Verificación y Validación de Software en un Sistema Complejo

Integrantes: Hans Huiniguir, Felipe Azocar, Sebastián Vásquez.

Modulo: Taller de Testing y Calidad de Software.

Fecha: 10/09/25

Profesor: Diego Obando Aguilera.

Contenido

Introducción

En nuestro informe buscaremos aplicar técnicas de pruebas vistas durante las clases de “Taller de Testing y Calidad de Software” incluyendo pruebas funcionales y no funcionales sobre un sistema complejo, utilizando un plan de pruebas alineado con los estándares de calidad (ISO/IEC 25010 e IEEE 829).

El sistema que hemos elegido es la aplicación digital de BancoEstado, es sistema financiero que permite a los usuarios realizar múltiples operaciones bancarias desde su plataforma web y móvil. Esta aplicación incluye las siguientes funcionalidades:

- Consulta de saldos y movimientos realizados.
- Transferencias entre cuentas propias y de terceros.
- Pagos de servicios (Cuentas de la luz, agua, compras en comercios).
- Gestión de productos financieros (cuentas, tarjetas, créditos).
- Acceso mediante autentificación en dos pasos y biometría (en la aplicación móvil).
- Módulos de atención al cliente vía chatbot o formulario.
- Funcionalidades adicionales como bloqueo de tarjetas, solicitud de productos, etc.

Este sistema está disponible para millones de usuarios de Chile ya que cuenta tanto con una aplicación móvil y una plataforma web.

Las razones de la elección de este sistema se deben a:

1. La complejidad del Sistema: Maneja múltiples capas tecnológicas (transferencias y pagos) y debe de operar en condiciones de alta demanda y disponibilidad.

2. Relevancia Nacional: Al ser un sistema al que todos los chilenos tienen acceso, la fiabilidad, accesibilidad y seguridad de su plataforma son de alto impacto social.

3. Importancia de la calidad del Software: Los errores en sistemas bancarios pueden significar pérdidas financieras y de clientes, por lo que es importante

realizar pruebas funcionales y no funcionales bajo estándares internacionales para asegurar la calidad del producto.

Objetivos del Plan de Pruebas

Con este plan de pruebas de busca garantizar la calidad, confiabilidad funcionamiento correcto del sistema digital BancoEstado, evaluando sus funcionalidades clave como sus atributos de calidad definidos por el estándar ISO/25010.

Objetivos Específicos:

1. Verificar que el sistema cumpla con los requerimientos funcionales definidos, como transferencias, consultas de saldo y la autentificación segura.

2. Validar cumpla con los requerimientos del usuario final, asegurando la usabilidad, accesibilidad y experiencia fluida.

3. Aplicar pruebas funcionales y no funcionales para identificar defectos, riesgos y oportunidades de mejorar el sistema.

4. Evaluar el sistema bajo condiciones normales y límites de uso, incluyendo estrés, concurrencia y escenarios de fallo.

5. Asegurar la conformidad del sistema con estándares de calidad del software(ISO/25010), como en los atributos de confiabilidad, desempeño, seguridad y mantenibilidad.

6. Generar evidencia documentada del proceso de pruebas, incluyendo casos, resultados de pruebas y trazabilidad.

Alcance del Plan de Pruebas

El plan de pruebas abarca los siguientes aspectos del sistema BancoEstado

- **Plataformas Cubiertas:** Sitio Web bancario aplicación móvil
- **Módulos Funcionales Incluidos:**
  - ✓ Autenticación (Login y validación en dos pasos)

- ✓ Consulta de saldos y movimientos
- ✓ Transferencias (propias o a terceros) y pago de servicios
- ✓ Gestión de productos (cuentas, tarjetas, créditos)
- ✓ Solicitudes Automatizadas (recuperación de claves, bloqueo de tarjetas).

- **Pruebas a aplicar:**
  - ✓ Pruebas Funcionales: Validación de flujos
  - ✓ Pruebas no Funcionales: Pruebas de rendimiento, usabilidad, seguridad y accesibilidad.
- Entorno de pruebas: Simulación de entorno real, uso de datos ficticios, pruebas controladas.

Limitaciones del Plan de Pruebas

El plan presenta las siguientes restricciones:

- **Acceso restringido al entorno productivo:** por políticas de seguridad de BancoEstado no es posible ejecutar pruebas en un entorno real, lo que limita la validación de comportamientos bajo condiciones idénticas a producción.
- **No es posible utilizar datos reales de Clientes:** Para resguardar la confidencialidad de los datos, las pruebas solo se realizan con datos sintéticos o anonimizados.

Plan de Pruebas según Estándares

Identificación del documento

- Nombre del plan: Plan de Pruebas de Verificación y Validación del Sistema Bancario BancoEstado.
- Versión :1.0
- Fecha: 12 de septiembre de 2025
- Responsable: Equipo de Aseguramiento de Calidad (QA) – BancoEstado TI.
- Aprobación: Gerencia de Desarrollo de Sistema.

Características a probar (ISO/IEC 25010)

Las pruebas cubrirán las principales características de la calidad definidas por la norma:

Característica	Subcaracterísticas	Objetivo de prueba
Funcionalidad	Adecuación, exactitud, seguridad.	Validar que las operaciones bancarias (transferencias, pagos, créditos) funciones según requisitos y que las API se integren con otros bancos de forma segura.
Fiabilidad	Madurez, disponibilidad, tolerancia a los fallos.	Garantizar continuidad 24/7, con conmutación por errores ante fallas de servidores o redes.

Usabilidad	Comprensibilidad, accesibilidad, diseño atractivo.	Confirmar que las aplicaciones web y móviles sean intuitivas y cumplan cormas de accesibilidad (WCAG).
Rendimiento/Eficiencia	Comportamiento / Uso de recursos	Asegurar tiempos de respuesta en transacciones con una carga máxima prevista.
Mantenibilidad	Modularidad, modificabilidad.	Evaluar que el código y la arquitectura permitan cambios y correcciones sin impacto mayor.
Portabilidad	Adaptabilidad, instabilidad	Verificar compatibilidad en sistemas operativos Windows, Linux y dispositivos iOS/Android.

Características que no se probaran

- Hardware propietario de cajeros automáticos, por depender de fabricantes externos.
- Integraciones con sistemas de terceros en fase beta (servicios aun no disponibles en ambiente de pruebas.
- Aspectos de marketing o diseño grafico que no afecten la funcionalidad.

Críticos de paso/fallo

- **Paso:** 100% de los casos críticos deben ser exitosos. Los casos no críticos requieren al menos 95% de éxito.
- **Fallo:** Cualquier vulnerabilidad de seguridad alta o defecto que impida operaciones bancarias básicas.
- Se registrará cada defecto en Jira, con sus prioridades y planes de corrección.

Estrategia de Pruebas

Niveles de prueba

1. Pruebas Unitarias:
    - Responsable: Desarrolladores.
    - Herramientas: xUnit, NUnit.
    - Objetivo: Validar funciones y métodos individuales, como cálculo de intereses o autenticación.
2. Pruebas de integración:
    - Verifican interacción entre módulos (Ejemplo, Autenticación-Motor de transacciones).
    - Herramientas: Postman, SoapUI para las APIs.
3. Pruebas de sistema:
    - Pruebas end-to-end en entornos de pruebas que replican la infraestructura de producción.
    - Incluyen pruebas de carga, seguridad y usabilidad.

Enfoques de prueba

- **Caja Negra:** Validación de funcionalidades desde la perspectiva del usuario sin conocer el código.
- **Caja blanca:** Revisiones de código, cobertura de sentencias y ramas para módulos críticos.
- **Caja gris**: Combinación de ambas para pruebas de integración y seguridad, especialmente en APIs y servicios internos.

Tipos de ejecución

- Pruebas manuales: Casos de aceptación de usuario (UAT), pruebas exploratorias y de usabilidad.
- Pruebas Automatizadas:
    - Integración continua en Jenkin/GitLab CI.
    - Appium para móviles.
    - JMerter para pruebas de rendimiento y estrés.
    - Selenium y Cypress para UI web.

Recursos y Cronograma

Herramientas necesarias

- **Gestión de pruebas y defectos:** Jira, TestRail.
- **Automatización:** Selenium, Cypress, Appium.
- **Rendimiento:** Apache JMerter.
- **Análisis estático:** SonarQube, OWASP ZAP para seguridad.

Personal requerido

Rol	Cantidad	Responsabilidades
Líder de QA	1	Coordinar y supervisar todo el plan de pruebas.
Ingenieros de QA	4	Diseñar, ejecutar y automatizar casos de prueba.
Desarrolladores	6	Corregir defectos y apoyar en pruebas unitarias.
Analista de Seguridad	1	Ejecutar pruebas de penetración y revisar vulnerabilidades.
Usuarios de Negocio	3	Pruebas de aceptación y validación funcional

Estimación de Tiempos

Actividad	**Duración**
Preparación del entorno y herramientas	2 semanas
Diseño de casos de prueba	3 semanas
Pruebas unitarias e integración	4 semanas
Pruebas de sistema (rendimiento, seguridad, usabilidad).	5 semanas
UAT y cierre	2 semanas

Implementación de Tipos de Pruebas

Es cuando se llevan a cabo los diferentes enfoques de testing definidos previamente compatibilidad, regresión, integración. La utilidad es asegurar que cada aspecto crítico del sistema sea verificado en escenarios reales y que los criterios de calidad se cumplan.

¿Que son las pruebas de compatibilidad?

Son Test o Pruebas que tienen como objetivo verificar que la aplicación funcione correctamente en distintas combinaciones de hardware, sistemas operativos, navegadores y redes. Su objetivo es asegurarse de que la interfaz y las funcionalidades clave en nuestro caso Banco Estado (login, transferencias, pagos) se comporten de forma idéntica en Android, iOS y en navegadores móviles, sin errores visuales ni de ejecución.

¿Que son las pruebas de regresión?

Las pruebas de regresión consisten en volver a ejecutar un conjunto de casos de prueba tanto funcionales como no funcionales cada vez que se introduce un cambio en el código. Su fin es detectar de manera temprana si nuevas funcionalidades o correcciones han provocado fallos en áreas previamente validadas, manteniendo la estabilidad de la app tras cada actualización.

¿Qué son pruebas de integración?

Las pruebas de integración validan la interacción y el flujo de datos entre módulos o componentes independientes (por ejemplo, API de autenticación, capa de negocio de transferencias y la interfaz de usuario). Se ejecutan una vez que las unidades básicas funcionan de forma aislada, para garantizar que, al combinarse, los servicios y componentes colaboran correctamente en escenarios reales.

Pruebas de Compatibilidad Caso App Banco Estado.

Diseño de casos:

- Matriz de compatibilidad priorizada combinaciones de mayor uso según análisis de datos de uso real de la aplicación móvil (analytics) como pueden ser:
- Redes más utilizadas (Wi-Fi, 4G, 5G)
- Android 12/13/14 en modelos populares; iOS 15/16/17 en iPhone recientes y antiguos.
- Casos: inicio de sesión, transferencia pequeña, pago QR, bloqueo de tarjeta, recuperación de clave en cada combinación.

Criterios de compatibilidad:

- Funcionalidad crítica operativa en al menos 80% de combinaciones prioritarias.
- UI (Interfaz de Usuario) no degradada y accesible; fallas específicas por dispositivo deben documentarse con screenshots y logs (registro).

Pruebas de Regresión Caso App Banco Estado.

Estrategia:

- Definir un conjunto organizado de casos de prueba para detectar si algo que antes funcionaba se rompió por error (suite de regresión) como puede ser smoke + 30–50 casos E2E críticos que debe correr en cada merge a rama release.
- Automatizar los casos repetitivos como lo son: login, listar cuentas, transferencia con y sin autentificación.
- Ejecutar suite completa de regresión en Integración Continua o CI nocturno y antes de release candidate (versión del software que está casi lista para ser lanzada al público pero aún se somete a pruebas finales para detectar errores críticos) .

Selección de casos críticos:

- Login/autenticación multifactor.
- Transferencia interna y a tercero.
- Pago con QR y confirmación.
- Bloqueo/desbloqueo tarjeta.
- Historial y conciliación de movimientos.

Pruebas de Integración Caso App Banco Estado.

Tipos de enfoques de Integración :

- **Top-down: se integra desde la interfaz hacia los módulos internos.**
- **Bottom-up: se integra desde los módulos base hacia la interfaz.**
- **Big bang: se integran todos los módulos de una sola vez.**

Metodo que hemos elegido: *Top-down* el cual es un método de integración que empieza por los componentes de alto nivel en nuestro caso la interfaz y conecta hacia abajo los componentes internos.

¿Por qué?

Para priorizar flujos de negocio: comenzar por la UI + API gateway (puerta de entrada única) simulando servicios externos, luego integrar pasarela de pagos y servicios de terceros en staging (entorno de prueba que simula producción para validar la app antes del lanzamiento).

Casos de prueba específicos:

- Integración login : — session o sesión — notificaciones push(mensajes que la app envía al dispositivo aunque esté cerrada)
- Transferencia: app -> API -> pasarela de pagos -> callback -> notificación de éxito
- Pago QR: generación del QR, lectura, autorización y confirmación en backend
- Manejo de fallas : cuando una operación tarda demasiado en completarse y se interrumpe automáticamente (timeouts) ejemplo : si la pasarela de pagos no responde en 10 segundos, se cancela la solicitud.
- reintentos automáticos (retries) ejemplo : si una transferencia falla por timeout, la app puede intentar de nuevo hasta 3 veces antes de mostrar error.
- propiedad de una operación que produce el mismo resultado aunque se ejecute varias veces (idempotencia) ejemplo : si el usuario presiona “pagar” dos veces por error, solo se realiza una sola transferencia, no dos.

Tipos de Pruebas Funcionales

1. Pruebas de Humo (Smoke Testing)

Qué son:

- Se ejecutan después de una nueva versión o despliegue.
- Su objetivo es verificar que las funciones **más críticas** del sistema funcionan y que "no se quemó el software" (de ahí el nombre "smoke").
- Si fallan, no se sigue probando porque la build es inestable.

Ejemplo en BancoEstado:

- Abrir la app → probar login básico.
- Consultar saldo → debe cargar correctamente.
- Intentar una transferencia pequeña → debe completarse.

Criterio de aceptación:

- Todas las funciones críticas deben ejecutarse sin errores.
- Si alguna falla (ej. no se puede logear), la build se rechaza y se devuelve a desarrollo.

2. Pruebas de Aceptación (User Acceptance Testing – UAT)

Qué son:

- Se hacen con usuarios finales o representantes del negocio.
- Validan que el sistema cumple lo que el cliente pidió en los requisitos.
- Son las últimas antes de liberar el sistema a producción.

Ejemplo en BancoEstado:

- Un usuario simula transferir a un tercero con clave dinámica → la app confirma el pago y genera comprobante.
- Un usuario paga una cuenta de luz desde la app → recibe confirmación en pantalla y correo.

Criterio de aceptación:

- El flujo completo debe ser exitoso desde la perspectiva del cliente.
- Debe cumplir los requisitos de negocio (ej. confirmación visible y comprobante enviado).

4. **Pruebas de Regresión**

Qué son:

- Se repiten pruebas ya ejecutadas para confirmar que **nuevos cambios no rompieron lo que antes funcionaba**.
- Son esenciales en aplicaciones bancarias con actualizaciones frecuentes.

Ejemplo en BancoEstado:

- Tras actualizar el módulo de transferencias internacionales, volver a probar login, consulta de saldo y transferencias locales.
- Validar que bloquear tarjeta sigue funcionando correctamente.

Criterio de aceptación:

- Todos los casos previamente validados deben seguir funcionando igual.
- Ningún cambio debe introducir errores en funciones estables.

Pruebas No Funcionales

1. Pruebas de Rendimiento

- **Objetivo:** Verificar que la aplicación soporte la carga de múltiples usuarios y que las operaciones críticas respondan en tiempos aceptables.
- **Ejemplo aplicado a BancoEstado:**
  - Consulta de saldo.
  - Transferencias a terceros en horarios de alta demanda.
- **Herramienta propuesta: Apache JMeter** (simula miles de usuarios concurrentes y mide tiempos de respuesta).
- **Métrica:**

    - Tiempo promedio de respuesta.
    - Cantidad de transacciones procesadas por segundo.
- **Criterio de aceptación:**
    - El 95% de las operaciones críticas deben responder en < 3 segundos.

2. Pruebas de Usabilidad / Accesibilidad

- **Objetivo:** Validar que la aplicación sea fácil de usar e inclusiva, cumpliendo estándares internacionales de accesibilidad.
- **Ejemplo aplicado a BancoEstado:**
    - Contraste adecuado en botones y menús.
    - Compatibilidad con lectores de pantalla para personas no videntes.
    - Navegación posible solo con teclado.
- **Herramientas propuestas:**
    - **Estándar WCAG 2.1 (nivel AA)** como referencia.
    - **Lighthouse** o **WAVE** para análisis automatizado.
- **Métrica:**
    - Puntuación de accesibilidad en la herramienta (0 a 100).
    - Porcentaje de cumplimiento de pautas WCAG 2.1.
- **Criterio de aceptación:**
    - Cumplimiento de al menos nivel AA de WCAG 2.1.
    - Puntuación mínima de 90/100 en accesibilidad (según Lighthouse).

3. Pruebas de Seguridad

- **Objetivo:** Garantizar que el sistema proteja los datos sensibles y no presente vulnerabilidades comunes.
- **Ejemplo aplicado a BancoEstado:**
    - Validar que el login sea resistente a ataques de fuerza bruta.
    - Revisar seguridad de las APIs de transferencias.

    - Confirmar que las sesiones expiren tras inactividad.
- **Herramienta propuesta: OWASP ZAP** (ejecuta pruebas de penetración automatizadas).
- **Métrica:**
    - Número y criticidad de vulnerabilidades encontradas (según **OWASP Top 10**).
- **Criterio de aceptación:**
    - No se deben detectar vulnerabilidades **críticas ni altas**.
    - Vulnerabilidades medias deben ser corregidas antes de salir a producción.

Análisis y Conclusiones

La implementación del plan de **Verificación y Validación (V&V)** en la aplicación de BancoEstado permitió identificar los principales retos y beneficios de asegurar la calidad del software, además de establecer recomendaciones prácticas para mantener y mejorar el sistema.

Desafíos identificados

- **Coordinación de equipos:** V&V requiere que desarrollo y QA trabajen juntos desde el inicio para encontrar errores tempranos.
- **Definición de requerimientos:** Si los requerimientos no están claros, los casos de prueba pueden quedar incompletos.
- **Pruebas no funcionales:** Medir rendimiento, accesibilidad y seguridad necesita herramientas especiales y más tiempo de planificación.

Beneficios esperados

- **Confiabilidad:** Mayor estabilidad en operaciones críticas, como login y transferencias.
- **Cumplimiento de normas:** Uso de estándares internacionales que elevan la calidad del sistema.

- **Experiencia de usuario:** Interfaces más accesibles e inclusivas, aumentando la confianza de los clientes.
- **Seguridad:** Menor riesgo de vulnerabilidades que afecten los datos financieros.

Recomendaciones para mejora continua

1. Aplicar pruebas automatizadas en cada sprint, siguiendo un enfoque ágil.
2. Capacitar constantemente al equipo en herramientas de V&V y estándares internacionales.
3. Incluir métricas de calidad en los reportes (cobertura de pruebas, tiempos de respuesta, vulnerabilidades).
4. Establecer retroalimentación continua entre usuarios, desarrolladores y testers.

Referencias a estándares utilizados

- **IEEE 829:** Documentación de casos de prueba.
- **IEEE 1012:** Procesos de Verificación y Validación.
- **WCAG 2.1 (Nivel AA):** Accesibilidad en aplicaciones web y móviles.
- **OWASP Top 10 / OWASP ZAP:** Seguridad en aplicaciones.