UT7.2. Realización de pruebas

18 horas

Contenidos

1. El proceso de desarrollo.
   a. Fases
   b. Objetivo, importancia y limitaciones del proceso de prueba.
   c. Depuración de código.
2. Tipos de pruebas
   a. Pruebas de integración: ascendentes y descendentes.
   b. Pruebas de regresión.
   c. Pruebas funcionales.
   d. Pruebas de sistema: configuración, recuperación, entre otras.
   e. Pruebas de volumen y stress.
   f. Pruebas de uso de recursos.
   g. Pruebas de seguridad.
   h. Pruebas manuales y automáticas. Herramientas software para la realización de pruebas.
   i. Pruebas de usuario.
   j. Pruebas de aceptación.
3. Versiones alfa y beta.

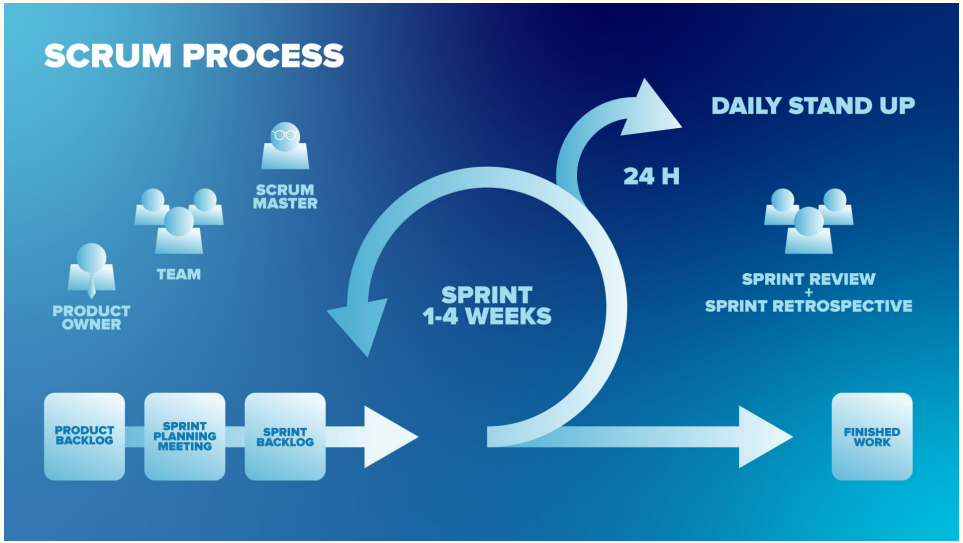
Resultados de aprendizaje

RA8: Evalúa el funcionamiento de aplicaciones diseñando y ejecutando pruebas	**15%**
a) Se ha establecido una estrategia de pruebas.	15%
b) Se han realizado pruebas de integración de los distintos elementos.	15%
c) Se han realizado pruebas de regresión.	15%
d) Se han realizado pruebas de volumen y estrés.	15%
e) Se han realizado pruebas de seguridad.	15%
f) Se han realizado pruebas de uso de recursos por parte de la aplicación.	15%
g) Se ha documentado la estrategia de pruebas y los resultados obtenidos.	10%

Pruebas de regresión

Las pruebas de regresión son imprescindibles en el desarrollo software actual.

Se usan para verificar los cambios (corrección de errores, mejoras, nuevas funcionalidades…)

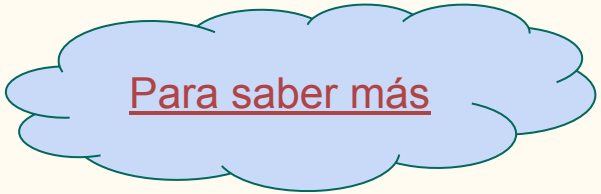


Si quieres profundizar un poco más en el tema: Estrategias de testing - Pruebas de Regresion

Pruebas de regresión

Imagina que entregas tu aplicación terminada pero el cliente quiere una nueva funcionalidad, la programas y creas una batería de pruebas para testear esta nueva funcionalidad y ¿ahora vuelves a ejecutar todas las pruebas que hiciste para la aplicación inicial?

Las pruebas de regresión testean los casos en los que la nueva funcionalidad afecta al código existente.



Por qué son útiles las pruebas de regresión

Ayudan a detectar y prevenir estos efectos dominó → Generan estabilidad y confianza en el usuario.

El nuevo código a menudo trae consigo efectos secundarios no deseados.

Uno de los problemas más frustrantes en el mantenimiento de software es ver cómo reaparece un antiguo error.

Antes de las pruebas de regresión	**Después de las pruebas de regresión**
Los antiguos errores reaparecen de forma impredecible.	Los problemas solucionados permanecen verificados de forma permanente.
Las correcciones de errores dependen de la memoria o la documentación.	Las pruebas confirman automáticamente las correcciones anteriores.
Los equipos de control de calidad repiten las comprobaciones manuales.	Los guardias automatizados evitan que se repita.

Tipos y técnicas de pruebas de regresión

1. Vuelve a comprobar todo → Se vuelve pesado cuando los sistemas crecen. Se utiliza a menudo al principio del ciclo de vida de un proyecto o cuando se valida una refactorización importante.

2. Pruebas de regresión selectivas → Se centra únicamente en las áreas directamente afectadas por los cambios recientes. Por ejemplo, si cambia un módulo de transformación de datos, es posible que no sea necesario volver a ejecutar las pruebas relacionadas con la autenticación o la representación de la interfaz de usuario.

3. Regresión priorizada en reposo → Se clasifican los casos de prueba por importancia y nivel de riesgo, lo que garantiza que las rutas críticas (como los pagos o la autenticación) se verifiquen en primer lugar.

4. Automatización → Las pruebas de regresión manuales se vuelven rápidamente insostenibles a medida que los proyectos crecen. Las suites automatizadas se ejecutan en cada compilación o solicitud de extracción, proporcionan comentarios instantáneos y se integran con herramientas de CI como Jenkins, GitHub Actions o GitLab CI/CD. Permiten una verificación continua en todos los entornos, lo cual es una capacidad esencial para los equipos ágiles y DevOps que buscan «avanzar rápidamente sin causar daños».

Ejemplos

- Comprobar la capacidad de respuesta de un sitio web en diferentes plataformas tras realizar cambios de diseño.
- Un sistema para reservar reservas de aerolíneas añade una nueva funcionalidad que permite pagos con tarjeta de débito además de los pagos con tarjeta de crédito permitidos anteriormente. Verificar que el flujo de pagos con tarjeta de crédito sigue funcionando según lo previsto.
- Se añaden nuevas características a una aplicación y luego queremos determinar cómo la nueva funcionalidad afecta a la velocidad de funcionamiento, si esos tiempos de carga han aumentado.
- Al parchear una vulnerabilidad en el módulo de login, se ejecutan pruebas para garantizar que los usuarios aún pueden registrarse.
- Se optimizan las consultas (queries) en el módulo de finanzas, se prueban los módulos de ventas y reportes.
- Al actualizar la API de un tercero...

Actividad 3

Veamos un ejemplo de pruebas de regresión utilizando metodología Scrum. Abre el documento *Prueba de regresión JUnit y Scrum.pdf* del Aula Virtual.

Pruebas de volumen y estrés

Se realizan para evaluar cómo un sistema maneja grandes cargas de datos y condiciones extremas.

Prueba de volúmen: Se enfocan en la capacidad de almacenamiento, procesamiento y respuesta del sistema. Por ejemplo probar una base de datos con millones de registros para ver si las consultas siguen siendo eficientes.

Prueba de estrés: Evalúan cómo el sistema responde cuando se somete a cargas extremas o condiciones anormales. El objetivo es identificar en qué punto falla y cómo se recupera. Por ejemplo probar una web de venta de entradas con miles de peticiones.

	Pruebas de volúmen	Pruebas de estrés
Objetivo	Evaluar el rendimiento bajo **cargas normales** o esperadas.	Evaluar el comportamiento y rendimiento bajo **condiciones extremas** o más allá de los límites esperados.
Tipo de carga	Cargas progresivas y **realistas**.	Cargas extremas y **no realistas**.
Nivel de carga	Subir **gradualmente** la carga hasta alcanzar el límite deseado.	Mantener la carga constante y **máxima** durante un período prolongado.
Duración de la prueba	Puede variar desde minutos **hasta horas**.	Puede extenderse durante horas o **días**.
Objetivo principal	Identificar **cuellos de botella** y problemas de rendimiento.	Evaluar la capacidad y **resistencia** del sistema.
Resultados esperados	**Tiempos de respuesta aceptables** bajo cargas normales.	**Rendimiento estable** y resistente bajo cargas extremas.

	Pruebas de volúmen	Pruebas de estrés
Puntos de fallo	Suelen ocurrir alrededor del **límite de capacidad** prevista.	Suelen ocurrir en condiciones cercanas o **más allá de los límites** de capacidad.
Preparación para la prueba	Simular **escenarios realistas** basados en patrones de uso.	Simular **escenarios improbables** pero posibles.
Optimizaciones antes de la prueba	**Ajustar configuraciones y recursos** para mejorar el rendimiento bajo cargas normales.	Asegurar que el **sistema ya esté optimizado** y preparado para enfrentar cargas extremas.
Objetivos de rendimiento	**Cumplir** con los niveles de servicio esperados bajo carga normal.	**Demostrar** que el sistema puede mantenerse **estable** y operativo bajo cargas máximas.
Importancia de las pruebas	Importantes **en etapas tempranas** y antes del lanzamiento.	Importantes para garantizar la resistencia **en momentos críticos** y eventos de alta demanda.

Casos reales

Pruebas de Estrés con Chaos Engineering

Netflix es famoso por su enfoque en pruebas de estrés para garantizar que su plataforma siga funcionando incluso en situaciones de fallo extremo.

- Usan Chaos Monkey, una herramienta que apaga aleatoriamente servidores en producción para ver cómo el sistema se recupera.
- Realizan pruebas de carga extrema simulando millones de usuarios viendo contenido simultáneamente.

Gracias a estas pruebas, su sistema puede manejar picos de tráfico masivos, como lanzamientos de series (Ejemplo: "Stranger Things").

Casos reales

Pruebas de Volumen en AWS

Amazon necesita asegurarse de que AWS puede manejar enormes volúmenes de datos y solicitudes de clientes globales.

- Pruebas de volumen en bases de datos como DynamoDB y S3, procesando billones de transacciones por día.
- Simulación de tráfico masivo en eventos como Black Friday y Prime Day.

AWS puede manejar cargas impredecibles sin afectar la velocidad ni la disponibilidad.

Casos reales

Pruebas de Carga en Mercado Libre

Mercado Libre enfrenta picos de tráfico en días de descuentos masivos como el Cyber Monday.

- Usan JMeter y Gatling para simular miles de usuarios comprando al mismo tiempo.
- Pruebas de estrés en su plataforma de pagos (Mercado Pago) para validar que miles de transacciones ocurran simultáneamente.

Capacidad para manejar picos de hasta 2 millones de visitas por segundo sin interrupciones.

Actividad 4

Veamos un ejemplo de pruebas de volumen y estrés. Abre el documento del Aula Virtual: *Prueba de volumen y estrés Spring, JMeter y Maven.pdf*