INVESTIGACIÓN GITHUB, GRADLE Y MAVEN

ASTRID CAROLINA DÍAZ GÓMEZ

ARQUITECTURA DE SOFTWARE

INGENIERÍA DE SISTEMAS

UDEA

2020-1

**Introducción**

**Objetivo General**

**Objetivos Específicos**

**Breve marco teórico**

**Procedimiento**

**GitHub**

El objetivo principal de GitHub.com es facilitar el control de versiones de problemas del desarrollo de software y gestión de proyectos. Un sistema de control de versiones es el que ayuda a registrar los cambios realizados a un código, registra quién realizó el cambio y puede restaurar el código borrado o modificado. Se debe tener en cuenta que github no es solo para programadores ya que permite administrar su proyecto de manera eficiente y trabajar en colaboración en proyectos que necesite actualizaciones constantes

Para el control de versiones, Git, desarrollado por Linus Torvalds, permite solicitudes de extracción para proponer cambios en el código fuente. Los usuarios con la capacidad de revisar los cambios propuestos pueden ver una diferencia de los cambios solicitados y aprobarlos, es bueno saber que todos los comandos estándar de Git funcionan con en GitHub. En la terminología de Git, esta acción se llama "committing" y una instancia de ella es un "commit", además se guarda un historial de todas las confirmaciones y se puede ver más adelante. Y por otro lado está Hub, ésta convierte a GitHub en la red social más grande para desarrolladores, ya que además de contribuir con un proyecto permite socializar con usuarios con ideas afines, permite seguir a las personas y ver qué hacen y con quien conectan.

Algunos de los términos importantes para poder entender GitHub son:

* Repositorio: Es un directorio donde se almacenan los archivos de tu proyecto. Puede estar ubicado en el almacenamiento de GitHub o en un repositorio local en tu computadora. Puedes almacenar archivos de código, imágenes, audios o todo lo relacionado con el proyecto en el repositorio.
* Branch: Que se traduce como rama, es una copia de tu repositorio se puede utilizar la rama cuando quieras hacer un desarrollo de forma aislada. Trabajar en esta no afectará el repositorio central u otras ramas. Cuando se haya completado el trabajo en la rama, puedes combinarla con otras ramas y con el repositorio central mediante un pull request.
* Pull Request: Significa que les informas a los demás que has enviado al repositorio principal el cambio que hiciste en una rama, los colaboradores del repositorio pueden aceptar o rechazar el pull request.
* Bifurcar un repositorio: Significa crear un nuevo proyecto basado en el repositorio existente. En términos simples, esto significa que copias un repositorio existente, haces los cambios necesarios, almacenas la nueva versión como un nuevo repositorio como un proyecto propio.

**Gradle**

Gradle es una herramienta que permite la automatización de compilación de código abierto centrada en la flexibilidad y el rendimiento, que bebe de las aportaciones que han realizado herramientas como ant y maven pero intenta llevarlo todo un paso más allá. Los scripts de compilación de Gradle se escriben utilizando Groovy o Kotlin DSL (Lenguaje específico de dominio) para trabajar con un lenguaje sencillo y claro a la hora de construir el build, además dispone de una gran flexibilidad que permite trabajar con ella utilizando otros lenguajes y no solo Java y también cuenta con un sistema de gestión de dependencias muy estable.

Gradle es altamente personalizable, se modela de una manera que es personalizable y extensible de las formas más fundamentales, es rápido, completa las tareas rápidamente reutilizando los resultados de ejecuciones anteriores, procesando solo las entradas que cambiaron y ejecutando tareas en paralelo y es potente; es la herramienta de compilación oficial para Android y viene con soporte para muchos idiomas y tecnologías populares. Además, es el sistema de compilación oficial para Android y cuenta con soporte para diversas tecnologías y lenguajes.

Algunas de las características de Gradle son:

* Depuración colaborativa: Permite compartir los resultados de la compilación para resolver en equipo de forma eficiente posibles problemas que aparezcan.
* Construcción incremental: Valida en el proceso de compilación si la entrada, salida o implementación de una tarea ha cambiado, en caso de no existir algún cambio la considera actualizada y no se ejecuta.
* Diseño de repositorio personalizado: Podremos tratar prácticamente cualquier estructura de directorios del sistema de archivos.
* Dependencias transitivas: Es uno de los principales beneficios que ofrece al utilizar la gestión de dependencias ya que se encarga de descargar y administrar las dependencias transitivas.
* Soporte a Groovy y Scala incorporado: Compatibilidad con los proyectos de Groovy, permitiendo trabajar con código Groovy o código Scala e inclusive desarrollar código mixto Java y Groovy o Java y Scala.
* Compilación incremental para Java: En caso de que el código fuente o la ruta de clase cambien, Gradle cuenta con la capacidad para detectar todas las clases que se vean afectadas por dicho cambio y procederá a recompilarlas.
* Embalaje y distribución de JAR, WAR y EAR: Cuenta con herramientas para empaquetar el código basado en JVM (Java Virtual Machine) en archivos de archivo comunes.
* Integración con Android Studio: Android Studio no cuenta con un generador interno, sino que delega todas las tareas de compilación en Gradle, garantizando la corrección en todas las construcciones, ya sea que se ejecuten desde Android Studio, la línea de comandos o un servidor de construcción de integración continua.
* Soporte de MS Visual C ++ y GoogleTest: Gradle acepta la construcción con el compilador de Visual C de Microsoft en Windows. (VS 2010, VS 2013 y VS 2015 compatibles), así como también realizar pruebas de aplicaciones C con GoogleTest.
* Publicar en repositorios Ivy y Maven: Permite publicar Artifacts en repositorios Ivy con diseños de directorios completamente personalizables
* TestKit para pruebas funcionales: Permite la ejecución prágramática de builds inspeccionando los resultados de compilación, ésta es una prueba de compatibilidad entre versiones.
* Distribuciones personalizadas: En Gradle cada distribución cuenta con un directorio init.d en el que se pueden colocar scripts personalizados que pre configuran su entorno de compilación.
* Lee el formato POM: Es compatible con el formato de metadatos POM, por lo que es posible recuperar dependencias de cualquier repositorio compatible con Maven.
* Compara builds: Resalta de forma rápida las diferencias entre compilaciones, lo que hace que el análisis de la causa raíz sea mucho más rápido y eficaz.
* Compilador daemon: Gradle crea un proceso de daemon que se reutiliza dentro de una compilación de múltiples proyectos, cuando necesita bifurcar el proceso de compilación, mejorando la velocidad de compilación.
* Personalizar y extender escaneos: Ofrece la opción de agregar sus propios datos para construir escaneos como etiquetas, valores y enlaces, integrando escaneos de compilación en la cadena de herramientas.
* Caché de dependencia de terceros: Las dependencias de repositorios remotos se descargan y almacenan en caché localmente, las compilaciones posteriores utilizan los artifacts almacenados en caché para evitar el tráfico de red innecesario.

**Maven**

Apache Maven es una herramienta de gestión y comprensión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la construcción, informes y documentación de un proyecto a partir de una pieza central de información.

El concepto de Maven es aglutinar conocimiento y reutilización de código, estandarización y una buena definición de configuraciones de proyectos.

Basado en el concepto de un modelo de objeto de proyecto: las compilaciones, la gestión de dependencias, la creación de documentación, la publicación del sitio y la publicación de la distribución se controlan desde el archivo declarativo pom.xml. Maven puede ampliarse mediante complementos para utilizar una serie de otras herramientas de desarrollo para la elaboración de informes o el proceso de compilación.

Apache Maven es una herramienta que estandariza la configuración de un proyecto en todo su ciclo de vida, como por ejemplo en todas las fases de compilación y empaquetado y la instalación de mecanismos de distribución de librerías, para que puedan ser utilizadas por otros desarrolladores y equipos de desarrollo.

También contempla temas relacionados con la integración continua, para poder realizar la ejecución de test unitarios y pruebas automatizadas, test de integración, etc.

Principales características de Maven

Básicamente, Apache Maven no deja de ser la base de los compiladores actuales, de IDES como Eclipse, NetBeans o IntelliJ, a los que ofrece soporte gracias a algunas de sus características, como por ejemplo:

Un sistema de gestión dependencias.

Un mecanismo distribuido de distribución de librerías. El comportamiento distribuido es siempre desde el repositorio local de Maven hacia los repositorios que están publicados en Internet o en la red corporativa.

Mecanismos para ser extensible, por la creación de plugins customizables.

Es multi-plataforma, puede funcionar tanto en entornos Linux como Windows al ser una aplicación Java.

Es software libre, con lo cual es el código está disponible, se podría modificar y customizar en caso de que fuera necesario.

Fomenta la reutilización de código y de librerías. El hecho de que Apache Maven ofrezca repositorios oficiales y públicos de software libre, con librerías desplegadas, que toda la comunidad de desarrolladores de software utiliza, hace que este concepto también pueda trasladarse al mundo empresarial, a través de repositorios remotos corporativos, compartidos por distintos equipos de proyectos o el propio equipo de desarrollo.

Es compatible con múltiples IDEs.

Definición de Maven

Usando una traducción más ajustada, se podría decir que Maven es:

Una herramienta de gestión de proyectos que fomenta, a través de la definición de un Project Object Model (el fichero POM que se define en cada uno de los proyectos o módulos), un conjunto de estándares que definen el ciclo de vida del proyecto.

Un sistema de gestión de dependencias muy potente.

Una lógica de ejecución de plugins que permite, en determinadas las fases del ciclo de vida, poder ejecutar un plugin concreto que ofrezca una solución a un problema. Por ejemplo, antes de la ejecución de los test, en muchos casos interesa levantar un servidor de aplicaciones, ejecutar los tests y una vez hayan finalizado, bajar ese servidor para que no ocupe ningún puerto y para mejorar el rendimiento de la máquina.

**Conclusiones**

**Bibliografía**

<https://docs.github.com/es/github> (2020 – gihub)

<https://www.hostinger.es/tutoriales/que-es-github/> (Gustavo B. - mayo 13, 2019)

<https://en.wikipedia.org/wiki/GitHub>

<https://openwebinars.net/blog/que-es-gradle/>

<https://docs.gradle.org/current/userguide/userguide.html>

<https://www.arquitecturajava.com/que-es-gradle/>

<https://maven.apache.org>