad de acceder a bases de datos o servicios externos.

Las pruebas de integración, por otro lado, permiten evaluar la interacción real entre componentes. Gracias a Spring Boot Test y MockMvc, fue posible simular flujos completos y validar respuestas HTTP, autenticación y seguridad, lo que garantiza una mayor confiabilidad del sistema.

Asimismo, la inclusión de pruebas de carga y estrés reveló cuellos de botella que podrían pasar desapercibidos en entornos de prueba tradicionales. Estos resultados justifican la necesidad de realizar pruebas bajo condiciones similares a producción para anticipar posibles fallos.

Finalmente, se reafirma que invertir en QA no solo reduce el número de errores en producción, sino que también mejora la experiencia del usuario, reduce costos a largo plazo y contribuye a la construcción de software robusto, confiable y mantenible. QA no es un lujo, sino una necesidad estratégica en cualquier proyecto serio de ingeniería de software.