# **Guía para Realizar Pruebas Funcionales**

# **Procedimientos de prueba**

Cada prueba será ejecutada en un entorno de pruebas con datos de ejemplo. Los resultados serán documentados y cualquier fallo será registrado en el reporte de incidencias.

# **Criterios de éxito y aceptación**

Una prueba se considera exitosa si:

- Los resultados observados coinciden con los esperados.  
- No se detectan errores funcionales.  
- Los datos se almacenan correctamente en la base de datos.  
- La interfaz responde adecuadamente a la interacción del usuario.

# **Escala de cumplimiento de los criterios BICEPS e CORRECT**

Por otro lado, las pruebas deben medirse para cumplir con los criterios de los métodos BICEPS y CORRECT que ayudarán a identificar casi todos los errores posibles dentro de cada historia de usuario. En la siguiente tabla se muestra a detalle lo que se debe tomar en cuenta al establecer las pruebas para cada historia de usuario.

**Boundaries (Límites)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Identificación de valores mínimos y máximos permitidos** | Definir los rangos de valores permitidos para cada campo de entrada. |  |
| **Prueba con valores límite inferiores** | Ingresar datos en los valores más bajos permitidos (ejemplo: contraseña de 8 caracteres si es el mínimo permitido). |  |
| **Prueba con valores límite superiores** | Ingresar datos en los valores más altos permitidos (ejemplo: nombre con 50 caracteres si es el máximo). |  |
| **Ingreso de datos fuera del límite** | Evaluar cómo responde el sistema cuando se ingresa un valor menor o mayor al permitido. |  |
| **Manejo de datos nulos o vacíos** | Verificar si el sistema permite guardar datos en blanco en campos obligatorios. |  |

**Inputs (Entradas)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de formatos de entrada** | Asegurar que los datos ingresados cumplan con las reglas de formato (ejemplo: correos electrónicos deben tener '@' y dominio válido). |  |
| **Prueba con caracteres especiales** | Ingresar datos con caracteres como @#$%&\* para evaluar su comportamiento en los campos de entrada. |  |
| **Prueba con datos no esperados** | Introducir tipos de datos incorrectos (ejemplo: letras en un campo numérico). |  |
| **Simulación de múltiples entradas simultáneas** | Evaluar cómo responde el sistema cuando varios usuarios ingresan datos al mismo tiempo. |  |
| **Validación de campos obligatorios** | Verificar que no se permita dejar campos esenciales vacíos. |  |

**Correctness (Corrección)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Comparación de resultados esperados con reales** | Ingresar datos conocidos y verificar si el sistema genera la salida esperada. |  |
| **Verificación de cálculos automáticos** | Si hay campos con cálculos automáticos, confirmar que los valores generados son correctos. |  |
| **Comprobación de almacenamiento correcto** | Asegurar que los datos ingresados se almacenen sin modificaciones incorrectas en la base de datos. |  |
| **Verificación de cambios en tiempo real** | Evaluar si las modificaciones realizadas en la interfaz se reflejan correctamente. |  |

**Errors (Errores)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Prueba de mensajes de error claros** | Asegurar que los mensajes de error sean comprensibles y ayuden al usuario a corregir su entrada. |  |
| **Manejo de errores en el sistema** | Simular errores comunes (ejemplo: fallas de conexión) y evaluar la respuesta del sistema. |  |
| **Evaluación de recuperación ante errores** | Verificar si el sistema permite continuar correctamente después de un error. |  |
| **Intento de ingresar datos incorrectos repetidamente** | Evaluar si el sistema maneja adecuadamente múltiples intentos de ingreso erróneo. |  |

**Performance (Rendimiento)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Medición de tiempos de carga de la interfaz** | Evaluar cuánto tarda la plataforma en cargar páginas y elementos. |  |
| **Prueba de respuesta con múltiples usuarios** | Simular el acceso simultáneo de varios usuarios para medir el rendimiento. |  |
| **Evaluación del uso de memoria y CPU** | Medir el consumo de recursos del sistema cuando se realizan múltiples operaciones. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Evaluación con CORRECT**

**Conformance (Conformidad)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de cumplimiento de normativas** | Revisar si el sistema cumple con regulaciones de seguridad y privacidad. |  |
| **Evaluación de conformidad con estándares de accesibilidad** | Verificar si la plataforma sigue principios de accesibilidad para personas con discapacidad. |  |
| **Revisión del cumplimiento con la arquitectura establecida** | Confirmar que el desarrollo respeta las reglas definidas en la arquitectura del software. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Conformance (Conformidad)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de cumplimiento de normativas** | Revisar si el sistema cumple con regulaciones de seguridad y privacidad. |  |
| **Evaluación de conformidad con estándares de accesibilidad** | Verificar si la plataforma sigue principios de accesibilidad para personas con discapacidad. |  |
| **Revisión del cumplimiento con la arquitectura establecida** | Confirmar que el desarrollo respeta las reglas definidas en la arquitectura del software. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Ordering (Orden)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de flujo de navegación** | Asegurar que el usuario pueda realizar tareas sin interrupciones innecesarias. |  |
| **Prueba de secuencia de acciones** | Evaluar si las tareas dentro del sistema siguen un orden lógico. |  |
| **Simulación de interrupciones en el flujo** | Probar qué sucede si el usuario abandona una acción a la mitad y regresa después. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Range (Rangos)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Definición de valores aceptables** | Especificar los valores permitidos en cada campo de entrada. |  |
| **Pruebas con valores en el límite del rango** | Evaluar qué sucede cuando los datos ingresados están justo en los extremos del rango permitido. |  |
| **Prueba con valores fuera del rango** | Verificar la respuesta del sistema al recibir datos más grandes o más pequeños de lo permitido. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Reference (Referencia)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de relaciones entre datos** | Revisar si la información referenciada en diferentes módulos es coherente. |  |
| **Comprobación de referencias cruzadas en la base de datos** | Asegurar que las relaciones de datos entre tablas sean correctas. |  |
| **Evaluación de consistencia en las referencias** | Si un usuario cambia de rol, verificar que sus permisos se actualicen adecuadamente. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Existence (Existencia)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Verificación de existencia de campos obligatorios** | Revisar que los campos esenciales no puedan omitirse. |  |
| **Pruebas con combinaciones de datos incompletos** | Simular la omisión de datos esenciales y evaluar la respuesta del sistema. |  |
| **Prueba con eliminación de registros referenciados** | Verificar qué sucede si se elimina un registro vinculado a otros datos. |  |
| **Simulación de estrés con grandes volúmenes de datos** | Probar cómo responde el sistema cuando se ingresan grandes cantidades de datos. |  |

**Cardinality (Cardinalidad)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Validación de restricciones en registros múltiples** | Revisar que no se permitan duplicados en campos que deben ser únicos. |  |
| **Pruebas de límites de cardinalidad** | Evaluar si el sistema permite la cantidad correcta de relaciones (ejemplo: un usuario puede tener varias postulaciones, pero un proyecto no debe tener el mismo usuario postulado varias veces). |  |

**Time (Tiempo)**

| **Actividad** | **Descripción** | **Pruebas necesarias** |
| --- | --- | --- |
| **Medición de tiempos de ejecución de procesos clave** | Evaluar si los tiempos de respuesta del sistema cumplen con los requisitos esperados. |  |
| **Simulación de retrasos en la red** | Probar la estabilidad del sistema cuando hay problemas de conexión. |  |
| **Evaluación de rendimiento en consultas a la base de datos** | Medir cuánto tarda el sistema en recuperar información almacenada. |  |

**RIGHT BICEP**

La información de la siguiente tabla es una guía para registrar el cumplimiento de los criterios BICEP los cuales son los acrónimos de todos los tipos de pruebas funcionales. Para llenar los ejemplos de dicha tabla se utilizó esta historia de usuario: *"Como usuario, quiero recibir notificaciones sobre los proyectos con cupos limitados para postularme antes de que se llenen”.*

|  | **Definición** | **Ejemplo** | **Pruebas necesarias** |
| --- | --- | --- | --- |
| **Right** | Los resultados de la prueba deben ser precisos y cumplir con los requisitos esperados. | Asegurar que el sistema envíe notificaciones correctas y oportunas sobre proyectos con cupos limitados, alertando a los usuarios antes de que se llenen los espacios disponibles. |  |
| **Boundary conditions** | Se deben evaluar condiciones extremas para garantizar que el sistema funcione correctamente en escenarios poco comunes. | Probar que el sistema funcione correctamente en situaciones extremas, como cuando solo queda un cupo o varios usuarios intentan postularse al mismo tiempo, garantizaría el envío adecuado de las notificaciones. |  |
| **Inverse relationships** | Se deben probar relaciones inversas para validar la coherencia y precisión del sistema. | Probar que, si un usuario recibe una notificación sobre un cupo limitado, el sistema no envíe alertas una vez que el proyecto esté lleno, asegurando coherencia y precisión. |  |
| **Cross-checking results** | Los resultados deben contrastarse con otros métodos de verificación para asegurar su validez. | Comparar las notificaciones automáticas con el registro manual de los proyectos que están por llenarse, aseguraría que el sistema envíe alertas correctamente sobre los proyectos con cupos limitados. |  |
| **Error conditions** | Se debe probar cómo el sistema maneja fallos y situaciones inesperadas. | Probar cómo el sistema maneja fallos al momento en que el usuario intenta postularse a un proyecto lleno, aseguraría que el sistema muestre un mensaje claro del error y sugiriendo otras opciones disponibles. Garantizando una experiencia fluida incluso en situaciones inesperadas. |  |
| **Performance characteristics** | Es necesario verificar que el sistema tenga un desempeño adecuado bajo diferentes condiciones. | Probar que el sistema sea capaz de enviar notificaciones rápidamente, incluso cuando muchos usuarios intentan postularse al mismo tiempo, asegurando que todos reciban la información a tiempo. |  |

# **Cumplimiento de las historias de usuario con los criterios INVEST**

Además para cada historias de usuario tiene que haber una revisión para que se garantice el cumplimiento del criterio INVEST para que cada historia sea una de calidad. Para llenar los ejemplos de esta tabla se utilizó esta historia de usuario: *"Como usuario, quiero recibir notificaciones sobre los proyectos con cupos limitados para postularme antes de que se llenen”.*

|  | **Definición** | **Ejemplo** | **Escala de cumplimiento** |
| --- | --- | --- | --- |
| **Independent** | La historia debe ser accionable y completable por sí sola. No debe depender inherentemente de otra. | Garantizar que la funcionalidad de notificaciones sea desarrollada y entregada sin requerir la finalización de otros módulos, como la creación de proyectos. |  |
| **Negotiable** | Hasta que se esté trabajando en ella, debe poder reescribirse. Debe permitir cambios de manera flexible. | Permitir ajustar la forma de entrega de las notificaciones para incluir diferentes tipos según las necesidades del proyecto durante el desarrollo. |  |
| **Valuable** | Debe aportar valor real a un cliente, usuario o parte interesada. | Asegurar que las notificaciones de proyectos con cupos limitados brinden valor al usuario, permitiéndole postularse antes de que se llenen. |  |
| **Estimable** | Debe ser medible en tamaño, permitiendo estimar el esfuerzo necesario para completarlo. | Asegurar que el sistema de notificaciones sea autónomo y detecte proyectos con cupos limitados y alerte a los usuarios, sin depender de otras funcionalidades como la gestión de postulaciones. |  |
| **Small** | La historia debe ser lo suficientemente pequeña para estimarla y planificarla con facilidad. Si es demasiado grande, se debe reescribir o dividir en historias más pequeñas. | Separar la detección de proyectos con cupos limitados del envío de notificaciones para que cada tarea sea más manejable y estimable, así como se pueda planificar mejor su desarrollo. |  |

# **Conclusión**