Manual técnico para desarrolladores principiantes de Java y Linux

Contents

Chapter 1. Integración continua

Esta sección describe cómo utilizar la disciplina de integración continua.

Este proceso implica construir, validar, verificar, empaquetar e implementar el activo de software en un repositorio corporativo.

Integración Continua

Práctica de desarrollo de software

La Integración Continua (CI) es una práctica de desarrollo de software en la cual los desarrolladores integran su código en un repositorio compartido varias veces al día.

El objetivo de la CI es detectar y corregir problemas de integración tan pronto como sea posible, asegurando que el código siempre esté en un estado funcional.

Mediante el uso de la CI, los equipos de desarrollo pueden automatizar el proceso de compilación, pruebas e implementación, mejorando así la calidad del software y acelerando el ciclo de desarrollo.

Herramientas de integración continua

Información sobre herramientas comunes utilizadas en la práctica de Integración Continua.

Jenkins

Jenkins es una herramienta de automatización de código abierto ampliamente utilizada para la implementación de CI y entrega continua (Continuous Delivery - CD). Puede encontrar más información en https://www.jenkins.io/.

Sonatype Nexus

Sonatype Nexus es una plataforma de gestión de repositorios de artefactos que se puede integrar con Jenkins para la entrega continua de artefactos de software. Puede encontrar más información en https://www.sonatype.com/nexus/repository-pro.

SonarQube

SonarQube es una herramienta de análisis estático de código que se utiliza en el proceso de integración continua para detectar problemas de calidad de código y vulnerabilidades. Puede encontrar más información en https://www.sonarqube.org/.

Testes Unitarios

Concepto sobre la práctica de realizar pruebas unitarias en el desarrollo de software.

Los testes unitarios son una práctica fundamental en el desarrollo de software.

Consisten en escribir y ejecutar pruebas automatizadas para verificar el funcionamiento de unidades de código a nivel de método o clase.

Los testes unitarios ayudan a identificar y corregir errores de manera temprana, garantizando la integridad y la calidad del código.

Testes Unitarios con Maven y JUnit 5

Información sobre la práctica de pruebas unitarias en Java utilizando Maven y JUnit 5.

Introducción

Las pruebas unitarias son una práctica fundamental en el desarrollo de software para garantizar la calidad y el funcionamiento correcto de las unidades de código.

Maven es una herramienta de automatización de compilación y gestión de dependencias ampliamente utilizada en el ecosistema Java.

JUnit 5 es un framework de pruebas unitarias para Java, conocido por su flexibilidad y poderosas funcionalidades.

Configuración

Para escribir y ejecutar pruebas unitarias con Maven y JUnit 5 en un proyecto Java, es necesario configurar el entorno de desarrollo correctamente y definir las dependencias apropiadas en el archivo de configuración del proyecto (pom.xml).

Con la configuración adecuada, las pruebas unitarias pueden ser escritas utilizando las anotaciones y clases proporcionadas por JUnit 5 y ejecutadas utilizando Maven a través del comando "mvn test".

Automatización

Mediante la integración entre Maven y JUnit 5, los desarrolladores pueden automatizar el proceso de ejecución de pruebas unitarias, garantizando la calidad del software desarrollado.

Chapter 2. Clonar un Proyecto de un Repositorio Git

Concepto sobre la acción de clonar un proyecto desde un repositorio Git.

Clonar un proyecto de un repositorio Git es el proceso de crear una copia local del repositorio en tu máquina.

Esta acción te permite trabajar con el código del proyecto, realizar cambios y contribuciones.

Clonar un Proyecto de un Repositorio Git

Tarea que describe cómo clonar un proyecto desde un repositorio Git.

1. Abre tu terminal o línea de comandos.
2. Navega al directorio donde deseas clonar el proyecto.
3. Ejecuta el comando "git clone [URL del repositorio]".
4. Presiona Enter y espera a que se complete el proceso de clonación.

Chapter 3. Compilar una Aplicación Java

Concepto sobre cómo compilar una aplicación Java.

Compilar una aplicación Java implica preparar tu entorno de desarrollo instalando y configurando el JDK (Java Development Kit).

Para compilar una aplicación Java, necesitas seguir varios pasos, incluyendo la instalación y configuración del JDK, así como configurar las variables de entorno.

Compilar una Aplicación Java en Linux

Tarea que describe cómo compilar una aplicación Java en un sistema Linux.

Siga estos pasos para instalar, configurar y compilar una aplicación Java con JDK 17 o superior en el entorno Linux

1. Descarga el JDK 17 o una versión posterior desde el sitio web oficial de Oracle o de un proveedor de confianza.
2. Instala el JDK en tu sistema Linux siguiendo las instrucciones proporcionadas por el proveedor o utilizando el gestor de paquetes de tu distribución.
3. Configura la variable de entorno JAVA_HOME apuntando al directorio de instalación del JDK.
4. Agrega la ruta del directorio bin del JDK al PATH del sistema.
5. Abre una terminal y navega hasta el directorio donde se encuentra el código fuente de tu aplicación Java.
6. Compila tu aplicación Java utilizando el comando "javac NombreDelArchivo.java".

Chapter 4. Testes Unitarios

Concepto sobre la práctica de realizar pruebas unitarias en el desarrollo de software.

Los testes unitarios son una práctica fundamental en el desarrollo de software.

Consisten en escribir y ejecutar pruebas automatizadas para verificar el funcionamiento de unidades de código a nivel de método o clase.

Los testes unitarios ayudan a identificar y corregir errores de manera temprana, garantizando la integridad y la calidad del código.

Ejecutar Testes Unitarios con Maven y JUnit

Tarea que describe cómo ejecutar testes unitarios en una aplicación Java utilizando Maven y JUnit.

Esta tarea asume que tienes Maven instalado en tu sistema y un proyecto Java configurado con JUnit para pruebas unitarias.

1. Abre una terminal o línea de comandos.
2. Navega hasta el directorio raíz de tu proyecto Java.
3. Ejecuta el comando "mvn test" para compilar el proyecto y ejecutar los testes unitarios.
4. Espera a que Maven descargue las dependencias, compile el código y ejecute los testes.
5. Verifica la salida de la consola para ver los resultados de los testes.

Chapter 5. Análisis de Calidad de Código Fuente

Concepto sobre la práctica de realizar análisis de calidad de código fuente en proyectos de software.

El análisis de calidad de código fuente es una actividad importante en el desarrollo de software para evaluar la salud general del código y detectar posibles problemas y áreas de mejora.

Esta práctica implica el uso de herramientas automatizadas que examinan el código fuente en busca de violaciones de estándares de codificación, bugs, vulnerabilidades de seguridad, y otros problemas que puedan afectar la calidad y el rendimiento del software.

El análisis de calidad de código fuente ayuda a los equipos de desarrollo a identificar y corregir problemas tempranamente, mejorar la mantenibilidad del código y reducir el riesgo de errores en el software.

Chapter 6. Trabajar en una Rama de Función y Realizar la Entrega en la Rama Principal

Tarea que describe cómo un desarrollador puede trabajar en una feature branch y realizar la entrega en la branch principal a través de una merge request.

Esta tarea asume que estás utilizando un sistema de control de versiones como Git y una plataforma de colaboración como GitLab o GitHub.

1. Crea una nueva rama de función (feature branch) en tu repositorio de Git.
2. Trabaja en la rama de función para implementar la nueva funcionalidad o corrección de errores.
3. Realiza commits frecuentes en tu rama de función para guardar tus cambios de forma incremental.
4. Cuando tu trabajo en la rama de función esté completo y listo para ser integrado en la rama principal, crea una solicitud de extracción (merge request) en tu plataforma de colaboración.
5. Asigna revisores a tu solicitud de extracción para que puedan revisar tu código y proporcionar comentarios si es necesario.
6. Realiza los cambios sugeridos por los revisores y actualiza tu solicitud de extracción según sea necesario.
7. Cuando tu solicitud de extracción haya sido aprobada por los revisores, realiza el merge (combinación) de tu rama de función en la rama principal (main o master) de tu repositorio de Git.
8. Verifica que tus cambios se hayan integrado correctamente en la rama principal y que la funcionalidad esté funcionando como se espera.