Informe de conocimientos previos de testing para un WIS Grupal – D01



Grupo: *C2.010*

Repositorio: [*Acme-ANS*](https://github.com/ASoult10/Acme-ANS)

(https://github.com/ASoult10/Acme-ANS)

Miembros:

* **Mario Benítez Galván (**[marbengal@alum.us.es](mailto:marbengal@alum.us.es)**)**
* **Enrique Nicolae Barac Ploae (**[enrbarplo@alum.us.es](mailto:enrbarplo@alum.us.es)**)**

Fecha: ***19/02/2025***

# Tabla de contenidos

[Tabla de contenidos 1](#_Toc190974380)

[1. Resumen Ejecutivo 2](#_Toc190974381)

[2. Tabla de versiones 3](#_Toc190974382)

[3. Introducción 4](#_Toc190974383)

[4. Conocimiento previo sobre el testing de WIS 5](#_Toc190974384)

[5. Plan de Testing anteriores 6](#_Toc190974385)

[6. Estrategia de pruebas 7](#_Toc190974386)

[7. Conclusiones 8](#_Toc190974387)

[8. Bibliografía 9](#_Toc190974388)

# 1. Resumen Ejecutivo

Como estudiantes de Ingeniería Informática, hemos tenido experiencias previas con pruebas de software en distintas asignaturas, lo que nos ha permitido familiarizarnos con conceptos generales de testing. Sin embargo, antes de abordar este tema, nuestro conocimiento sobre la evaluación de un Sistema de Información Web (WIS, por sus siglas en inglés) era limitado.

Si bien hemos trabajado con herramientas y metodologías de prueba en aplicaciones convencionales, no contábamos con una comprensión profunda de los enfoques específicos para evaluar la funcionalidad, seguridad, rendimiento y usabilidad de un WIS. Este informe busca reflejar nuestro punto de partida antes de cursar esta materia, destacando lo que sabíamos y las lagunas de conocimiento que esperamos llenar. Se presenta una visión detallada sobre el punto de partida del grupo en términos de testing, la estrategia de prueba a seguir durante el proyecto junto con el plan de testing aplicado en el desarrollo y mantenimiento de estos sistemas.

El propósito de este informe es plasmar nuestro nivel actual de experiencia y comprensión en el desarrollo de un WIS, estableciendo un punto de partida que nos permita medir nuestro progreso a lo largo del curso. Asimismo, funcionará como un recurso comparativo para evaluar el crecimiento de nuestros conocimientos en esta área.

Conforme avancemos en la asignatura, iremos confrontando nuestras percepciones iniciales con los nuevos aprendizajes adquiridos, lo que nos facilitará una comprensión más profunda y la adopción de enfoques más efectivos y sólidos en la creación de Sistemas de Información Web. Este ejercicio no solo reforzará nuestro aprendizaje, sino que también potenciará nuestras competencias para afrontar futuros retos en el ámbito tecnológico.

# 2. Tabla de versiones

|  |  |  |
| --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** |
| 18/02/2025 | 0.0 | Creación del documento plantilla |
| 19/02/2025 | 1.0 | Clonación de plantilla para realizar el documento |
| 19/02/2025 | 2.0 | Realización del documento |

# 3. Introducción

El presente informe tiene como objetivo establecer una base inicial sobre nuestro conocimiento en el ámbito del testing de Sistemas de Información Web (WIS). Dado que hasta el momento nuestra experiencia en pruebas de software se ha centrado en aplicaciones convencionales, es fundamental identificar nuestras fortalezas y áreas de mejora antes de profundizar en metodologías específicas para la evaluación de estos sistemas. A través de este documento, buscamos no solo reconocer nuestro nivel de comprensión actual, sino también sentar las bases para un análisis comparativo que nos permita medir el progreso alcanzado a lo largo de la asignatura.

# 4. Conocimiento previo sobre el testing de WIS

Contamos con experiencia previa en distintas metodologías y prácticas de prueba adquiridas en asignaturas anteriores. Hemos trabajado con pruebas unitarias, aplicando la estrategia *Arrange-Act-Assert* para diseñar tanto casos positivos como negativos, asegurando que cada unidad de código se comporte según lo esperado en diferentes escenarios. Para ello, hemos utilizado frameworks de pruebas unitarias, lo que nos ha permitido estructurar y ejecutar los tests de manera eficiente y automatizada.

También hemos aplicado pruebas específicas en el contexto de Spring Validators, verificando que las validaciones de datos cumplan con los requisitos establecidos. Además, hemos empleado test *doubles*, como *mocks*, *stubs* y *spies*, para simular el comportamiento de componentes externos y aislar el código en pruebas unitarias. En este sentido, Mockito ha sido una de las herramientas principales para la creación de mocks, facilitando la validación del comportamiento tanto en repositorios como en servicios.

Nuestra experiencia también incluye pruebas de integración, asegurando la correcta comunicación entre distintos módulos de la aplicación.

Sin embargo, aunque hemos trabajado con pruebas *end-to-end* (*E2E*) y pruebas de aceptación, estas han sido las menos exploradas en comparación con otros tipos de testing. No obstante, entendemos su importancia en la validación del sistema completo desde la perspectiva del usuario.

Durante todas estas pruebas, hemos seguido los principios DRY (Don't Repeat Yourself) para evitar redundancias en el código de test y DAMP (Descriptive And Meaningful Phrases) para asegurar que los casos de prueba sean comprensibles y mantenibles.

# 5. Plan de Testing anteriores

Durante el desarrollo de los sistemas con los que se ha trabajado con anterioridad, se garantizó que cada componente funcionara adecuadamente y cumpliera con los requisitos establecidos. Al seguir este enfoque estructurado que se detalla a continuación, logramos identificar y corregir errores de manera efectiva, lo que contribuyó a la mejora continua de nuestras prácticas de testing y aseguró la satisfacción del usuario final.

*Entorno y herramientas:*

Anteriormente, el testing del WIS se llevó a cabo en un entorno controlado que reflejó la configuración del sistema en producción, lo que garantizó la validez de las pruebas. Utilizamos herramientas de pruebas ampliamente reconocidas en la industria, como JUnit para las pruebas unitarias y Mockito para la creación de mocks. Además, implementamos Postman para validar APIs y servicios web. La configuración de estas herramientas nos permitió realizar pruebas exhaustivas y eficaces, asegurando que cada funcionalidad del sistema se evaluara correctamente antes de su lanzamiento.

*Criterios de finalización:*

La finalización del proceso de testing en proyectos anteriores se determinó en base a varios criterios predefinidos. Se consideró que el testing era suficiente cuando se logró una cobertura de pruebas del 80% del código, lo que indicaba que la mayoría de las funcionalidades habían sido evaluadas. Se usó la herramienta *JaCoCo* para la medición de la misma. Asimismo, todos los defectos críticos debían ser corregidos y verificados antes de la entrega final del sistema. La validación de los criterios de aceptación establecidos en las pruebas funcionales también fue un factor clave en la determinación de la finalización, asegurando que el sistema cumpliera con los requisitos del cliente y proporcionara una experiencia positiva al usuario.

*Gestión de incidencias:*

La gestión de incidencias fue un componente esencial del proceso de testing en nuestros proyectos anteriores, asegurando que cualquier error o fallo identificado durante las pruebas fuera documentado y abordado de manera eficaz. Utilizamos herramientas de seguimiento como Jira o Trello para registrar y priorizar las incidencias, categorizándolas según su gravedad: bloqueantes, críticos o menores. El equipo evaluó cada incidencia y asignó responsabilidades para su resolución, garantizando que se abordaran en un plazo razonable. Este enfoque nos permitió mantener un control adecuado sobre los problemas y asegurar que no afectaran la calidad final del WIS.

# 6. Estrategia de pruebas

A continuación, se detallan los tipos de pruebas conocidos previamente en asignaturas y proyectos anteriores.

Pruebas unitarias:

* Se implementaron siguiendo la metodología *Arrange-Act-Assert*.
* Se evaluaron tanto casos positivos como negativos.
* Se empleó Mockito para simular dependencias en repositorios y servicios.
* Se aplicaron test doubles (*mocks* y *stubs*) para aislar componentes.

Pruebas de integración:

* Se verificaron las interacciones entre distintos módulos del WIS.
* Se probaron la comunicación con la base de datos y APIs externas.
* Se validó la correcta configuración de contexto de Spring.

Pruebas *End-to-End*:

* Se simularon escenarios completos de usuario desde el inicio hasta el final del proceso.

Pruebas de aceptación:

* Se definieron criterios de aceptación basados en los requisitos funcionales.

Aunque esta es la estrategia de pruebas con la que partimos, conforme se vaya avanzando en la asignatura y ampliemos nuestros conocimientos, se incluirán más pruebas y se mejorará la calidad de las ya implementadas.

# 7. Conclusiones

Este informe ha establecido un marco sólido para comprender nuestro nivel actual de conocimiento en testing de Sistemas de Información Web (WIS). A través de la identificación de nuestras experiencias previas y la presentación de un Plan de Testing detallado, hemos sentado las bases para un enfoque sistemático que nos permitirá evaluar la calidad y funcionalidad del sistema a medida que avanzamos en el desarrollo.

La gestión de incidencias y la definición de criterios de finalización son aspectos clave que contribuirán a la mejora continua de nuestras prácticas de testing, permitiéndonos abordar cualquier problema de manera efectiva y mantener un alto estándar de calidad. A medida que avanzamos en la asignatura, nos comprometemos a ampliar nuestras habilidades y conocimientos, incorporando nuevas técnicas de testing y mejorando continuamente los enfoques existentes. Esto no solo beneficiará nuestro aprendizaje, sino que también garantizará la satisfacción del usuario final en el desarrollo de este Sistema de Información Web.

# 8. Bibliografía

* [*06 – Annexes.dock*](https://ev.us.es/bbcswebdav/pid-5227970-dt-content-rid-62285040_1/xid-62285040_1)– Project Statement, Asignatura de Diseño y Pruebas II del grado de Ingeniería Informática del Software, Universidad de Sevilla.
* Recursos de asignaturas de cursos pasados en el grado de Ingeniería Informática del Software (AISS, IISSI 1, DP 1).