Plan de Pruebas

**Testify**

OSLO

Ojeda Valeria – Sly Eduardo

Levipichun Emilio – Oyarzo Malena



El propósito del plan de pruebas es explicitar el alcance, enfoque, recursos requeridos, calendario, responsables y manejo de riesgos de  un proceso de pruebas.

Este plan de Pruebas está contemplado dentro del plan SQA para un proyecto dado

Tabla de contenido

[Introducción: 4](#_Toc177859457)

[Antecedentes y propósito 5](#_Toc177859458)

[Propósito de la evaluación 5](#_Toc177859459)

[Objetivos 5](#_Toc177859460)

[Motivadores 6](#_Toc177859461)

[Ámbito de las pruebas 6](#_Toc177859462)

[Dentro del ámbito 6](#_Toc177859463)

[Funcionales 7](#_Toc177859464)

[Usabilidad 7](#_Toc177859465)

[Compatibilidad 7](#_Toc177859466)

[Seguridad 7](#_Toc177859467)

[Rendimiento 8](#_Toc177859468)

[Integración 8](#_Toc177859469)

[Regresión 8](#_Toc177859470)

[Mantenimiento 8](#_Toc177859471)

[Fuera del ámbito 8](#_Toc177859472)

[Checklist 9](#_Toc177859473)

[Lista de Ideas de las Pruebas 11](#_Toc177859474)

[Enfoque de las Pruebas 11](#_Toc177859475)

[Pruebas Funcionales 12](#_Toc177859476)

[Pruebas de Integración 12](#_Toc177859477)

[Pruebas de Seguridad 12](#_Toc177859478)

[Pruebas de Rendimiento 12](#_Toc177859479)

[Pruebas de Usabilidad 13](#_Toc177859480)

[Pruebas de Compatibilidad 13](#_Toc177859481)

[Pruebas de Regresión 13](#_Toc177859482)

[Pruebas de Mantenimiento 13](#_Toc177859483)

[Herramientas para las Pruebas 13](#_Toc177859484)

[Software 13](#_Toc177859485)

[Herramientas de Soporte y Productividad 14](#_Toc177859486)

[Secuencias de Comandos Personalizadas (Script de Pruebas) 14](#_Toc177859487)

[Hardware 14](#_Toc177859488)

Plan de Pruebas

Introducción:

Testify es una aplicación web que permite llevar a cabo el registro, administración y control de los casos de pruebas creados por los usuarios para proyectos de desarrollo de software con el objetivo de que se realicen las pruebas necesarias, en el presente plan de pruebas se revisará que la aplicación cumpla con los requisitos expedidos por el cliente, además también que cada elemento/modulo, función e integración cumpla con los criterios de aceptación que son requeridos para certificar que el software esté listo para pasar a un ambiente productivo mediante la correspondiente validación de las pruebas para garantizar la calidad en el producto final.

El resultado de la ejecución del presente plan de pruebas afecta a las categorías de operación (OPE), de gerencia (GER) y de alta dirección (DIR), a continuación se detalla la intervención de cada nivel categórico y jerárquico.

Durante la ejecución de las pruebas si se detectasen fallos se documentaran en un informe detallado para ello se asignaran las correcciones pertinentes a la categoría de operación (OPE) donde los técnicos y desarrolladores evalúan, planifican y ejecutan las correcciones que implican:

1. Revisar el código y/o componentes.
2. Implementación de las correcciones necesarias basadas en la revisión.
3. Volver a ejecutar la prueba para verificar que el fallo se corrigió.
4. Generar informes/documentación correspondiente.

Posteriormente el área de gerencia (GER) debe revisar los informes de correcciones y asegurar que las pruebas se repiten para verificar que la aplicación está funcionando correctamente.

Al finalizar se debe dar una revisión por parte de la alta dirección (DIR) de forma periódica el estado del proceso de pruebas informando si existen problemas críticos relacionados/nuevos que puedan afectar la entrega del proyecto.

Este proceso debe repetirse en forma de ciclo continuo que involucran: pruebas, correcciones y re-ejecución hasta que todas las pruebas sean superadas sin fallos garantizando que el producto final Testify esté listo para ser entregado avalado por procesos de calidad.

Antecedentes y propósito

El desarrollo de Testify una aplicación web para el registro, administración y control de casos de pruebas en proyectos de desarrollo para la cátedra de laboratorio de desarrollo de software será utilizando JAVA para el backend y Angular para el frontend. Java es un lenguaje ampliamente utilizado para la creación de aplicaciones robustas, mientras que Angular es un framework de frontend que permite la creación de interfaces de usuario interactivas y dinámicas.

El propósito principal de las pruebas es crear una página web eficiente y accesible para actividades contempladas dentro de un entorno universitario, dado que no se cuenta con antecedentes previos, se han tomado en cuenta las mejores prácticas y estándares aprendidas e investigadas para asegurar que el producto final cumpla con los requisitos esperados de rendimiento, usabilidad y accesibilidad.

Propósito de la evaluación

La calidad del software se mide por su capacidad para cumplir con los requisitos establecidos, tanto funcionales como no funcionales y ofrecer una experiencia de usuario satisfactoria. El presente plan de pruebas tiene como propósito realizar pruebas para garantizar que la aplicación web Testify sea entregado al cliente con un alto nivel de calidad.

El plan de pruebas es esencial para garantizar que se realicen las actividades necesarias durante el ciclo de vida del proyecto, desde el desarrollo hasta la entrega final, estas pruebas estarán enfocadas en que los componentes del sistema funcionen según lo previsto y que cumplan con las especificaciones del cliente.

Objetivos

Los objetivos que se pretenden alcanzar con la ejecución de este plan de pruebas son los siguientes:

* Realizar pruebas sobre la interfaz de usuario verificando no solo su funcionalidad, sino también su compatibilidad en diferentes dispositivos y navegadores, con especial atención a la integración de **Angular 18** y **Bootstrap 5.3**.
* Verificar el cumplimiento de especificaciones de diseño y estándares visuales establecidos por la universidad asegurando que el diseño se alinee con los principios de accesibilidad y usabilidad, cumpliendo las guías **WCAG 2.1**.
* Supervisar el cumplimiento de los requisitos funcionales definidos durante la fase de análisis y diseño del proyecto asegurando que estén trazados de manera clara con los casos de uso y cubran el ciclo completo de las pruebas funcionales.
* Realizar pruebas de rendimiento y capacidad para garantizar que la aplicación responda adecuadamente bajo diferentes niveles de carga.
* Verificar que la navegación por la interfaz de usuario sea intuitiva y sin errores evaluando la organización y disposición de los elementos visuales según las mejores prácticas de diseño con **Angular** y **Bootstrap**.
* Asegurar mediante pruebas de aceptación de usuario que se el sistema cumple con sus expectativa considerando tanto la funcionalidad como la experiencia de usuario en su interacción con el sistema.

Motivadores

Los principales motivadores que impulsan la realización de este plan de pruebas incluyen:

* **Validar el comportamiento funcional de la interfaz de usuario**, incluyendo la interacción correcta de componentes como botones, campos de texto, checkboxes, listas, gráficos dinámicos (basados en **Ng2-charts** y **Chart.js**) y vistas por roles, garantizando que las restricciones de acceso de **Spring Security** funcionen adecuadamente.
* **Revisión del formato visual** para garantizar una presentación profesional y consistente con las normas establecidas por la universidad, asegurando que las guías de estilo de **Bootstrap** se respeten en todo momento.
* **Cumplimiento de los requerimientos funcionales y no funcionales**, incluyendo no solo la funcionalidad básica, sino también requisitos de rendimiento, seguridad y accesibilidad, destacando la importancia de las **pruebas de seguridad** y de **autenticación OAuth2**.
* **Adaptación a los cambios en los requerimientos** que surjan durante el ciclo de vida del proyecto, verificando que el sistema sea lo suficientemente flexible para aceptar modificaciones sin comprometer la calidad del software.
* **Verificación de que la interfaz de usuario cumpla con las mejores prácticas de diseño y navegación fluida**, garantizando que la organización de los elementos visuales y el flujo de interacción entre pantallas sean consistentes y optimizados para una experiencia de usuario eficiente.

Ámbito de las pruebas

El presente plan de pruebas debe asegurar que la aplicación web Testify cumpla con los requerimientos funcionales y no funcionales definidos en el documento de requerimientos, y que **el diseño del sistema este correctamente implementado en el código**. Las pruebas se realizaran con el objetivo de verificar la calidad del sistema asegurando la trazabilidad entre los requerimientos, el diseño y el código.

Dentro del ámbito

Las pruebas están diseñadas para verificar que el sistema implementa correctamente los requerimientos descritos en el documento de requerimientos y que estos fueron aprobados, trazados y que se cumple con ellos, para ello se deberán probar mediante **revisiones técnicas formales (RTF)** las siguientes características:

### Funcionales

1. Verificar que las funcionalidades implementadas cumplan con los requerimientos definidos.
2. Comprobar que se pueda navegar por la aplicación de manera intuitiva y fluida.
3. Revisar que todos los formularios y campos de entradas como imputs, textbox, comboboxs, listas aceptan y validan los datos correctamente.
4. Asegurar que el frontend basado en Angular se comunique correctamente con backend (JAVA), garantizando que las solicitudes y respuestas sean adecuadas.
5. Validar que el sistema gestione adecuadamente los errores y las respuestas sean adecuadas.
6. Garantizar que las sesiones de usuario sean correctamente gestionadas, manteniendo la seguridad e integridad de la aplicación.

Usabilidad

Empleando métricas de aprendizaje y midiendo el tiempo que los usuarios tardan en aprender a usar la aplicación de manera efectiva mediante la observación se realizaran evaluaciones heurísticas, es decir cuántos problemas/barreras encuentran los usuarios.

1. Se debe verificar que los elementos de la interfaz estén bien organizados y sean fáciles de usar.
2. Se debe asegurar que se cumpla con los estándares de accesibilidad, ya sean colores o fuentes.
3. Se debe validar que la aplicación se vea y funcione correctamente en diferentes versiones de navegadores y sistemas operativos.

Compatibilidad

1. Verificando que la aplicación funcione correctamente en distintos sistemas operativos, navegadores y dispositivos.

Seguridad

Empleando métricas para la tasa de vulnerabilidades criticas detectadas midiendo el número encontrado, análisis estático del código que generaran fallos de seguridad.

1. Probar que el sistema maneja de manera segura la autenticación y que solo los usuarios autorizados puedan acceder a ciertas funcionalidades o datos.
2. Comprobar que la aplicación sea resistente a inyecciones de datos SQL o Scripting.
3. Asegurar que los datos sensibles como información personal estén correctamente protegidos.

Rendimiento

1. Evaluando cómo responde el sistema cuando es sometido a altos niveles de tráfico o carga de usuarios. Se mide la capacidad de respuesta bajo carga y la tasa de solicitudes por segundo.

Integración

Empleando métricas para la frecuencia de fallos de integración de módulos, es decir cuántas veces los módulos del sistema fallan al integrarse después de ser modificados.

1. Verificar que los diferentes módulos del backend interactúen correctamente con el frontend.
2. Asegurar que las integraciones con API’s o servicios externos como bases de datos funcionen correctamente.

Regresión

Empleando métricas para la cobertura de pruebas de regresión midiendo el porcentaje de casos de prueba automatizados que cubren las funcionalidades, métricas para la tasa de defectos recurrentes, es decir el numero de defectos que se reintroducen en el sistema después de realizar cambios o actualizaciones, como así también métricas para el tiempo promedio de ejecución de las mismas pruebas de regresión, es decir cuánto tiempo tarda en completarse un ciclo de pruebas de regresión ayudando de esta forma a la eficiencia del proceso de pruebas.

1. Se debe garantizar que las funcionalidades previamente implementadas no se vean afectadas con nuevos cambios o actualizaciones de código.

Mantenimiento

1. Evalúan la facilidad con la que se pueden realizar cambios en el sistema, especialmente en el backend (Java/Spring Boot). Se mide el tiempo medio para realizar modificaciones.

Fuera del ámbito

En el presente plan de pruebas se estructuraran revisiones correspondientes a documentos/productos complementarios considerados “**claves**” para asegurar que los estándares de calidad establecidos en el Plan SQA realmente cumplan, para ello se emplearan dos técnicas de revisión basadas en Checklist para los entregables/productos rutinarios y las revisiones técnicas formales RTF al finalizar cada iteración, ambas serán llevadas a cabo por todo los integrantes del equipo de desarrollo OSLO.

Checklist

Esta técnica de revisión tendrá su propio anexo ubicado en el repositorio de “Garantía de calidad” del proyecto Testify, en el se irán anotando los productos considerados “**claves”** y otros productos considerados de “apoyo” con sus correspondientes datos: numero de error encontrado, el criterio de verificación, si fue chequeado no chequeado, integrante del equipo que realiza la revisión de rutina, la fecha, el tiempo invertido en la tarea, si fue resuelta o esta sin resolver o si fue notificado al administrador de SQA, y un apartado para las descripciones que deben incluir: **descripción, nivel de gravedad y ubicación.** La revisión de rutina contiene a los productos/entregables categorizados como “**clave**” con sus correspondientes pruebas, a continuación se hará mención de los documentos claves:

1. Especificación de Requerimientos de Software.
2. Modelo de Casos de Uso.
3. Modelo de diseño.
4. Arquitectura del Sistema.
5. Plan de pruebas.
6. Plan de riesgos.
7. Manual de Usuario.

A su vez cada revisión de los productos/entregables “**claves**” deben evaluar aspectos propios y únicos correspondientes a su naturaleza inherente, a continuación se detallaran cuales son estas actividades y como saber cuándo detectar un fallo/desviación/error en estos:

1. Especificación de Requerimientos de Software.
2. Descripción poco Funcional/comprensible.
3. Faltan RNF/RF/RN.
4. Verificabilidad de que no falten RF/RNF/RN.
5. Trazabilidad entre los documentos de requerimientos, diseño y código.
6. Prioridad.
7. Ortografía.
8. Estándar OSLO.
9. Modelo de Casos de Uso.
10. No anexado.
11. Inconsistencia CU-RF.
12. Ambigüedad.
13. Actor/CU no identificados.
14. Criterio de aceptación.
15. Ortográfico.
16. Estándar OSLO.
17. Modelo de diseño.
18. Modelo MVC.
19. Inconsistencia D-UML.
20. No se realiza RTF.
21. Seguridad.
22. Verificabilidad.
23. Ortográfico.
24. Estándar OSLO.
25. Arquitectura del Sistema.
26. Inconsistencia modularidad.
27. Incumplimiento de RNF.
28. Inconsistencia de estándar.
29. No se realiza RTF.
30. No es flexible.
31. Ortográfico.
32. Estándar OSLO.
33. Plan de pruebas.
34. Poca cobertura.
35. No alineado con RF.
36. Pruebas no diversificadas.
37. No se documenta prueba.
38. No se prioriza escenario critico.
39. Ortográfico.
40. Estándar OSLO.
41. Plan de riesgos.
42. Riesgo no identificado.
43. No existe contingencia.
44. No se realiza revisión-control.
45. No existe mitigación.
46. No se realiza re-evaluación.
47. Ortográfico.
48. Estándar OSLO.
49. Manual de Usuario.
50. No se realiza UAT.
51. No se realiza RTF de manual.
52. Inconsistencia del estándar.
53. Estructura poco intuitiva.
54. Faltan escenarios/flujos de trabajo.
55. Ortográfico.
56. Estándar OSLO.
57. Documentos de **apoyo/**secundarios.
58. En los documentos de apoyo/secundarios el encargado de la revisión debe especificar escribiendo el criterio de verificación empleado y el nombre del documento afectado a dicha actividad.

Como se establece en el Plan SQA todas las revisiones descritas en este apartado para actividades y productos/entregables se basan en métricas para el aseguramiento de calidad alineadas con normas ISO SQuaRE 25000, IEEE/ANSI 730.1 e ISO 9001.

Lista de Ideas de las Pruebas

Las pruebas serán identificadas siguiendo la técnica de generación de casos de prueba a través de los casos de uso, detallando los siguientes pasos:

Para cada caso de uso, se identifican los posibles caminos, estableciendo los escenarios (Técnica de análisis de camino).

Para cada uno de los caminos, se identifican los conjuntos de valores de entrada y precondiciones, al igual que el resultado esperado.

Se hace, a través de una tabla, un resumen por cada caso de uso que muestre los distintos caminos posibles con sus entradas y salidas.

Los recursos utilizados para la identificación de las pruebas se mencionan a continuación:

El documento de **especificación de requerimientos del software** para definir los escenarios funcionales claves.

El documento **de arquitectura de software** para obtener la información necesaria sobre la interacción entre componentes para las pruebas de integración.

Generación de pruebas de sistema a partir de la especificación funcional.

Mejora de la calidad de los requisitos mediante la generación de pruebas tempranas en prototipos evitando así problemas o ambigüedades en los requerimientos.

Especificación e implementación de casos de prueba.

Enfoque de las Pruebas

Los tipos de pruebas que se realizarán al software son:

1. Funcionales.
2. Usabilidad.
3. Seguridad.
4. Integración.
5. Regresión.

Pruebas Funcionales

1. **Pruebas de casos de uso:** Verifican que el sistema cumpla con todos los requisitos funcionales descritos. Incluir pruebas en prototipos tempranos permitirá identificar errores en los requisitos y refinar su especificación.
2. **Pruebas de aceptación del usuario (UAT)**: Se realizan con los usuarios finales para asegurarse de que el sistema cumple con sus expectativas. Estas pruebas pueden llevarse a cabo sobre prototipos para obtener retroalimentación inicial.
3. **Pruebas de navegación**: Verifican que la navegación por la interfaz de usuario sea intuitiva y sin errores.

Pruebas de Integración

1. **Pruebas de API**: Validan la correcta comunicación entre el frontend (Angular) y el backend (Java/Spring Boot). Se puede medir la tasa de éxito de llamadas a la API y el tiempo de respuesta.
2. **Pruebas de integración de base de datos**: Verifican la correcta interacción entre la aplicación y MySQL a través de Hibernate/JPA.
3. **Pruebas de integración continúa**: Realizadas cada vez que se actualiza el código para detectar posibles fallos en la integración de los distintos componentes del sistema (se mide la frecuencia de fallos en la integración).

Pruebas de Seguridad

1. **Pruebas de penetración**: Evalúan la resistencia del sistema ante ataques como inyecciones SQL o XSS (Scripting). Miden la tasa de vulnerabilidades críticas detectadas.
2. **Pruebas de autenticación y autorización**: Verifican la correcta implementación de Spring Security y OAuth2, asegurándose de que solo los usuarios autorizados puedan acceder a ciertas funcionalidades. La métrica puede ser el porcentaje de intentos de acceso no autorizado bloqueados.
3. **Pruebas de manejo de datos sensibles**: Validan que los datos personales estén correctamente protegidos y encriptados.

Pruebas de Rendimiento

1. **Pruebas de carga**: Evalúan cómo responde el sistema cuando es sometido a altos niveles de tráfico o carga de usuarios. Se mide la capacidad de respuesta bajo carga y la tasa de solicitudes por segundo.
2. **Pruebas de estrés**: Determinan el punto de fallo del sistema sometiéndolo a condiciones extremas.
3. **Pruebas de escalabilidad**: Verifican si el sistema es capaz de manejar un aumento en la carga y crecer de acuerdo a las demandas futuras.

Pruebas de Usabilidad

1. **Pruebas de usabilidad con usuarios**: Verifican que la interfaz de usuario sea fácil de usar y que los usuarios puedan completar tareas con éxito. La métrica sería el tiempo de aprendizaje o la tasa de errores del usuario.
2. **Evaluaciones de accesibilidad**: Verifican que la interfaz cumple con los estándares de accesibilidad (como WCAG 2.1).

Pruebas de Compatibilidad

1. **Pruebas multiplataforma**: Verifican que la aplicación funcione correctamente en distintos sistemas operativos, navegadores y dispositivos.
2. **Pruebas de visualización**: Validan que la interfaz de usuario basada en Bootstrap se vea y funcione correctamente en distintas resoluciones y dispositivos.

Pruebas de Regresión

**Pruebas de regresión automatizadas**: Garantizan que las funcionalidades existentes no se vean afectadas cuando se introducen nuevos cambios en el código. Se mide la cobertura de las pruebas automatizadas y el tiempo de ejecución de la suite de regresión (puede ser probando Testify, volviendo a ejecutar los casos de prueba).

Pruebas de Mantenimiento

**Pruebas de modificabilidad**: Evalúan la facilidad con la que se pueden realizar cambios en el sistema, especialmente en el backend (Java/Spring Boot). Se mide el tiempo medio para realizar modificaciones.

Herramientas para las Pruebas

Esta sección describe las herramientas utilizadas durante la prueba del sistema, organizadas en tres categorías: **Software**, **Herramientas de Soporte y Productividad**, y **Secuencias de Comandos Personalizadas (Script de Pruebas)**.

Software

Se utilizaran las siguientes herramientas de software para facilitar la instalación, configuración y prueba del sistema:

|  |  |  |
| --- | --- | --- |
| Nombre | Versión | Descripción |
| Postman | 10 | Herramienta para la prueba de API’s, que permite realizar peticiones HTTP/HTTPS y verificar respuestas. |
| Selenium | 4 | Framework de automatización para pruebas de interfaz de usuario en múltiples navegadores. |
| JMeter | 5 | Herramienta de pruebas de carga para medir el rendimiento del sistema bajo diferentes niveles de tráfico. |
| Jenkins | 2 | Herramienta de integración continua que automatiza la ejecución de pruebas. |
| JUnit | 5 | Marco de pruebas unitarias para aplicaciones JAVA, utilizado para verificar la lógica interna del backend. |

Herramientas de Soporte y Productividad

Durante las pruebas, se utilizaran las siguientes herramientas para supervisar y verificar el rendimiento y la consistencia del sistema:

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | Versión | Tipo de herramienta | Descripción |
| Grafana | 9 | Supervisión de rendimiento | Plataforma de análisis que permite visualizar métricas y alertas del sistema en tiempo real. |
| Prometheus | 2 | Monitoreo y alertas | Sistema de monitoreo de tiempo real que captura métricas del sistema, como uso de CPU, memoria y tiempos de respuestas. |
| DbVisualizer | 14 | Supervisión de base de datos | Permite realizar consultas, supervisar y verificar la coherencia de la bases de datos MySQL. |
| SonarQube | 9 | Análisis de calidad de código | Analiza el código fuente para detectar vulnerabilidades y problemas de calidad. |

Secuencias de Comandos Personalizadas (Script de Pruebas)

Durante las pruebas, se utilizaran los siguientes scripts personalizados para validar diversas funcionalidades en distintas fases del proyecto:

|  |  |  |
| --- | --- | --- |
| Script | Fase del proyecto | Descripción |
| Test\_Endpoints.sh | Pruebas de integración | Para realizar pruebas automáticas sobre los diferentes endpoints de la API. |
| LoadTest.jmx | Pruebas de rendimiento | Para simular un alto número de usuarios recurrentes con JMeter. |
| BackupTest.sqa | Pruebas de base de datos | Para validar la restauración y la creación de copias de seguridad en la base de datos MySQL. |
| UAT\_Automation.py | Pruebas de aceptación | Para automatizar las pruebas de aceptación de usuario en la interfaz de Angular mediante Selenium. |

Hardware

Los siguientes dispositivos físicos son los que se emplearan para la realización de pruebas:

| Recurso | Cantidad | Descripción |
| --- | --- | --- |
| Notebook | 1 | Nombre del dispositivo Bea Procesador AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz  RAM instalada 16,0 GB (15,3 GB utilizable), tipo de sistema operativo de 64 bits procesador  x64, con acceso a internet. |
| PC-Escritorio | 1 | Computadora AMD Ryzen 5 4600G with Radeon Graphics 3.70 GHz 32,0 GB (31,9 GB usable),  tipo de sistema operativo de 64 bits procesador x64, con acceso a internet. |
| Notebook | 1 | ASUS NOTEBOOK 14 R7 3700U 8GB 512GB, MSI Notebook GF63 11UC 15,6 I5 11400H 8GB  512SSD W10H. |
| Server | 1 | Server AMD Ryzen 3 32,0gb, 1TB disco, tipo de sistema operativo de 64 bits procesador x64,  con acceso a internet. |

Configuraciones de Pruebas de ambiente

| Recurso | Cantidad | Descripción |
| --- | --- | --- |
| Entorno de desarrollo | 1 | Nombre del dispositivo Bea Procesador AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz  RAM instalada 16,0 GB (15,3 GB utilizable), tipo de sistema operativo de 64 bits procesador  x64, con acceso a internet. |
| Entorno de pruebas QA | 1 | Computadora AMD Ryzen 5 4600G with Radeon Graphics 3.70 GHz 32,0 GB (31,9 GB usable),  tipo de sistema operativo de 64 bits procesador x64, con acceso a internet. |
| Entorno de producción simulado | 1 | ASUS NOTEBOOK 14 R7 3700U 8GB 512GB, MSI Notebook GF63 11UC 15,6 I5 11400H 8GB  512SSD W10H. |
| Entorno de pre-producción | 1 | Server AMD Ryzen 3 32,0gb, 1TB disco, tipo de sistema operativo de 64 bits procesador x64,  con acceso a internet. |

Casos de Prueba

A continuación se detalla la información importante mediante un ejemplo lo que deberán contener los documentos/entregables “caso de pruebas” artefacto adjunto al plan de pruebas que debe realizarse durante el desarrollo de la aplicación Testify.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID/Nombre/Sistema/Proyecto:** Testify | | | **Nivel de Prueba: 1** | | | |
| **ID Caso de Uso:** | | | **Tipo(s) de Pruebas(s):** Prueba de funcionalidad | | | |
| **ID Requerimiento:** | | | **Ambiente de Prueba:** Entorno de pruebas. | | | |
| **ID/Nombre Escenario:** ES-001F | | | **Autor del Caso de Prueba:** Valeria Ojeda | | | |
| **ID/Nombre Caso de Prueba:** CP01- | | | **Nombre del Probador:** Malena Oyarzo | | | |
| **Versión del Caso de Prueba:** CP-v1.0 | | | **Fecha de Creación:** 22/09/2024 | | **Fecha de Ejecución:** 24/09/2024 | |
| **Condición(es) para que se ejecute el Caso de Prueba:** | | | | | | |
| El usuario debe estar autenticado. | | | | | | |
| AMD Ryzen 5 4600G with Radeon Graphics 3.70 GHz | | | Software de auditorías antes mencionados | | | |
| Memoria RAM de 32,0 GB (31,9 GB usable) | | | Software para medir el rendimiento de la aplicación | | | |
| Software de soporte | | |  | | | |
| **Para la Ejecución del Caso de Prueba:** Contar con todas las herramientas citadas anteriormente | | | | | | |
| **Elemento a Probar** | **Condición** | **Valor(es)** | | **Resultado Esperado** | | **Resultado Obtenido** |
| Pantalla de bienvenida – inicio | El usuario abre la aplicación | N/A | | La pantalla de bienvenida carga correctamente | |  |
| Login | El usuario se logea mediante su cuenta de google | Datos Validos | | El usuario ingresa con éxito. | |  |
|  |  |  | |  | |  |
|  |  |  | |  | |  |
|  |  |  | |  | |  |
| **Criterios de Aprobación del Caso de Prueba:** Deben cumplir con los resultados esperados al menos en un 95% | | | | | | |
| **Decisión de Aprobación del Caso de Prueba:** Aprobó: X Fallo: \_\_\_ | | | | | | |
| **Fecha de Aprobación del Caso de Prueba:**  24/09/2024 | | | | | | |

Prioridades

Casos de Prueba por Características de Prioridad

Se establecieron niveles de prioridad en las actividades, que deben cumplirse de mayor a menor prioridad para cubrir todas las necesidades del cliente en la gestión de casos de prueba.

Esenciales

* Probar la funcionalidad de inicio de sesión para usuarios con diferentes roles (visitante, gestor de pruebas, desarrollador).
* Probar la visualización de proyectos para visitantes, asegurando que solo puedan ver información sin realizar cambios.
* Probar la creación y diseño de casos de prueba por el gestor de pruebas, validando la capacidad de agregar, editar y guardar casos.
* Probar la aprobación y rechazo de casos de prueba por el gestor de pruebas, asegurando que se sigan los flujos de trabajo establecidos.
* Probar la eliminación de casos de prueba por el desarrollador, verificando que se eliminen correctamente y se registre la acción.
* Probar la funcionalidad de navegación entre diferentes proyectos.
* Probar el rendimiento del sistema bajo cargas múltiples, simulando el acceso de varios usuarios.
* Probar la interfaz de usuario para asegurar que sea intuitiva y accesible para todos los roles.
* Probar la visualización de gráficos estadísticos en el dashboard, asegurando que reflejen la salud de los proyectos de manera precisa.
* Probar la respuesta del sistema ante entradas erróneas o inesperadas para garantizar una gestión adecuada de errores.

Flujos de Trabajo de Pruebas

| **Id** | **Nombre de Tarea** | **Comienzo** | **Fin** | **Duración** |
| --- | --- | --- | --- | --- |
| 01 | Planificación de pruebas | - | - | 1 día |
| 02 | Diseño de casos de pruebas | - | - | 2 días |
| 03 | Configuración del entorno de pruebas | - | - | 1 día |
| 04 | Ejecución de pruebas funcionales | - | - | 3 días |
| 05 | Ejecución de pruebas de usabilidad | - | - | 1 día |
| 06 | Ejecución de pruebas de rendimiento | - | - | 2 días |
| 07 | Ejecución de pruebas de seguridad | - | - | 2 días |
| 08 | Ejecución de pruebas de regresión | - | - | 2 días |
| 09 | Ejecución de pruebas de mantenimiento | - | - | 2 días |
| 10 | Evaluación y reporte de resultado | - | - | 2 días |

Entregables

Lista de Entregables de Pruebas

| Entregables | Descripción |
| --- | --- |
| Performance Testify | La presente prueba revisa que cada uno de los elementos que conforman la aplicación Testify funcionen correctamente, se revisan detalladamente uno a uno lo componentes y areas, si es necesario hacer correcciones se reportan al departamento correspondiente para que haga la corrección de los errores aquí encontrados. |

Ficha: escenarios por caso de uso: (a modo de ejemplo)

| Inserción de Nuevo caso de prueba | |
| --- | --- |
| ID Escenario | Flujo Básico |
| 1 | El gestor de pruebas accede al sistema, selecciona la opción “agregar caso de prueba”, llena la información solicitada (titulo, descripción, criterios de aceptación, etc) y almacena esta información en la base de datos. |
| Modificar caso de prueba | |
| ID Escenario | Flujo Básico |
| 2 | El gestor de pruebas accede al sistema, busca el caso de prueba a modificar, selecciona la opción “Modificar”, cambia la información requerida y actualiza el caso de prueba en la base de datos. |
| Eliminar caso de prueba | |
| ID Escenario | Flujo Básico |
| 3 | El gestor de pruebas accede al sistema, busca el caso de prueba a eliminar, selecciona la opción “Eliminar” y confirma la eliminación, para que el caso de prueba se elimine de la base de datos. |

| Acceso de visitante | |
| --- | --- |
| ID Escenario | Flujo Básico |
| 1 | El visitante accede al sistema, navega por los proyectos disponibles y visualiza la información básica de cada uno, sin poder realizar ningún tipo de modificación. |

| Eliminar caso de prueba por error | |
| --- | --- |
| ID Escenario | Flujo Básico |
| 1 | El desarrollador accede al sistema, busca un caso de prueba marcado con errores y selecciona la opción “Eliminar”, confirmando su eliminación. |

Ficha: Resumen de Ciclos de Prueba

ID del Proyecto/ Nombre: Testify

ID del Ciclo de Prueba: Performance Testify

Fechas para el Ciclo de Prueba: Desde: - Hasta: -

| **ID Caso de Uso** | **ID Caso de Pruebas** | **Resultados Esperados** | **Resultados Obtenidos** | **Observación** |
| --- | --- | --- | --- | --- |
| 1 | 1 | 95% | 100% | No se encontraron errores |
| 2 | 1 | 95% | 100% | No se encontraron errores |
| 3 | 1 | 95% | 100% | No se encontraron errores |

**Resultados/Observaciones para el Ciclo de Prueba**: El resultado que se obtuvo estuvo a la altura de los resultados que se esperaban, todo funcionó en base a lo que se tenía planeado en el diseño, cada uno de los botones, cuadros de texto, etiquetas, combos, etc. funcionaron conforme a su respectiva acción que debían realizar.

Aprobado Ciclo de Prueba por:

CLIENTE ORGANIZACIÓN PROBADOR

Criterio para el Inicio y Fin del Plan de Pruebas

Criterios de Inicio

Para que el plan de pruebas se pueda llevar a cabo en la aplicación web Testify para la gestión de casos de prueba, se deben cumplir los siguientes requisitos:

* Contar con los equipos necesarios (servidores, computadoras) para llevar a cabo el plan de pruebas.
* Contar con el software para realizar las pruebas de rendimiento y funcionalidad.
* Contar con las herramientas de soporte necesarias (por ejemplo, para gestión de pruebas y seguimiento de errores).
* Contar con el personal capacitado y entrenado en las metodologías de prueba y uso de herramientas.

Criterios de Fin

Para que el plan de pruebas de la aplicación web Testify de gestión de casos de prueba se dé por concluido, se deben cumplir las siguientes actividades:

* Probar la funcionalidad de agregar nuevos casos de prueba.
* Probar la funcionalidad de editar casos de prueba existentes.
* Probar la funcionalidad de eliminar casos de prueba.
* Probar la funcionalidad de guardar cambios en los casos de prueba.
* Probar la funcionalidad de cancelar cambios sin guardar.
* Probar cada una de las etiquetas (Labels) en la interfaz de usuario.
* Probar cada uno de los cuadros de texto (TextBox) para asegurarse de que aceptan la entrada correcta.
* Probar el DataGridView para la visualización de los casos de prueba.
* Evaluar el rendimiento del sistema bajo condiciones de carga.
* Verificar que la interfaz de usuario sea intuitiva y cumpla con el diseño establecido.

Se realizarán las iteraciones necesarias hasta que la aplicación funcione conforme a lo establecido y se logren los resultados esperados.

Criterios de Suspensión y Retomo de Actividades

Las pruebas podrán ser suspendidas en las siguientes situaciones:

* Enfermedad de alguno de los miembros del equipo de pruebas. La reanudación se llevará a cabo cuando la persona esté en condiciones de continuar o sea reemplazada por otra persona capacitada.
* Problemas técnicos graves que impidan la continuación de las pruebas, los cuales deberán ser resueltos antes de reanudar.

Criterios para el Lanzamiento

Para que el plan de pruebas de la aplicación de gestión de casos de prueba Testify se considere concluido, se deben cumplir las siguientes condiciones:

* El botón de agregar debe funcionar correctamente y permitir la adición de nuevos casos de prueba.
* El botón de editar debe poder modificar el registro seleccionado adecuadamente.
* El botón de eliminar debe permitir la eliminación de registros sin errores.
* El botón de guardar debe cumplir su función y almacenar cambios de manera efectiva.
* El botón de cancelar debe deshacer cualquier cambio no guardado.
* Los cuadros de texto deben enviar la información correcta a la base de datos correspondiente.
* Las etiquetas deben coincidir con los cuadros de texto y deben estar correctamente escritas.
* La interfaz debe estar organizada conforme al diseño definido y ser accesible para los usuarios.
* El rendimiento de la aplicación debe ser óptimo, asegurando que no haya bloqueos ni lentitud durante la ejecución.

Clasificación de los errores

| Calificación | Definición de gravedad | Definición de prioridad |
| --- | --- | --- |
| 1 *(Alta)* | El error provoca inserción de datos incorrectos, o causa fallos completos del sistema, impidiendo el uso de funcionalidades clave. | El error debe corregirse lo antes posible. El error bloquea el progreso y afecta la operatividad del sistema, interrumpiendo las pruebas y entregas del producto. |
| 2 *(Media)* | El error causa problemas significativos en la funcionalidad o en aspectos importantes del sistema, como interacción con usuario o integración con módulos. | El error debe corregirse antes del lanzamiento del producto. Aunque no bloquea completamente su impacto debe resolverse para garantizar la entrega exitosa. |
| 3 *(Baja)* | El error es menor y no afecta la funcionalidad central del sistema, pero genera inconvenientes estéticos o errores menores en la usabilidad. | El error puede corregirse en futuras iteraciones o versiones del producto, sin bloquear el lanzamiento inicial. |

Resultados de la prueba

Tras la ejecución de todos los casos de prueba planificados, se obtuvieron resultados positivos en la mayoría de ellos. No se encontraron errores críticos o bloqueantes de **Gravedad 1**, lo que indica que el sistema está listo para ser lanzado. Los errores menores identificados de **Gravedad 2 o 3** serán documentados para su corrección en próximas versiones, sin afectar el lanzamiento inicial del sistema.

Reportes del problema, escalada y resolución

Durante el proceso de pruebas, cualquier problema detectado será documentado detalladamente en un reporte de errores, incluyendo la descripción del fallo, la severidad y la prioridad de corrección. En caso de que un problema no pueda ser resuelto inmediatamente, se deberá seguir el proceso aquí detallado por los siguientes pasos:

1. **Identificación del problema**: Documentar el problema de manera clara y precisa.
2. **Asignación del problema**: El equipo de desarrollo correspondiente será informado para corregir el fallo según la prioridad.
3. **Resolución**: Una vez corregido, se validará la solución mediante pruebas de regresión para asegurarse de que el problema ha sido solucionado y no se han generado nuevos errores.

Este ciclo se repetirá hasta que todos los problemas críticos se hayan solucionado y se verifique que el sistema cumple con los requisitos para su lanzamiento.

Riesgos

| Riesgos | Plan de Contingencia | Impacto |
| --- | --- | --- |
| Dado que el equipo encargado de las pruebas no esté debidamente capacitado en las herramientas o procesos, podría ocurrir una deficiente ejecución de pruebas, lo que conducirá a resultados incorrectos y retrasos en la detección de errores críticos. | Capacitar al personal. | Todas las áreas pueden verse afectadas por la incorrecta identificación de errores o capacitación lo que puede comprometer la calidad del sistema. |
| Dado que el tiempo estimado para las pruebas sea insuficiente o mal calculado, podría ocurrir una extensión significativa del tiempo de pruebas, lo que conducirá a retrasos en el lanzamiento del proyecto. | Revisar y mejorar el plan de pruebas, capacitar mejor al personal y asignar más recursos a las pruebas. | El área de gestión del proyecto se verá afectada, especialmente en cuanto a expectativas y fechas de entregas con el cliente. |
| Dado que los errores en la ejecución de pruebas no sean detectados a tiempo, podría ocurrir una acumulación de defectos que no se corrijan antes del lanzamiento, lo que conducirá a una disminución de la calidad del producto final y posibles insatisfacciones del cliente. | Volver a ejecutar las pruebas con un control más exhaustivo y establecer protocolos de revisión cruzada. | Las áreas de pruebas y desarrollo se verán directamente afectadas, comprometiendo los tiempos y la calidad del sistema afectando a la entrega final. |

Reportes de Problemas y Resolución

Los problemas identificados durante la ejecución de las pruebas serán documentados de manera detallada, asignado un nivel de prioridad y gravedad. Para cada problema reportado se seguirá el siguiente proceso de resolución:

Identificación y Reporte:

1. Cada problema será registrado, con detalles del entorno, pasos para replicarlo, resultados esperados y obtenidos, y cualquier otra información relevante.
2. Los roles asignados (desarrollador, gestor de pruebas) recibirán notificaciones automáticas sobre los problemas reportados.

Asignación y Priorización:

1. El gestor de pruebas revisará cada problema y le asignará una prioridad en función de su impacto y gravedad.
2. El desarrollador será el encargado de analizar los problemas y determinar la causa raíz, con un plazo de resolución adecuado a la prioridad asignada.

Resolución y Verificación:

1. El desarrollador corregirá el problema y documentará la solución aplicada.
2. El gestor de pruebas realizará pruebas de regresión para verificar que la solución resuelve el problema sin introducir nuevos errores.

Cierre del Problema:

1. Una vez verificado, el problema será marcado como resuelto y se actualizará el registro de pruebas para reflejar los resultados de la corrección.
2. En caso de que la corrección no sea efectiva, el problema se reabrirá y el proceso se repetirá.

Este proceso se aplicará a todos los problemas identificados, desde errores funcionales hasta problemas de usabilidad o rendimiento. La transparencia en la documentación y la trazabilidad de los problemas permitirán un seguimiento efectivo del estado de cada uno y garantizarán la calidad final del producto.

Responsabilidades, Personal y Necesidades de Capacitación

Personal y Roles Necesarios

| **Roles** | **Recursos Necesarios** | **Estado** | **Responsabilidades Específicas o Comentarios** |
| --- | --- | --- | --- |
| Administrador de plan de pruebas | 1-2 | Asignado (Valeria Ojeda) | Diseñar y aprobar los casos de pruebas debe además coordinar la ejecución del plan de pruebas, garantizando que todos los casos de pruebas se desarrollen y gestionen de acuerdo a los objetivos del proyecto. Planificar los ciclos de pruebas con fecha y responsable y ejecutar y revisar los resultados de los casos de pruebas. |
| Gestor de pruebas. | 1-2 | Asignado (Oyarzo Malena) | Diseñar y aprobar los casos de pruebas, ejecutar y revisar los resultados aprobando o rechazando según los criterios definidos, priorizar las pruebas. |
| Desarrollador | 1 | Asignado (Sly Eduardo) | Corregir los errores detectados, verificar que las soluciones implementadas funcionen correctamente mediante pruebas de regresión y asegurarse de que los errores sean resueltos sin afectar otras funcionalidades del sistema. |
| Asegurador de la calidad | 1 | Asignado (Levipichun Emilio) | Verificar que el software se adhiere a los estándares de calidad establecidos en el plan SQA, asegurar que las pruebas cubran adecuadamente todas las áreas criticas del sistema (seguridad, usabilidad, rendimiento). |

Al momento de la confección del presente plan de pruebas el equipo de desarrollo OSLO se encuentra en la fase de Elaboración iteración 1, y actualmente las pruebas de funcionalidad del sistema cubren un 0%, se espera que a medida que el desarrollo avance en las siguientes iteraciones se cubran parte de las pruebas como por ejemplo: pruebas de prototipos o pruebas de fases temprana de desarrollo, aumentando gradualmente hasta cumplir los requisitos establecidos por el cliente.