DESARROLLO E INTEGRACIÓN CONTINUA

UNIDAD II

Sección 2 »

La necesidad de la Integración Continua

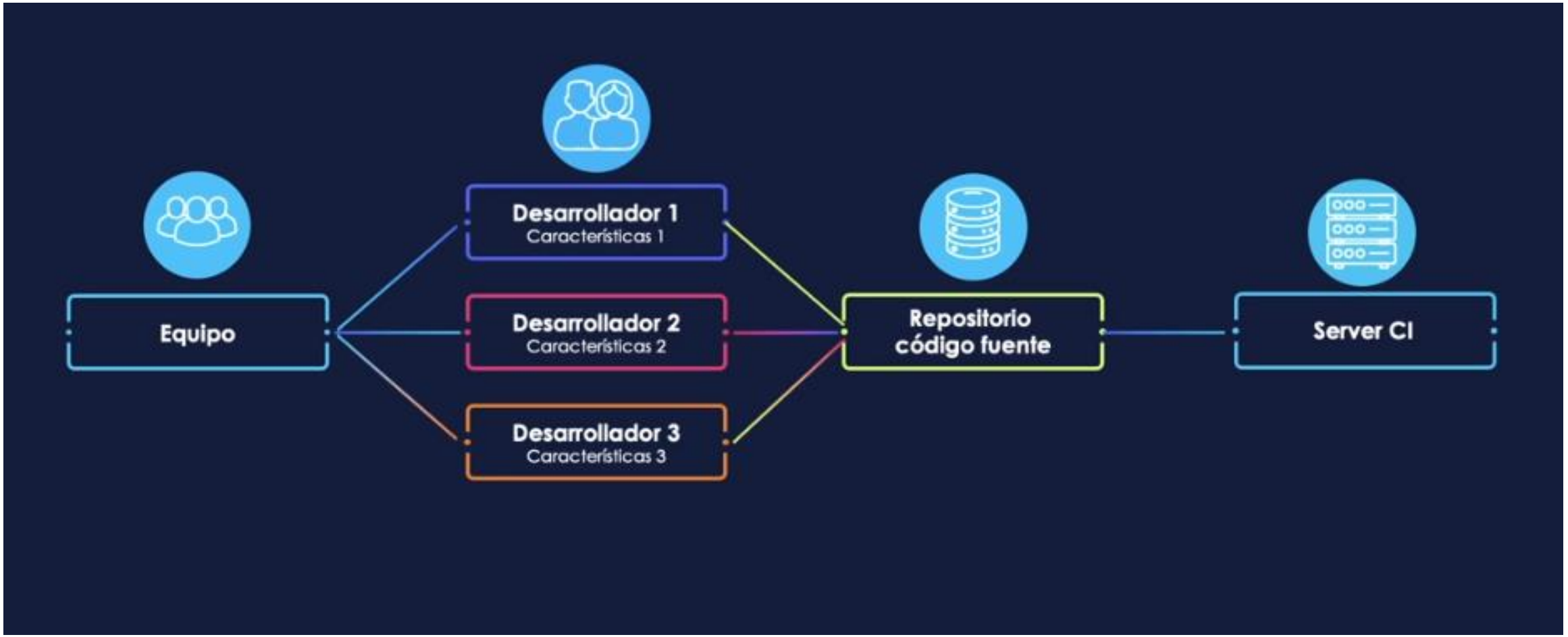

Características
Eliminando código
1
Características
Agregando código eliminado
2
Características
Nuevo código
3
Equipo


Características
Línea 1 eliminada
1
Características
Línea 1 conservada
2
Equipo

CONFLICTO DE FUSIÓN
MERGE CONFLICT

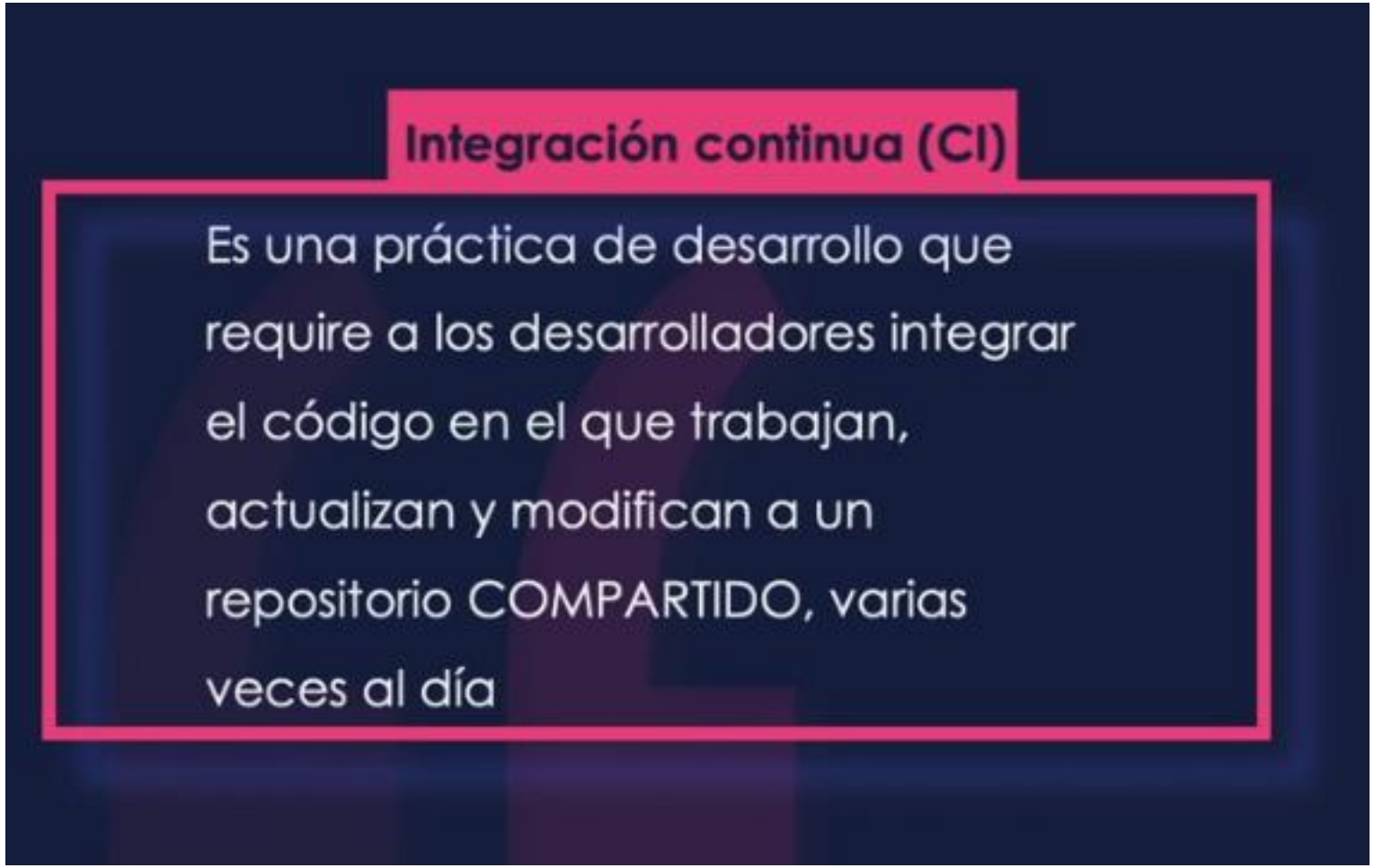

MERGE HELL

INTEGRACIÓN CONTINUA

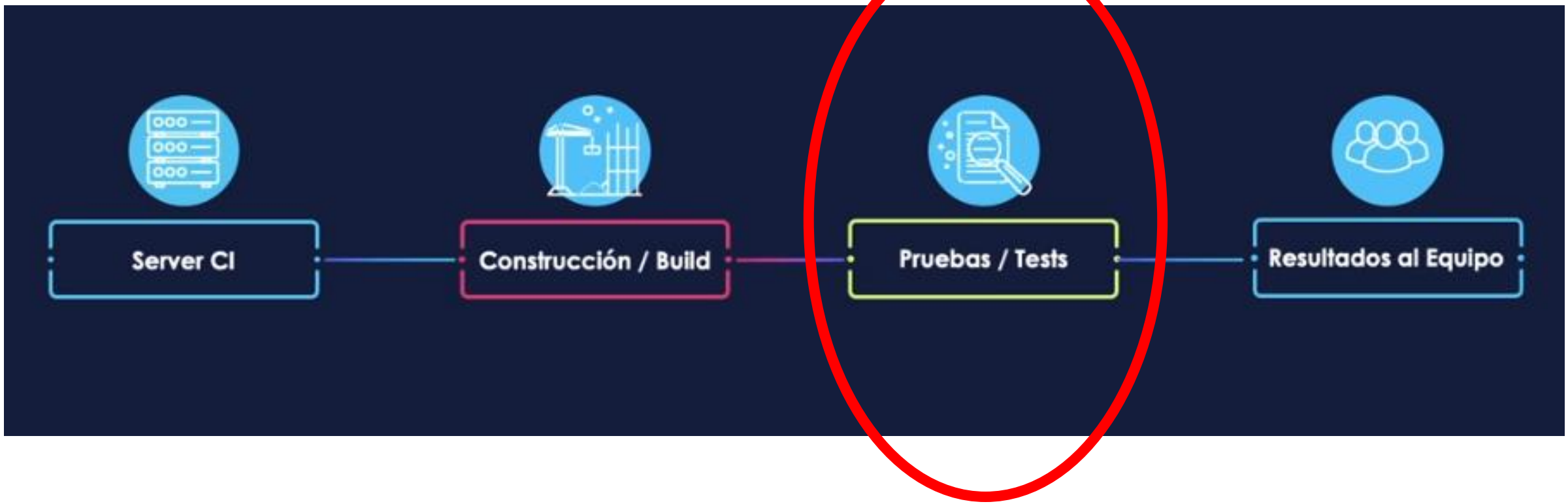



Server CI
Construcción / Build
Pruebas / Tests
Resultados al Equipo

DEFINICIÓN


Integración continua (CI)
Es una práctica de desarrollo que require a los desarrolladores integrar el código en el que trabajan, actualizan y modifican a un repositorio COMPARTIDO, varias veces al día

PRUEBAS / TEST



Pruebas comunes en DevOps

- Unitarias,
- análisis de código estático,
- aceptación,
- integración,
- regresión,
- rendimiento,
- esfuerzo.
- 5 herramientas para automatización de pruebas.

PRUEBAS COMUNES EN DEVOPS

UNITARIAS

- Su objetivo es probar unidades individuales de código fuente, con lo que se logra determinar si son aptas para su uso.

- Generalmente las escriben y ejecutan los desarrolladores de software para garantizar que el código cumpla con su diseño y se comporte según lo previsto.

Algunos beneficios de las pruebas unitarias son:

Facilita el cambio: estas pruebas unitarias sirven para que el programador compruebe si un fragmento del código continúa funcionando de manera correcta.

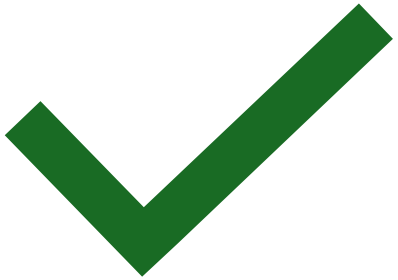
Ayuda a simplificar la integración: A pesar de que en las pruebas unitarias no se incluyen las pruebas de integración, estas dependen en gran medida de las pruebas realizadas de manera manual por el desarrollador. Una prueba de alto nivel puede resultar difícil de automatizar, por lo que las pruebas manuales son importantes.

Una apropiada documentación: las pruebas unitarias proporcionan una especie de documentación viva del sistema. Los desarrolladores que deseen saber qué funcionalidad proporcionan una unidad y cómo usarla pueden consultar las pruebas unitarias para obtener una comprensión básica de la API de la unidad.

Influye en el diseño de la aplicación o página: si una aplicación o software se desarrolla utilizando un enfoque basado en pruebas unitarias, esta puede reemplazar por completo al diseño formal ya que cada prueba puede verse como un elemento de diseño que especifica clases y comportamiento observable.

ANÁLISIS DE CÓDIGO ESTÁTICO

Una herramienta de análisis estático escanea el código en busca de errores y vulnerabilidades comunes conocidas, como fugas de memoria o desbordamientos de búfer.



El análisis de código estático garantiza que las comprobaciones se realicen de **forma coherente** y proporciona feedback rápido sobre los últimos cambios.

ACEPTACIÓN

- Las pruebas de aceptación eran cruciales para determinar si el software cumplía las expectativas y era viable para sus usuarios, es decir, si era aceptable y apto para salir al mercado. Sólo si lo era, se añadía a un entorno de producción y se utilizaba en el curso normal de la actividad empresarial.

- En el desarrollo ágil, las pruebas de aceptación forman parte del proceso y no son una ocurrencia tardía. Sin embargo, la intención sigue siendo la misma: verificar que el software cumple las expectativas desde el punto de vista del cliente y de los usuarios finales.

INTEGRACIÓN

1

Se encargan de verificar que los componentes del programa o aplicación están unidos e interactúan de manera eficiente.

2

Las pruebas de integración o pruebas integradas se definen como un mecanismo de testeo de *software*, donde se realiza un **análisis de los procesos relacionados con el ensamblaje o unión de los componentes**, sus comportamientos con múltiples partes del sistema (ya sea de archivos operativos) o de *hardware*, entre otras.

3

De modo que las pruebas de integración **están a cargo del examen de las interfaces entre los subsistemas** o los grupos de componentes del programa o aplicación que se analiza, lo que contribuye a garantizar su funcionamiento correcto.

REGRESIÓN

- Las pruebas de regresión se enfocan en verificar que los cambios, como las correcciones de errores, las mejoras de las funciones y las nuevas funciones añadidas no afecten negativamente las funciones existentes o la funcionalidad. Normalmente se realizan después de un cambio de código relevante o de haber desarrollado una nueva versión del software, y se emplean para asegurar que las funciones existentes sigan respondiendo según lo previsto. Las pruebas de regresión ayudan a garantizar que los nuevos cambios no generen un nuevo error ni provoquen fallos en las funciones existentes.

RENDIMIENTO

Las pruebas de rendimiento son una técnica de pruebas de software no funcional que determina cómo se mantiene la estabilidad, velocidad, escalabilidad y capacidad de respuesta de una aplicación bajo una carga de trabajo determinada. Es un paso clave para garantizar la calidad del software, pero, por desgracia, a menudo se considera una ocurrencia tardía, aislada y que comienza una vez que se han completado las pruebas funcionales y, en la mayoría de los casos, después de que el código esté listo para su publicación.

Los objetivos de las pruebas de rendimiento incluyen la evaluación del rendimiento de la aplicación, la velocidad de procesamiento, la velocidad de transferencia de datos, el uso del ancho de banda de la red, el número máximo de usuarios simultáneos, la utilización de la memoria, la eficiencia de la carga de trabajo y los tiempos de respuesta de los comandos.

ESFUERZO O DE ESTRÉS

- El objetivo final de las pruebas de esfuerzo es la capacidad de recuperación, lo que significa garantizar que el sistema se recupere sin problemas después de la falla. Las pruebas de estrés analizan el sistema y el comportamiento del usuario para determinar la causa raíz del fallo del sistema y tomar medidas en función de los errores y los datos recopilados durante las pruebas.

Algunas de las razones que justifican la realización de pruebas de resistencia.

- Determinar la estabilidad y confiabilidad del sitio web o la aplicación en condiciones de tráfico intenso.
- Mostrar el mensaje de error respectivo y otra información a los visitantes.
- Optimización del sistema para evitar averías.
- Planificar correctamente la escalabilidad y los recursos necesarios.

HERRAMIENTAS PARA PRUEBAS

Testing de unidad (Unit testing)

- Es el proceso de **probar cada componente individual del software**. (Un componente puede ser una función, una clase o un módulo).
- Este tipo de testing se utiliza para verificar si cada unidad del software funciona correctamente.
- Los testing de unidad se realizan generalmente en la **etapa de desarrollo**.
- También es útil para **reducir el tiempo y el costo de las pruebas de integración y sistema**, ya que permite detectar errores de forma temprana en el proceso de desarrollo.

Herramientas

JUNIT:

HERRAMIENTA DE TESTING DE UNIDAD PARA JAVA. ES FÁCIL DE USAR Y SE INTEGRA BIEN CON LOS FRAMEWORKS DE DESARROLLO DE JAVA.

NUNIT:

HERRAMIENTA DE TESTING DE UNIDAD PARA .NET. TIENE UNA SINTAXIS SIMILAR A JUNIT Y ES FÁCIL DE USAR.

PYTEST:

HERRAMIENTA DE TESTING DE UNIDAD PARA PYTHON. SU USO ES SENCILLO Y SE INTEGRA BIEN CON LOS FRAMEWORKS DE DESARROLLO DE PYTHON.

Testing de sistema (System testing)

- El **testing de sistema** es el proceso de **probar todo el sistema de software en su conjunto**.
- Este tipo de testing se utiliza para verificar si el software cumple con los requisitos funcionales y no funcionales, como **el rendimiento, la seguridad y la usabilidad**.
- Las pruebas de sistema se realizan generalmente en la **etapa final del desarrollo** antes de que se libere la aplicación.

Herramientas

Selenium:

Herramienta de testing de integración para aplicaciones web. Se utiliza para pruebas de interfaz de usuario o pruebas funcionales en aplicaciones web.

HP Quality Center:

Herramienta de gestión de testing de sistema que ayuda a planificar, diseñar y ejecutar pruebas de sistema. Ofrece una amplia gama de funcionalidades para la gestión de pruebas de sistema.

IBM Rational Functional Tester:

Herramienta de testing de sistema para aplicaciones Java y .NET. Su utilización es sencilla y ofrece una amplia gama de funcionalidades para el testing de sistema.

Testing de integración (Integration testing)

- El **testing de integración** es el proceso de **probar cómo los diferentes componentes del software interactúan entre sí**.
- Este tipo de testing se utiliza para **detectar errores de integración de forma temprana** en el proceso de desarrollo.
- Las pruebas de integración son necesarias para garantizar que el software sea coherente y que todos los componentes estén funcionando correctamente en conjunto.

Herramientas

Selenium:

Aunque se utiliza principalmente para pruebas de interfaz de usuario en aplicaciones web, también se puede utilizar para pruebas de integración. Concretamente, se emplea en contextos donde la interacción entre distintos sistemas o módulos se produce a través de la interfaz web.

Apache JMeter:

Herramienta de testing de carga y de integración. Aunque su fortaleza principal es en el testing de carga, también se utiliza para pruebas de integración, especialmente, para validar las interacciones entre sistemas a nivel de protocolo y servicio.

Testing de aceptación (Acceptance testing)

- El **testing de aceptación** es el proceso de **probar si el software cumple con los requisitos del usuario y del negocio**.
- Este tipo de testing se utiliza para **verificar si el software es adecuado para su uso en el mundo real**.
- Las pruebas de aceptación suelen ser realizadas por el cliente o el usuario final y pueden incluir **pruebas de funcionalidad, rendimiento y seguridad**.

Herramientas

Cucumber: herramienta de testing de aceptación para aplicaciones web. Se utiliza para escribir pruebas en lenguaje natural y es fácil de entender para los no técnicos.

FitNesse: herramienta de testing de aceptación para aplicaciones web. Se utiliza para escribir pruebas en lenguaje natural y es fácil de entender para los no técnicos.

Testing de humo (Smoke testing)

- El **testing de humo** es el proceso de **realizar pruebas básicas en el software para asegurarse de que está listo para ser probado más exhaustivamente**.
- Este tipo de testing se utiliza para **detectar errores graves** que pueden impedir el testing más exhaustivo.
- El testing de humo se realiza generalmente después de la compilación del software o después de cualquier actualización importante.

Herramienta

Jenkins:

Herramienta de integración continua que se utiliza para realizar el testing de humo. Se integra bien con otras herramientas de testing y se puede utilizar para automatizar el proceso de testing de humo.

Jenkins

Testing de regresión (Regression testing)

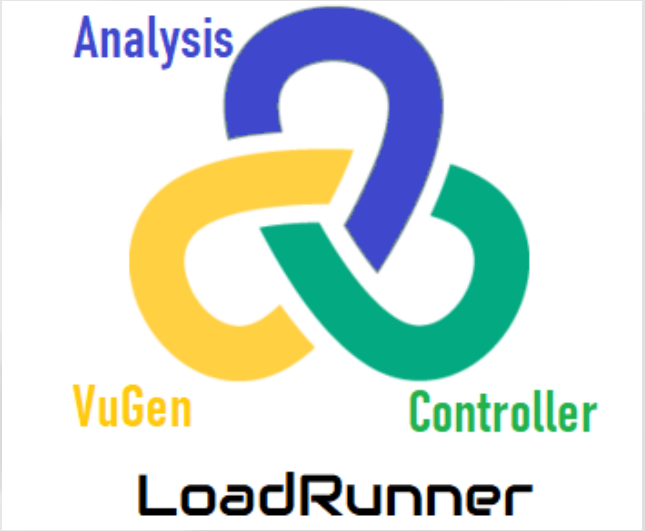
- El **testing de regresión** es el proceso de **volver a probar el software después de que se han realizado cambios en el mismo**.
- Este tipo de testing se utiliza para asegurarse de que **los cambios no han afectado el funcionamiento del software en otras áreas**.
- Las pruebas de regresión suelen ser automatizadas para **ahorrar tiempo y costos**.

- **TestComplete**: herramienta de testing de regresión que se utiliza para probar aplicaciones de escritorio, web y móviles. Es fácil de usar y ofrece una amplia gama de funcionalidades para el testing de regresión.

- **Ranorex**: herramienta de testing de regresión que se utiliza para probar aplicaciones de escritorio, web y móviles. Su uso es sencillo y ofrece una amplia gama de funcionalidades para el testing de regresión.

Testing de carga (Load testing)

- El **testing de carga** es el proceso de **probar cómo funciona el software bajo una carga específica**.
- Este tipo de testing se utiliza para verificar cómo se comportará un sistema o aplicación cuando se enfrenta a un volumen elevado de usuarios o transacciones.
- Su objetivo principal es asegurarse de que el software **mantiene un rendimiento óptimo y proporciona una experiencia de usuario satisfactoria**, incluso en condiciones de alta demanda.
- También se utiliza para identificar cuellos de botella, limitaciones o fallos en el sistema, permitiendo a los desarrolladores tomar **medidas correctivas** antes de lanzar el producto al mercado.



Apache JMeter:

Como se mencionó anteriormente, es una herramienta de testing de carga y de integración. Se utiliza para probar el rendimiento de la aplicación bajo diferentes cargas y condiciones.

LoadRunner:

Herramienta de testing de carga que se utiliza para probar aplicaciones web, de escritorio y móviles. Es una herramienta sencilla que ofrece una amplia gama de funcionalidades para el testing de carga.

RESUMEN

- En resumen, hay muchas herramientas de testing de software disponibles en el mercado y cada una de ellas tiene sus propias fortalezas y debilidades.
- La elección de la herramienta adecuada dependerá de las necesidades específicas del proyecto y el tipo de pruebas que se quieran realizar.
- Elegir la herramienta correcta es esencial para garantizar una implementación efectiva de las pruebas y, finalmente, para **asegurar la calidad y fiabilidad del software entregado**.

ACTIVIDAD 2
Unidad 2

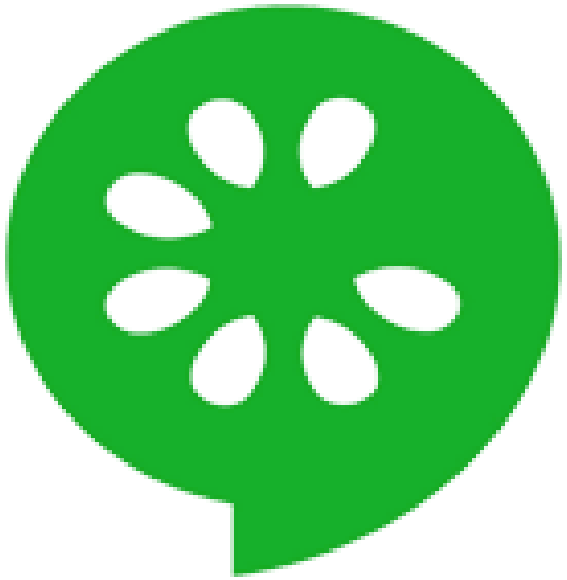
Presentación de las características y funcionalidad de la herramienta

Martes 11 de junio

Testing de unidad (Unit testing)

Testing de sistema (System testing)

Testing de integración (Integration testing)

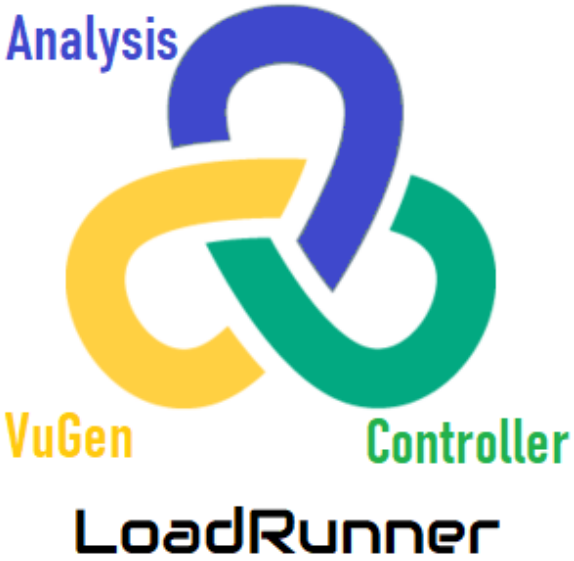


Testing de aceptación (Acceptance testing)

Testing de humo (Smoke testing)

Jenkins

Testing de regresión (Regression testing)



Testing de carga (Load testing)