



Ingeniería Web: Visión General (IWVG)

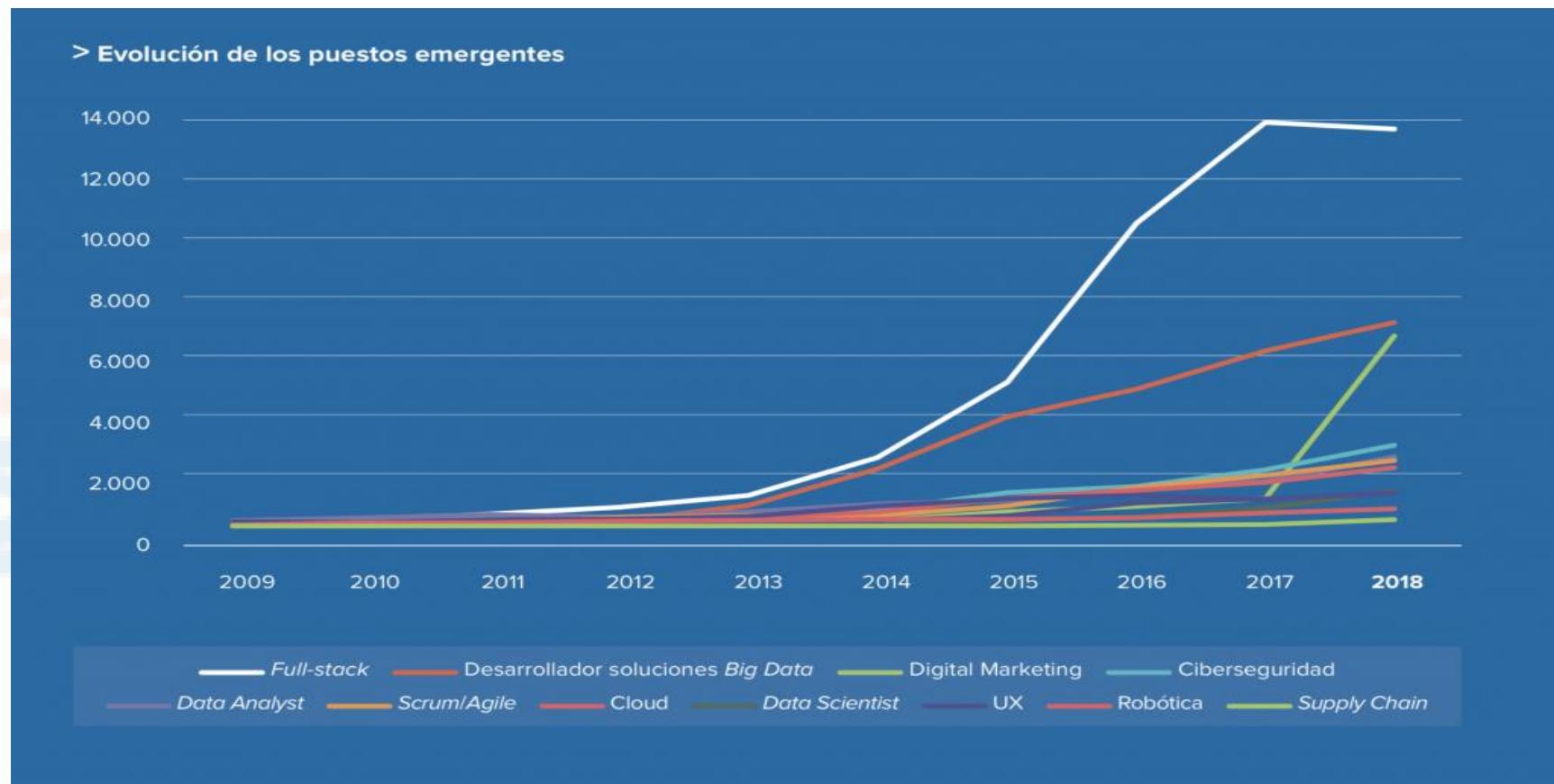
Ecosistema

<https://github.com/miw-upm/iwvg-ecosystem> (Teoría)

<https://github.com/miw-upm/iwvg-git-workflow> (Ejercicios-clase)

<https://github.com/miw-upm/iwvg-fork-workflow> (Ejercicios-clase)

Introducción



- <https://orientacion-laboral.infojobs.net/empleos-emergentes-con-mas-futuro>
- Perfil: **Full-stack (Front-end & Back-end)**
- Perfil: **Arquitecto QA (Quality Assurance)**

Introducción

■ Ecosistema software

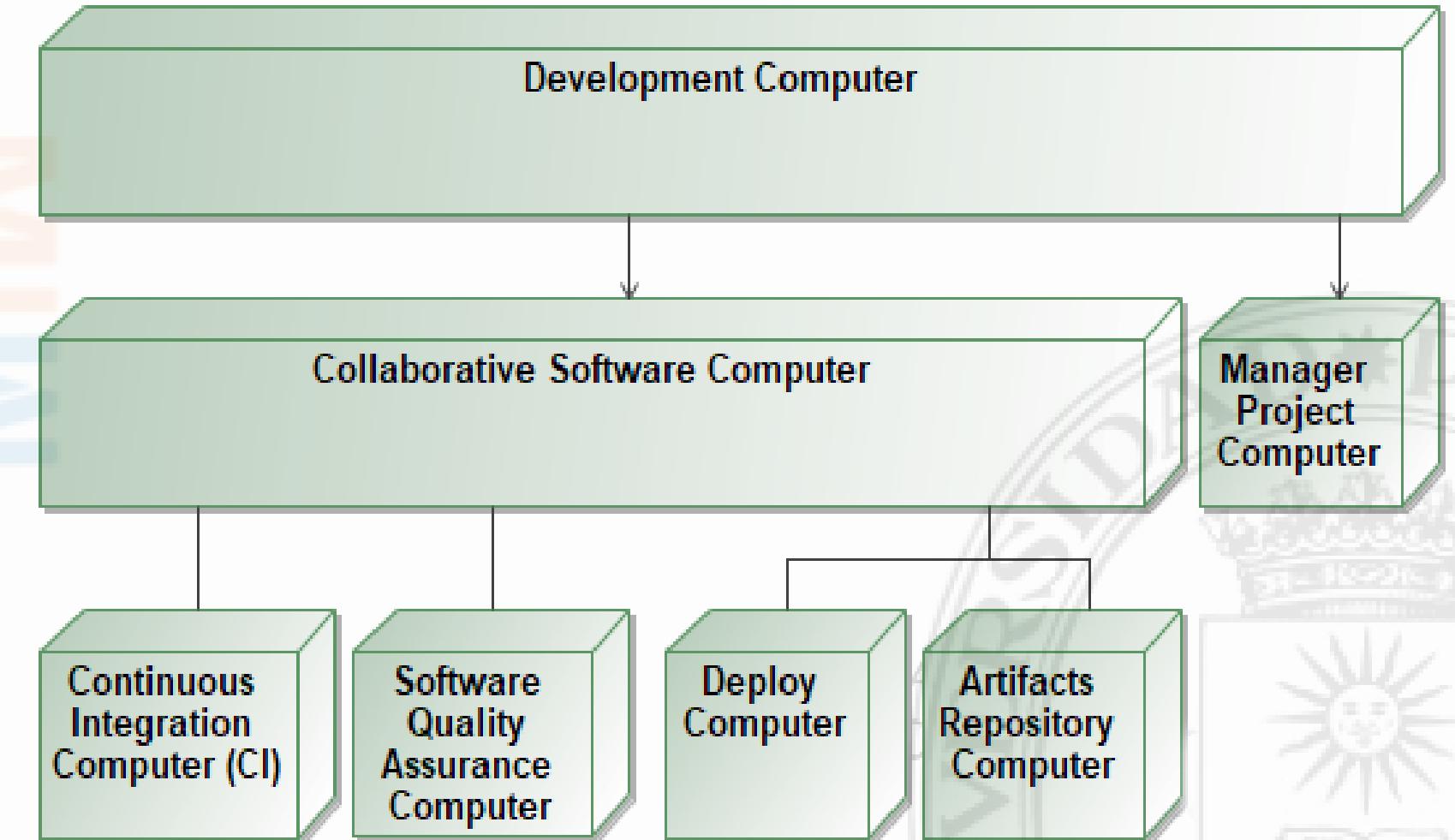
- Se define como un espacio de trabajo donde un conjunto de herramientas interactúan y funcionan como una unidad para el *desarrollo de software colaborativo* en todas sus fases (Gestión, Requisitos, Análisis, Diseño, Programación, Pruebas y Despliegue)

■ Entorno de Desarrollo Integrado (IDE)

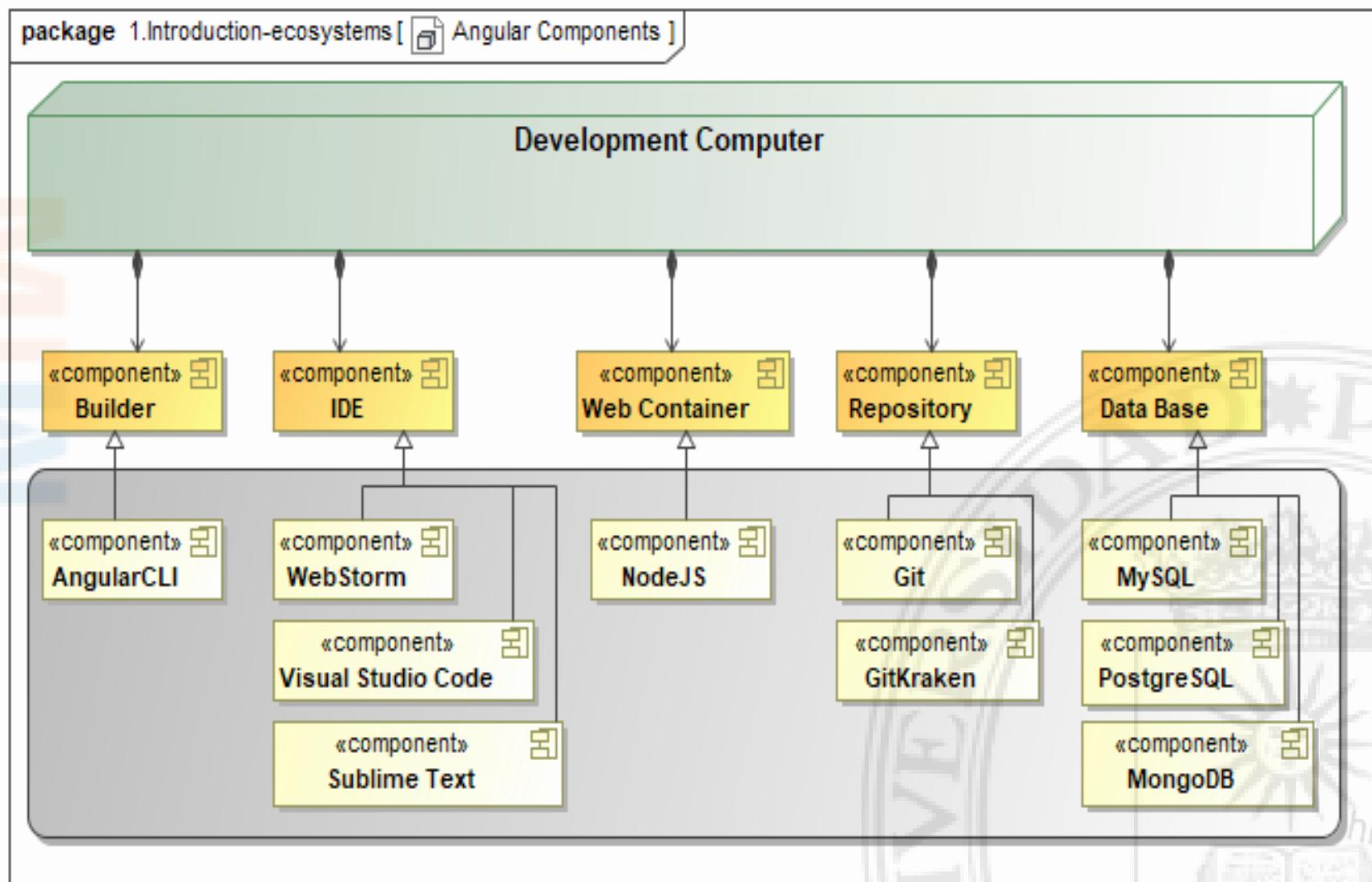
- Se define como una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador de software

Introducción

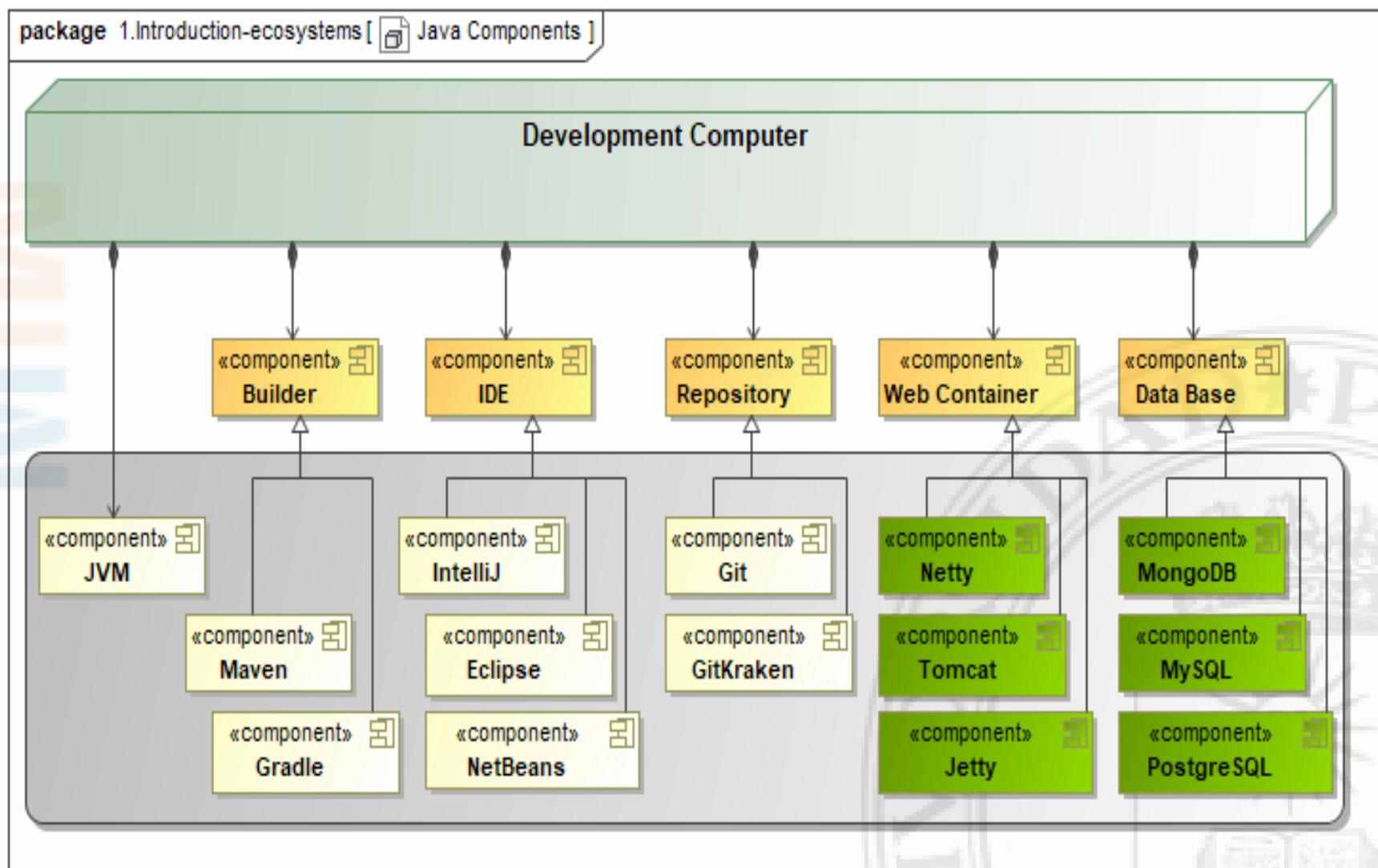
package 1.Introduction-ecosystems [ Server Computers]



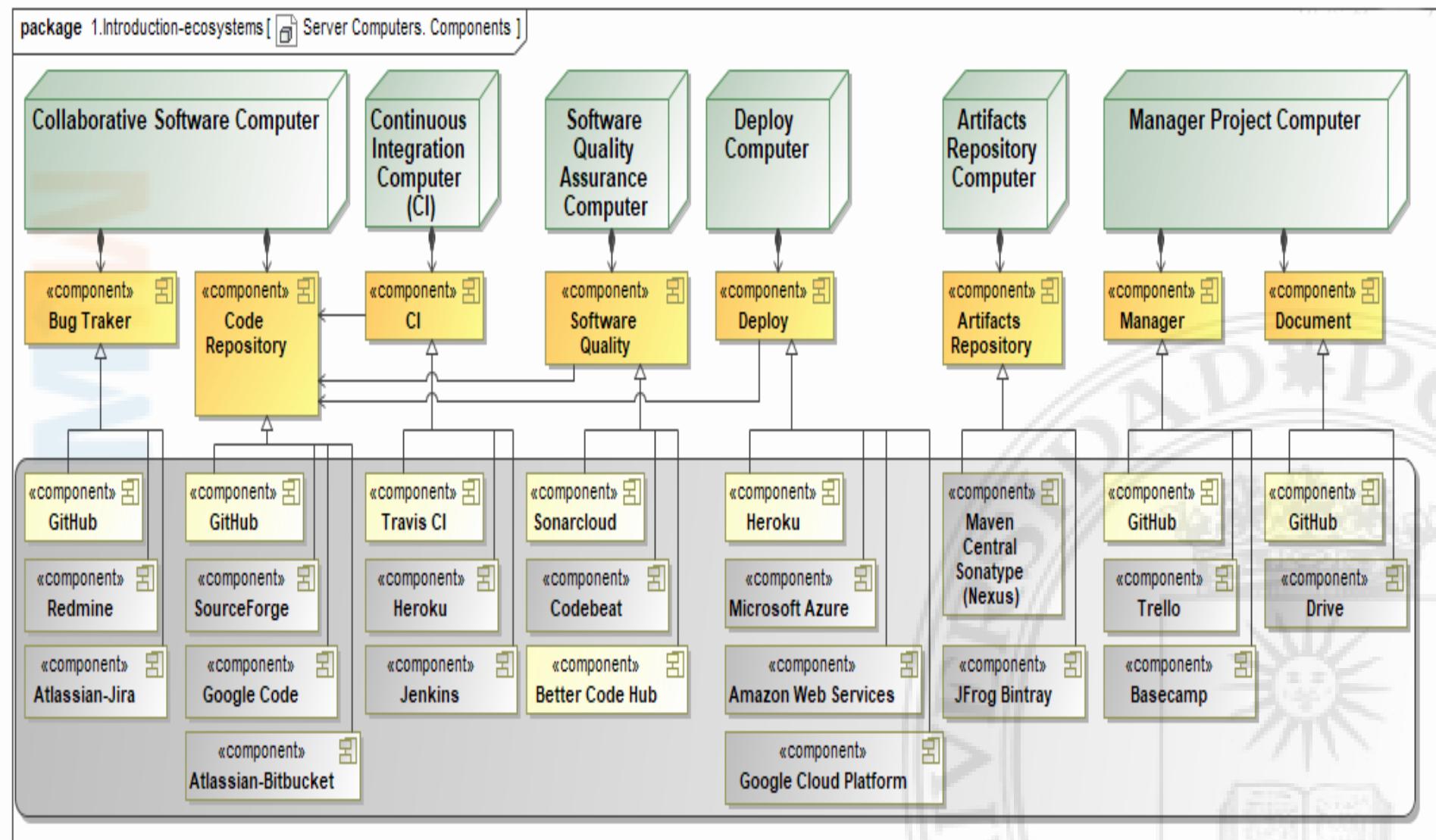
Equipo de desarrollo: Angular



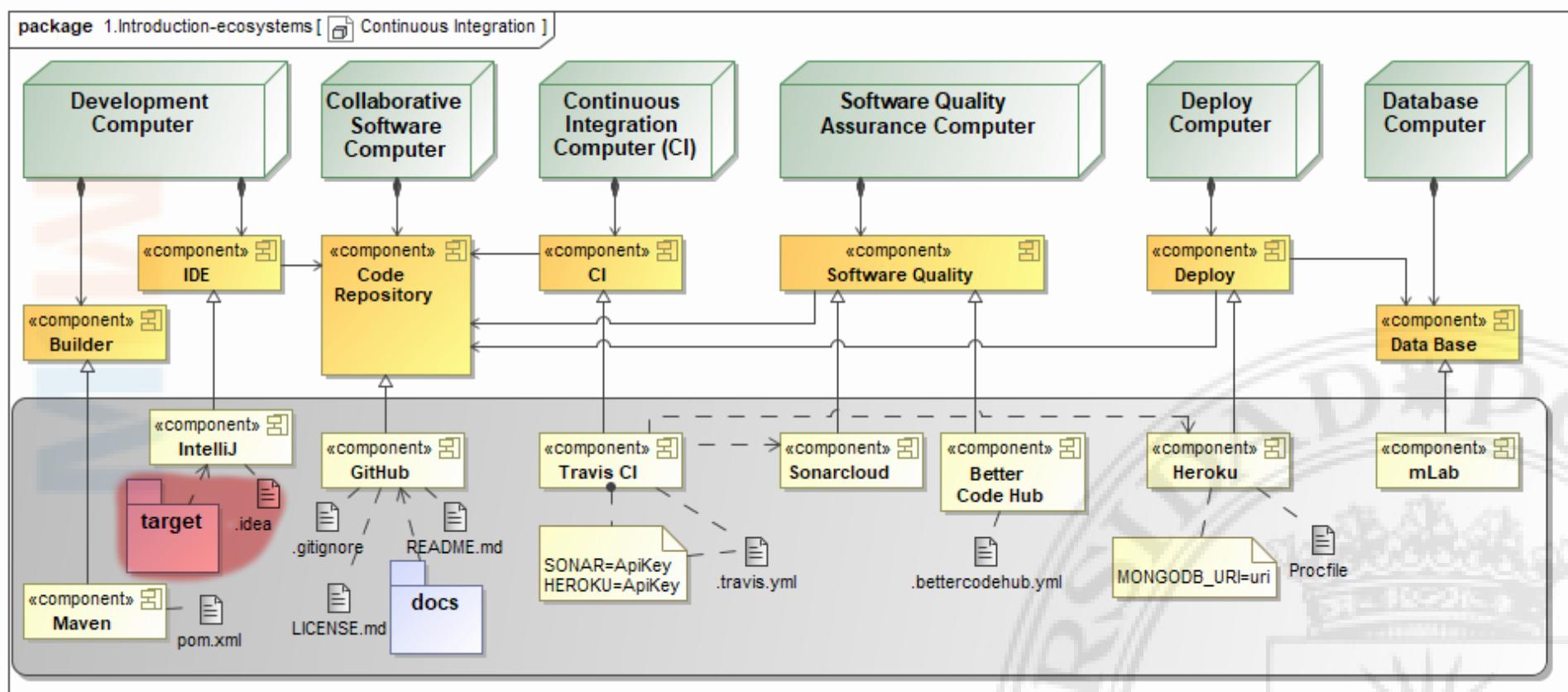
Equipo de desarrollo: Java



Software como Servicio (*SaaS*)



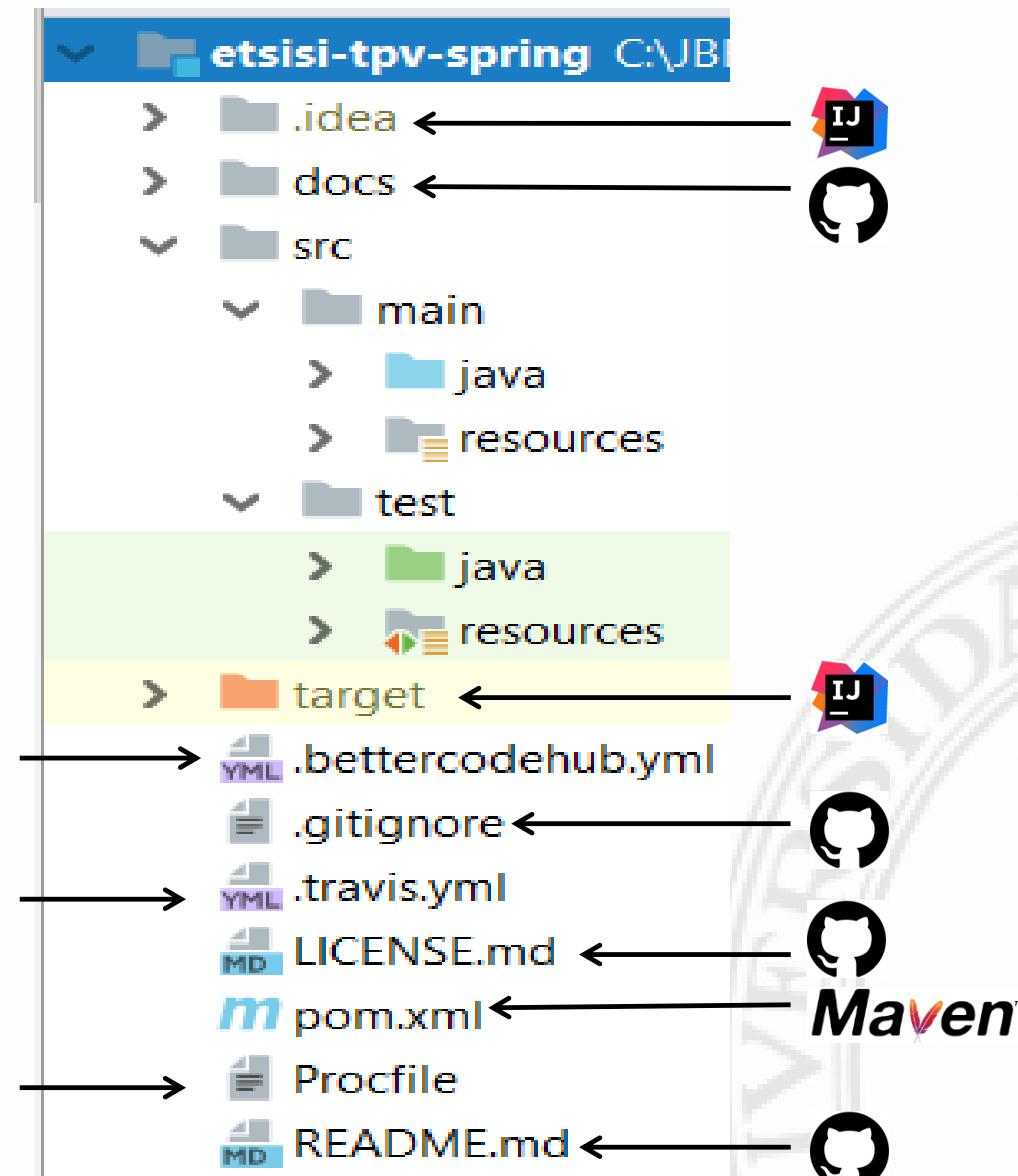
Plataformas



Maven™



Plataformas



Instalación del SDK de Java

- Para el desarrollo y compilación de aplicaciones Java, utilizaremos: Standard Edition (Java SE) o Java Development Kit (JDK) de Oracle:
 - <http://https://www.oracle.com/technetwork/java/index.html>
 - Para Instalar se realiza instalación de un exe sobre Windows
 - Se establece las variables de entorno *JAVA_HOME* (al raíz) Y *PATH* (al bin)
- Para la ejecución de aplicaciones Java es el *Java Runtime Environment* (JRE). Se instala automáticamente con Java SE

Construcción de proyectos

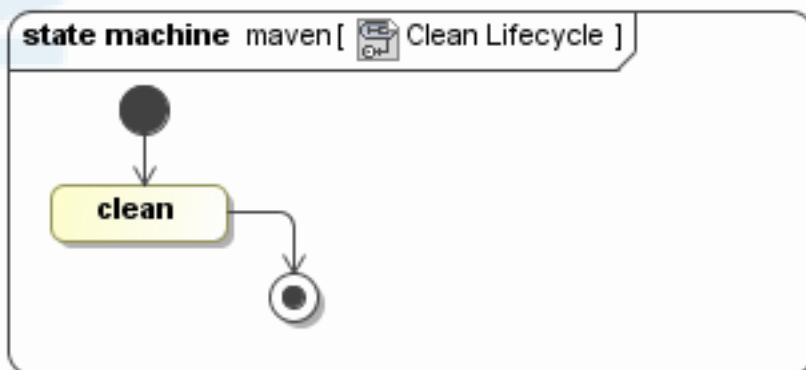
- La construcción del proyecto requiere realizar diferentes tareas como: compilación, pasar los test, pasar el control de calidad de fuentes, generar el jar, resolver las dependencias... Existen herramientas que nos ayudan y automatizan esta labor
 - *Maven* y *Gradle* (más reciente)
- *Apache Maven* es una herramienta de gestión de proyectos de software. Se basa en un modelo de objetos del proyecto (POM),
 - <http://maven.apache.org/>
 - Documentación: <http://www.sonatype.com/books/mvnref-book/reference/public-book.html>
- POM: Archivo(XML) donde se define el proceso de construcción, información del proyecto, y en general, cada una de las fases propuestas
- Instalación
 - <https://maven.apache.org/>
 - Bajar el zip y descomprimir (*Binary zip archive: apache-maven-3.6.1-bin.zip*)
 - Se establece la variable de entorno *M2_HOME* (al raíz) y *PATH* (al bin)
- Maven crea una carpeta llamada **.m2/repository** en el perfil del usuario que contiene una copia de los artefactos

Conceptos de Maven

- **Artefacto:** Es la unidad mínima con la que trabaja Maven para gestionar sus dependencias, son los componentes software
- **Coordenadas:** Sistema con el Maven determina de forma única a cada uno de sus artefactos: *groupId:artifactId:version*
 - *Group Id:* identificación del grupo. Normalmente se utiliza el nombre del dominio, al revés: *es.upm.miw*
 - *Artifact Id:* identificación del artefacto: *iwvg-ecosystem*
 - *Version:* versión del artefacto: *1.0.0-SNAPSHOT, 1.3.4-RC (Release Candidate), 1.4.5-Release*
- **Empaquetado:** tipo de fichero generado, normalmente *JAR (WAR, EAR, RAR...)*
- **Repositorio:** Estructura de directorios y archivos que usa Maven para almacenar, organizar y recuperar artefactos. Existen repositorios locales, privados y remotos
 - El repositorio por defecto es el repositorio central de Maven
 - Existe un repositorio privado en la empresa para los artefactos de desarrollo
 - Existe un repositorio local, donde se copian las dependencias:
%User%/.m2/repository
- **Arquetipos:** Son plantillas para definir proyectos tipo con el fin de ser reutilizados

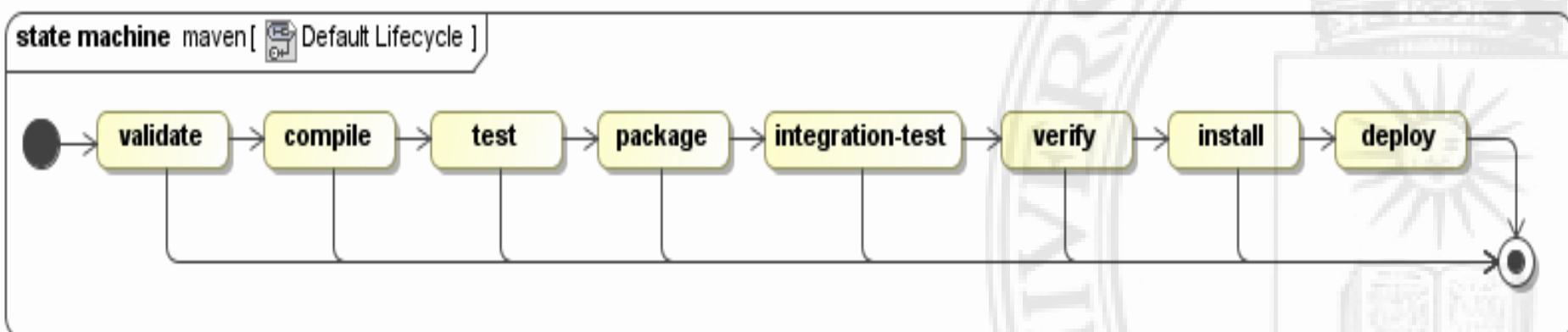
Ciclos de vida

- Clean Lifecycle
 - *clean* : Elimina todos los ficheros generados por construcciones anteriores
- Comandos
 - > *mvn clean*
 - > *mvnw clean* (*si se utiliza un Maven embebido -Wrapper- en el proyecto*)



Ciclos de vida

- Default Livecycle
 - *validate* : Valida el proyecto si es correcto
 - *compile* : Genera los ficheros .class compilando los fuentes .java
 - *test* : Ejecuta los test automáticos de JUnit existentes, abortando el proceso si alguno de ellos falla.
 - *package* : Genera el empaquetado final (jar, war...)
 - *integration-test*: Procesar y desplegar el paquete si fuera necesario en un entorno donde se puedan ejecutar pruebas de integración
 - *verify*: Ejecutar todas las comprobaciones para verificar la validez del paquete y que cumpla con los criterios de calidad
 - *install* : Copia el paquete en un directorio de nuestro ordenador, de esta manera pueden utilizarse en otros proyectos maven del mismo ordenador
 - *deploy* : Copia el paquete *jar* a un servidor remoto, poniéndolo disponible para cualquier proyecto maven con acceso a ese servidor remoto
- Comandos:
 - `> mvn -v`
 - `> mvn compile`
 - `> mvn -Dmaven.test.skip=true package`
 - `> mvn clean verify`



Plugin de objetivos

- Es una tarea específica, más pequeña que una fase de construcción, que contribuye a la construcción y gestión del proyecto
- En el ciclo de vida, va después de *package*
- Comando: *[nombre del plugin]:[función del plugin a ejecutar]*
 - *mvn org.jacoco:jacoco-maven-plugin:prepare-agent verify*
 - JaCoCo - Java Code Coverage Library, is a free Java code coverage library distributed under the Eclipse Public License.
 - Tambien se puede incorporar como *plugin*
 - *mvn sonar:sonar*
 - SonarQube is an open-source platform for continuous inspection of code quality
 - *mvn spring-boot:run*

IntelliJ IDEA

- IntelliJ IDEA es un entorno de desarrollo integrado (IDE), tiene una versión gratuita: *Community Edition*.
- Es una buena plataforma para el desarrollo de aplicaciones Java
 - <https://www.jetbrains.com/idea/download>
 - Versión: *Community*

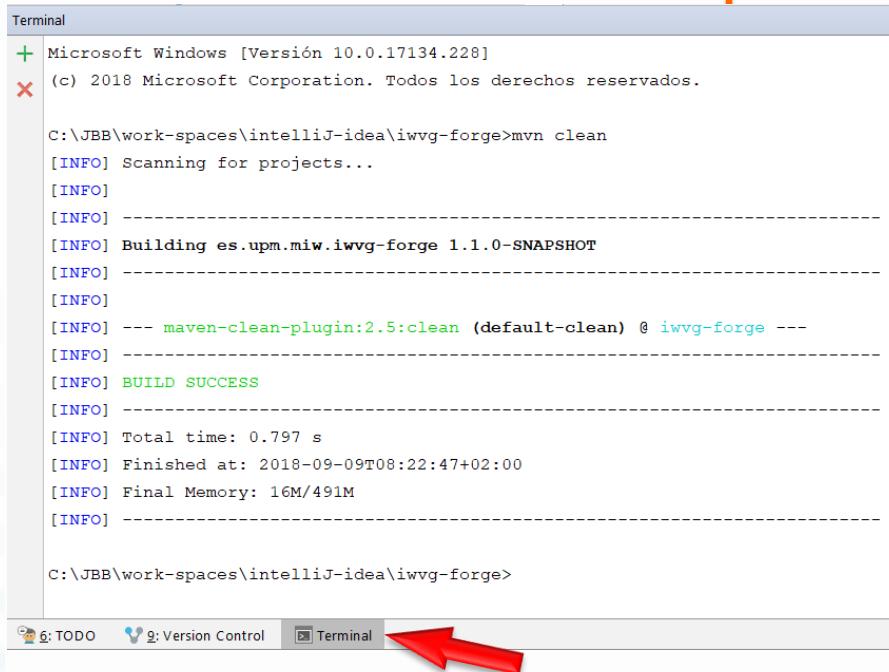
Cliente Git

- Git es un software de control de versiones, distribuido, gratuito y de código abierto
- Sitio Web: <https://git-scm.com>
- Documentación:
 - <https://git-scm.com/book>
 - <https://git-scm.com/book/es/v2>
- *Cliente:*
 - <https://git-scm.com/download/win>
- *GUI:*
 - <https://git-scm.com/downloads/guis/>
 - <https://www.gitkraken.com/>

✍ Instalación del Entorno. Primeros pasos

1. Instalar SDK de Java. Definir variables de entorno
2. Instalar *Maven*. Definir variables de entorno
 - Se manejará maven desde la consola, nos facilitará la creación de scripts
3. Instalar cliente Git
4. Instalar IntelliJ IDEA
5. Crear la carpeta de los workspace
6. Crear un proyecto Java con maven en el workspace. Se ofrece una plantilla a modo de ejemplo:
 - <https://github.com/miw-upm/iwvg-ecosystem>
 - Recordar cambiar el nombre de la carpeta y del proyecto en el fichero *pom.xml*
7. Importar el proyecto desde IntelliJ IDEA
 - *Cerrar proyecto si estuviese abierto*
 - *>>Import Project*, y seleccionar la carpeta del proyecto
 - > marcar *Create Project from external model*, elegir *Maven*
 - > *Next... Finish*
8. Probar comandos maven por consola

✍ Primeros pasos: terminal

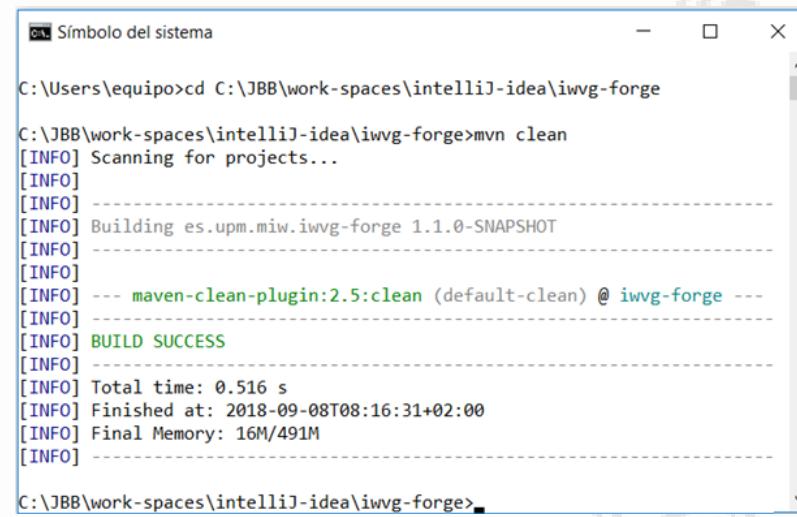


A screenshot of the IntelliJ IDEA interface. On the left, there's a vertical bar with colored sections (orange, yellow, green). At the top, a toolbar has icons for TODO, Version Control, and Terminal. A red arrow points to the Terminal icon. Below the toolbar is a tab bar with 'Terminal' selected. The main area shows a terminal window with the following text:

```
Microsoft Windows [Versión 10.0.17134.228]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\JBB\work-spaces\intelliJ-idea\iwvg-forge>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building es.upm.miw.iwvg-forge 1.1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ iwvg-forge ---
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 0.797 s
[INFO] Finished at: 2018-09-09T08:22:47+02:00
[INFO] Final Memory: 16M/491M
[INFO]
[INFO]

C:\JBB\work-spaces\intelliJ-idea\iwvg-forge>
```



A screenshot of a Windows Command Prompt window titled 'Símbolo del sistema'. The window shows the following text:

```
C:\Users\equipo>cd C:\JBB\work-spaces\intelliJ-idea\iwvg-forge
C:\JBB\work-spaces\intelliJ-idea\iwvg-forge>mvn clean
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building es.upm.miw.iwvg-forge 1.1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ iwvg-forge ---
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] -----
[INFO] Total time: 0.516 s
[INFO] Finished at: 2018-09-08T08:16:31+02:00
[INFO] Final Memory: 16M/491M
[INFO]
[INFO]

C:\JBB\work-spaces\intelliJ-idea\iwvg-forge>
```

✍ Clonar un proyecto Maven desde Github

- Clonar en repositorio en tu equipo:

1. Situarse en una carpeta raíz donde se encuentran los proyectos, mediante la consola:
 - `>cd %ruta-de-la-carpeta%`
2. Clonar el repositorio, se visualizará el contenido de la rama por defecto:
 - `>git clone https://github.com/miw-upm/iwvg-ecosystem`
 - Actualizar el repositorio local con los cambios del remoto
 - `>git fetch origin`

- Importar el proyecto desde IntelliJ IDEA

- *Cerrar proyecto si estuviese abierto*
- *>>Import Project*, y seleccionar la carpeta del proyecto
- *> marcar Create Project from external model, elegir Maven*
- *> Next... Finish*

Gestión de registros: Log4j

- Un log es un registro de un evento ocurrido en un instante dado. Se suele usar para registrar datos o información sobre quién, qué, cuándo, dónde y por qué (*who, what, when, where y why*) un evento ocurre en una aplicación.
- Log4j es una biblioteca *open source* desarrollada en Java por Apache que permite elegir la salida y el nivel de granularidad de los mensajes o *logs*.
- <http://logging.apache.org/>
- Niveles de prioridad
 - FATAL: mensajes críticos del sistema
 - ERROR: mensajes de error de la aplicación
 - WARN: mensajes de alerta, pero que no afectan al correcto funcionamiento
 - INFO: mensajes de información
 - DEBUG: se utiliza para escribir mensajes de depuración.
 - TRACE: se utiliza para mostrar mensajes con un mayor nivel de detalle que debug.
- *Appenders*. Puede una o varias salidas de destino
 - Consola (Console); Fichero (File); Base de datos (JDBC, JPA, MongoDB...); Correo (SMTP)
- Configuración. Permite varios sistemas de configuración (*properties, xml, yaml, ...*)
-  Ejemplo LoggerDemo
- ~~System.out.println()~~ queda encarecidamente desaconsejado!!!

Pruebas

- Tipos
 - **Pruebas Unitarias (**Test)**. Los desarrolladores prueban correcto funcionamiento de un módulo de código o clase independientemente del resto. Si existen dependencias se rompen con los *mocks*, son clases que simulan a otras y aportan una funcionalidad limitada
 - **Pruebas de Integración (**IT)**. Los desarrolladores prueban los diferentes componentes que dependen de otros componentes
 - **Pruebas Funcionales**. Los desarrolladores prueban al sistema como un todo
 - **Pruebas de Aceptación**. Los clientes prueban las versión entregada
- Característica de calidad:
 - Automática. Clases que prueban clases
 - Cobertura: % de líneas de código ejecutadas (>80%)
 - Repetibles. Cuando parte del código ha sido modificado, se vuelven a lanzar las pruebas para comprobar que no se ha alterado su funcionalidad
 - Independiente. Se prueban los módulos por separado

JUnit 5

- **JUnit** es un framework que nos ayuda a la realización de pruebas unitarias. Fue creado por Erich Gamma y Kent Beck.
 - <http://www.junit.org/>
- Test Case: Clases de prueba
 - Una clase es un test, si tiene algún método de test
 - Un test es un método con la anotación `@Test`
 - Los test deben ser independientes entre si, el orden no debe afectar al resultado
 - El nombre del método debe describir el tipo de prueba. Empiezan con la palabra “test”
- Test Suites: Contenedor de Test Case o Test Suites. Se crean estructura en árbol
- Ciclo de vida
 - `@BeforeAll`: Se ejecuta una sola vez antes de la batería de pruebas definida en la clase y el método debe ser *static*
 - `@BeforeEach`: Se ejecuta antes de cada uno de los marcados con `@Test` (es decir, si existen varios test, se ejecuta varias veces). Suele ser una inicialización por todas las pruebas de la clase
 - `@Test`: Marca un método como prueba
 - `@AfterEach`: Se ejecuta después de cada uno de los `@Test`. Suele ser una liberación de recursos
 - `@AfterAll`: Se ejecuta al final del proceso completo y el método debe ser *static*
 - `@Disabled`: Marca un método o una clase completa para que no se ejecute
 - `@Tag`. Para el filtrado de test
 - `@DisplayName`. Define un nombre del test propio
 - `@RepeatedTest`. Repetición de test
 - `@ParameterizedTest`. Define test parametrizados con diferentes valores

JUnit. Comprobaciones

- assertEquals (valor Esperado, valor Real);
 - Los valores pueden ser de cualquier tipo
 - Si son arrays, no se comprueban elemento a elemento, sólo la referencia
- assertEquals (double Esperado, double Real, double Error);
- assertNotEquals
- assertArrayEquals (array[] Esperado, array[] Real)
- assertTrue (condición booleana)
- assertFalse (condición booleana)
- assertSame (Objeto Esperado, Objeto Real)
 - Comprueba que son la misma referencia
- assertNotSame
- assertNull (Objeto)
 - Comprueba que el objeto es Null
- assertNotNull (Objeto Objeto)
 - Comprueba que el objeto no es Null
- fail ()
 - Falla siempre
- assertThrows
 - Se debe lanzar una excepción
- assertDoesNotThrow
 - No se debe lanzar una excepción
- assertTimeout

Organización

- Paquetes
 - Fuentes: `src/main/java/**`
 - Pruebas: `src/test/java/**`
 - Se puede lanzar un solo test, los test de una clase o los test de un paquete y subpaquetes
- Léxico
 - Pruebas Unitarias
 - Si la clase se llama `**Test`, se ejecutarán en la fase Maven de *test*
 - Pruebas de Integración
 - Si la clase se llaman `**IT`, se ejecutan en la fase Maven de *integration-test*
- Metodología de Trabajo para mantenimiento del código
 1. Modificar el código de la clase y los test afectados
 2. Ejecutar los test de la clase para comprobar que todo sigue igual
 3. Ejecutar el test del paquete, para comprobar que todo sigue igual
 4. Ejecutar todos los test
-  Ejemplos de test
 - Point: PointTest
 - DecimalCollection: DecimalCollectionTest
-  Lanzamiento de test con IntelliJ
-  Ejercicios: Crear los test para las clases del proyecto que faltan
 - Fraction
 - User

Repositorio de código. Introducción

■ Tipos

- Distribuidos. Aumenta la flexibilidad pero complica la sincronización y gestión. Ejemplos: *Git, Mercurial...* En la actualidad se están imponiendo estos sistemas
- Centralizados. Dependiente de un responsable. Facilita la gestión pero reduce la potencia y flexibilidad. Ejemplos: *CVS, Subversion...*

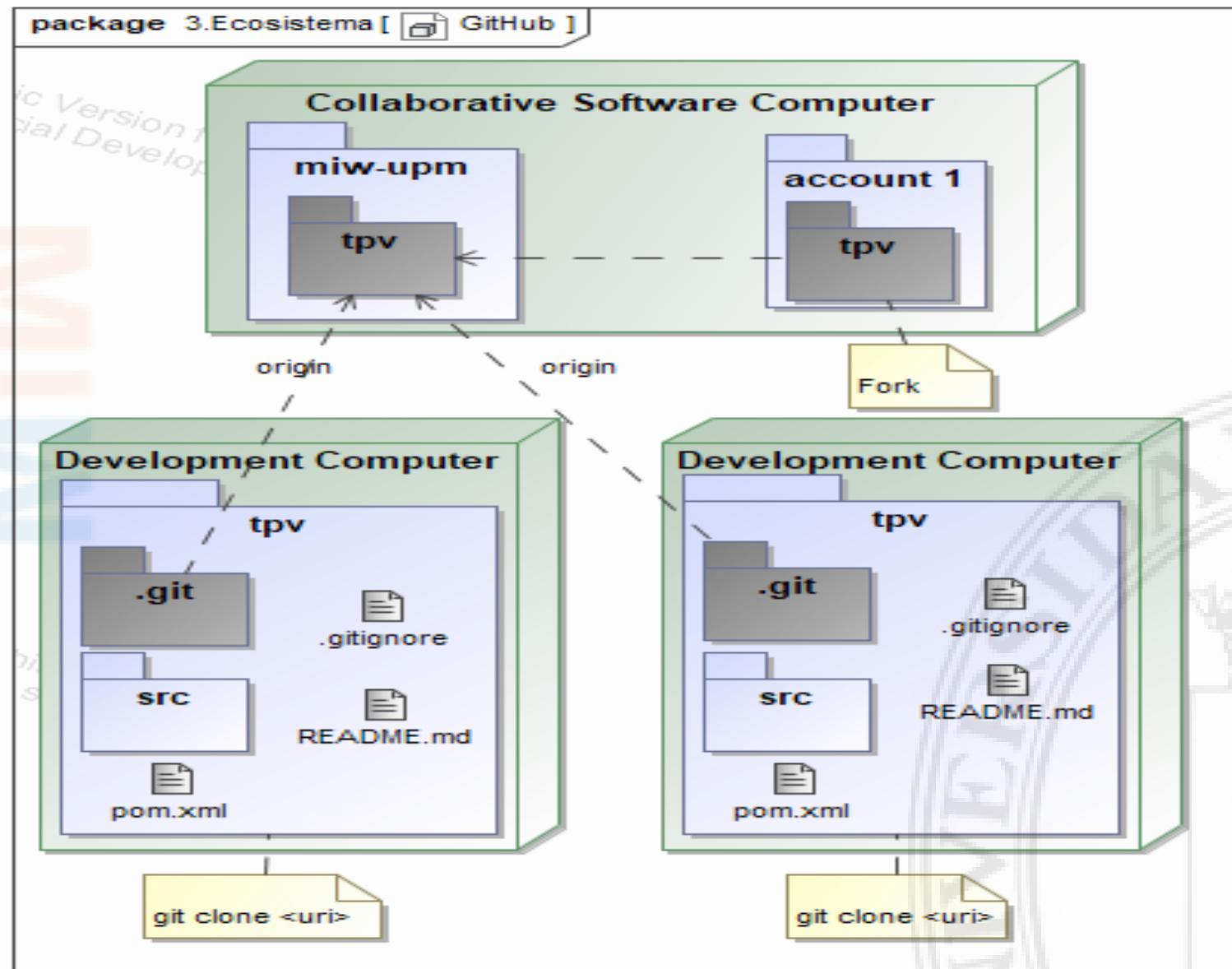
■ En la nube

- GitHub. <https://github.com/>. Sitio web que lleva integrada la forja. Para código abierto es gratuito, existe una versión de pago para proyectos privados
- Google Code. <https://code.google.com/intl/es/>
- SourceForge. <http://sourceforge.net/>
- Otras: Bitbucket (Atlassian), GitLab, BerliOS, Gforce, Savannah...

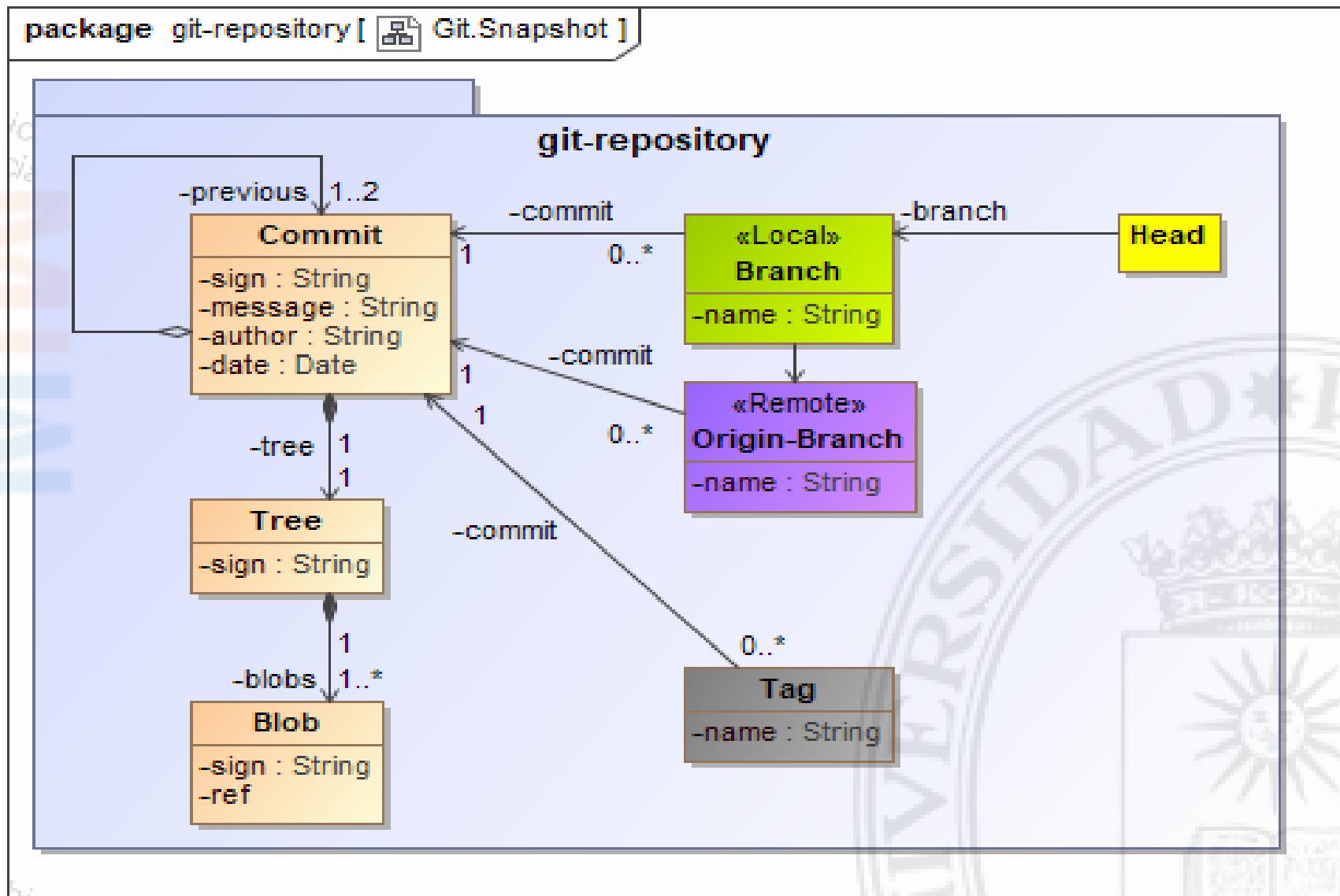
Sistema de control de versiones: Git

- <https://github.com/>
- Nace en 2005, tomando como experiencia el proyecto Bitkeeper (propietario).
 - En 2008 nace GitHub, Forja en Web con repositorio basado en Git.
 - En el 2018 Microsoft adquiere GitHub por 7.500 millones de dólares.
- Documentación: <https://git-scm.com/book>.
- Características
 - Control de versiones distribuido.
 - Muy fiable, casi imposible perder el proyecto.
 - Trabaja sin necesidad de conexión al remoto, muy rápido. Se podrá sincronizar con el remoto, **pero con asistencia...**
 - **Snapshot: instantánea (commits)**
- **Git-CLI:**
 - <https://git-scm.com/download/win>
- **GUI:**
 - <https://git-scm.com/downloads/guis/>
 - <https://www.gitkraken.com/>

Sistema de control de versiones: Git



Instantánea (snapshot)



Git: IntelliJ

The screenshot shows the IntelliJ IDEA interface with the 'Version Control' tool window open. The 'Log' tab is selected, displaying a list of recent git commits. A context menu is open over a commit from May 19, 2019, at 16:22, showing options like 'master', 'release-1.0', 'origin/master', 'origin/release-1.0', and 'release-1.0.2'. The commit list includes:

- reformat all code
- update Swagger page description #1 on develop
- version 1.0.2 & Heroku Procfile
- Merge bug #1 into release-1.0
- update swagger page description #1
- add mongodb embedded test scope
- version 1.0.1
- update sprig-boot version to 2.1.5
- version 1.1.0-SNAPSHOT
- prepare release: version 1.0.0 & prod profile
- heroku add Procfile
- heroku add deploy
- better-code-hub depth to 7
- ecosystem add sonarcloud & better-code-hub
- ecosystem add Travis-CI
- add docs & readme
- Initial TPV

On the right side, there's a tree view showing a folder named 'etsisi-tpv-spring' containing '.travis.yml' and 'README.md'. At the bottom, there's a status bar with tabs for Version Control, Terminal, Java Enterprise, Spring, Messages, Find, Run, and TODO.



Git-CLI

- Sitio web: <https://git-scm.com/>
- Ayuda
 - >`git help --all`
 - >`git <command> -h`
- >`git config --global user.name "..."`
- >`git config --global user.email "..."`
- >`git config credential.helper store`
 - Para guardar las credenciales, después de logearse
- >`git clone <uri>`
- >`git checkout -b <branch>`
- >`git add --all`
- >`git commit -m "mi mensaje de commit"`
- >`git merge --no-ff -m "<message>" <branch>`
-  **Demo in Action**
-  **En una carpeta de trabajo propia, clonar repositorio**
 - >`git clone https://github.com/miw-upm/iwvg-git-workflow`
-  **Branches exercises: Note 0**
 - Proyecto: <https://github.com/miw-upm/iwvg-git-workflow/projects>

Git-CLI

- Consultas
 - >*git config --list*
 - >*git status*
 - >*git remote -v*
 - *v:--verbose*
 - >*git log*
 - *q para salir*
 - >*git reflog*
 - *q para salir. Referencia de los commits*
- Crear repositorio con todos los ficheros
 - >*git init*
 - >*git add --all*
 - >*git commit -m "mi mensaje de commit inicial"*
- Commit (todos los ficheros con mensaje)
 - >*git add --all*
 - >*git commit -m "mensaje"*
 - >*git commit -am "add + commit"*
 - *No vale para ficheros nuevos*

Git

■ Ramas

- Mostrar ramas: `>git branch`
- Creación de rama y cambio de rama: `>git checkout -b <branch>`
- Cambio de rama: `>git checkout <branch>`
- Borrado de rama: `>git branch -d <branch>`
- Crear rama en el commit referenciado: `>git checkout -b <branch> <commit>`

■ Fusión de ramas:

- Fusión sobre la rama actual: `>git merge <branch>`
- Forzando commit (not fast forward): `>git merge --no-ff -m "<message>" <branch>`

■ Recuperación de situaciones anómalas

- Incorpora los nuevos cambios al último commit: `>git commit --amend --no-edit`
- Cambiar el mensaje del último commit: `>git commit --amend -m "new message"`
- Volver a un commit concreto:
 - `>git reset --hard HEAD`
 - `>git reset --hard <commit>`

Git

■ Repositorios remotos

- Consultar los repositorios remotos: `>git remote -v`
- Añadir repositorio remoto: `>git remote add origin <url repository>`
- Eliminar un repositorio remote: `>git remote rm origin`
- Subir una rama al remoto: `>git push origin <branch>`
- Subir todas las ramas: `>git push origin -all`
- Bajarse todas las ramas remotas: `>git fetch origin`
- Bajarse una rama: `>git fetch origin <branch>`
- Subir una rama de manera forzada **PELIGROSO!!!**:
 - `>git push origin <branch> --force`

■ Etiquetas

- Consulta de etiquetas: `>git tag`
- Crear etiqueta: `>git tag -a <etiqueta> -m "mensaje"`
- Borrado de etiqueta: `>git tag -d <etiqueta>`
- Subir etiqueta al remoto: `>git push origin <etiqueta>`

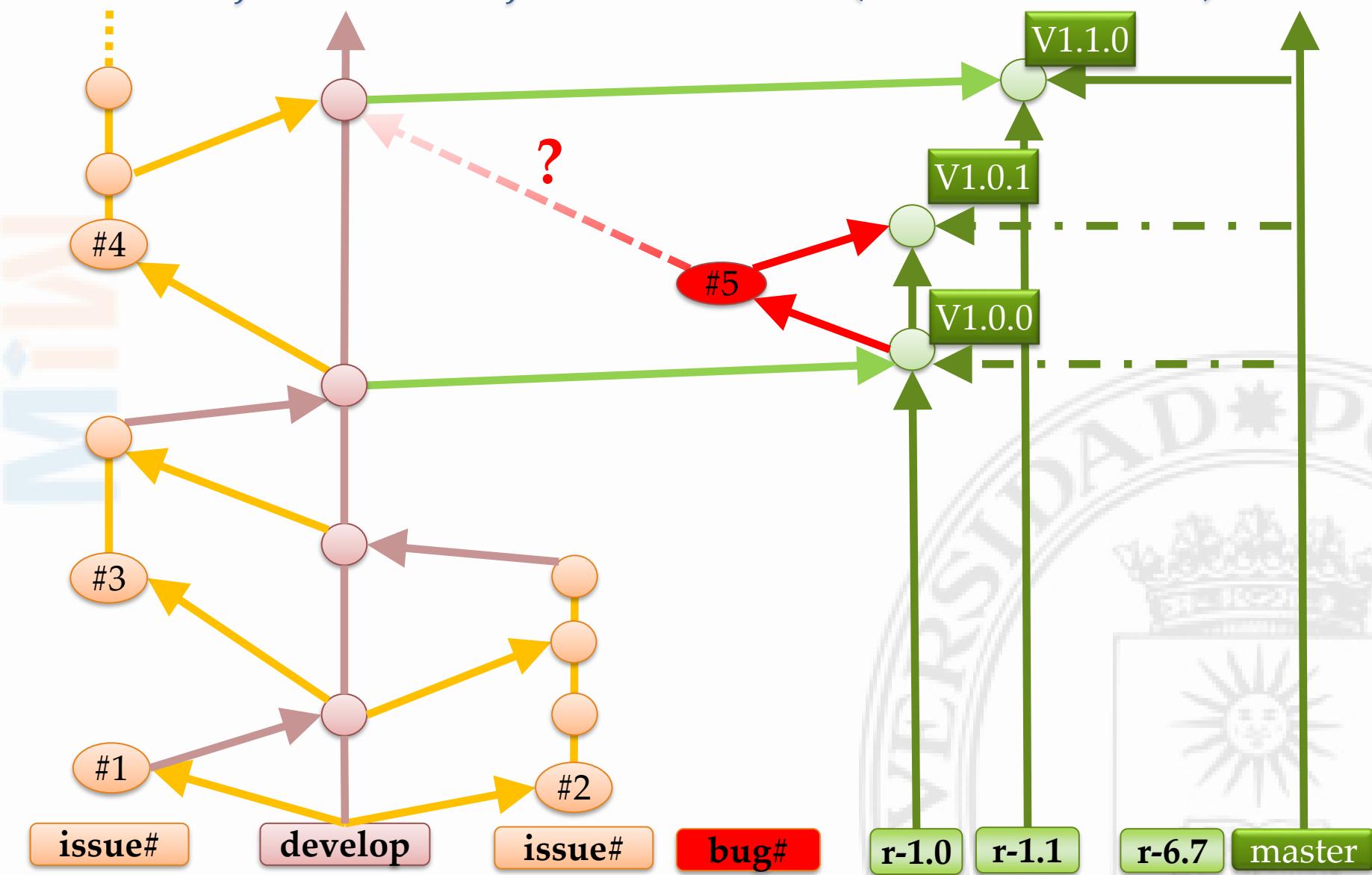
Flujo de Trabajo (workflow)

- El flujo de trabajo de un sistema de control de versiones indica cómo se relacionan los miembros del equipo para colaborar entre sí en el desarrollo del software colaborativo
 - Flujo de trabajo centralizado (*Centralized Workflow*). Todos comparten el código de un repositorio central, o en Git, todos comparten la única rama... Normalmente, se realiza una fusión al final del día, el último se come la fusión!!!
 - Flujo de trabajo ramificado (*Git Workflow*). Existen varias ramas con distintas funcionalidades dentro del equipo.
 - Flujo de trabajo con bifurcación (*Forking Workflow*). El usuario bifurca el proyecto realizando una copia completa del repositorio. Realiza las ampliaciones y realiza una petición de agregación (*pull request*). Se abre una discusión, y el dueño del repositorio decide la fusión.

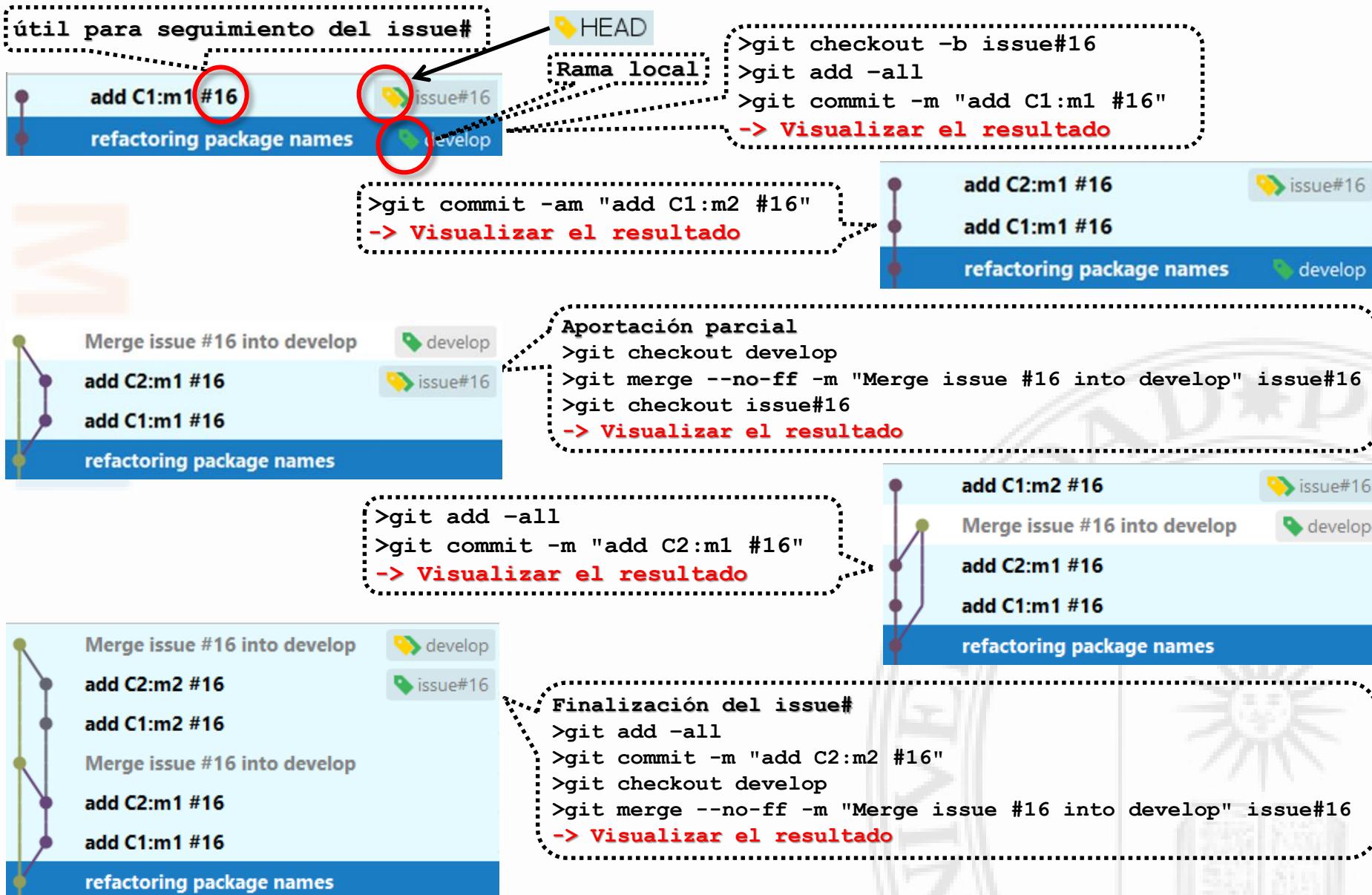
Flujo de Trabajo ramificado (Git Workflow)

- Rama master: apunta al último *commit* de producción, es decir, a la última versión del código liberado
- Rama develop: es la rama de desarrollo, siempre estable, código que cumple los requisitos de calidad y con todos los test superados
- Ramas issue#XX, feature#XX, topic#XX: parte de develop, se añaden las nuevas características y vuelve a *develop* cuando vuelve a ser estable
- Ramas release#XX: se utiliza para estabilizar un código para salir a producción. Una vez terminada, se fusiona con *master*
- Ramas bug#XX: se utilizan para corregir errores (*bugs*) en el código en producción

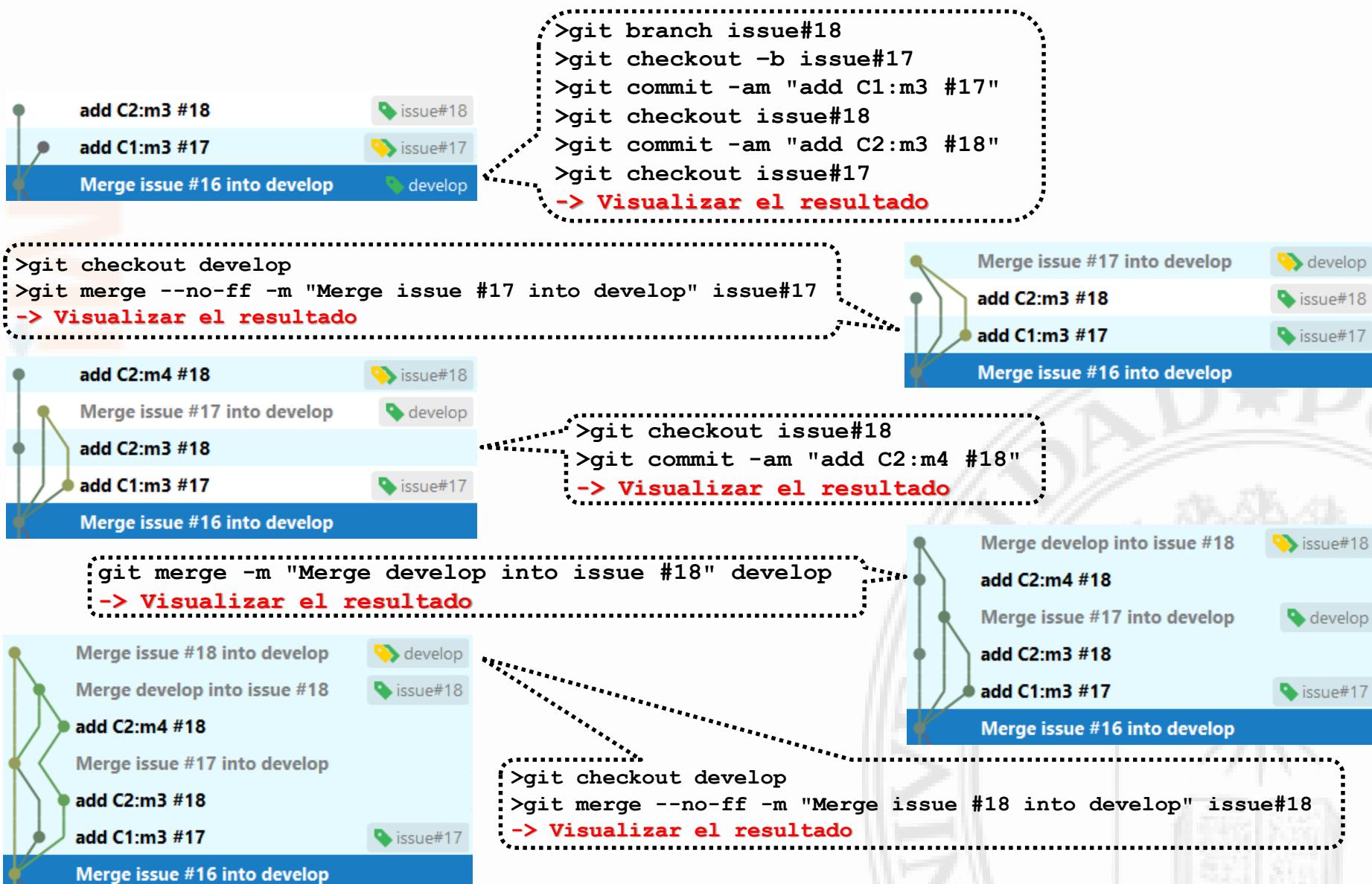
Flujo de Trabajo ramificado (Git Workflow)



Git (sin competencia, y por lo tanto, sin conflicto)



Git (con competencia pero sin conflicto)



Git (con conflicto)



```
>git merge -m "Merge develop into issue #20"
```

```
C:\JBB\work-spaces\intelliJ-idea\iwvg-forge>git merge -m "Merge develop into issue #20" develop
Auto-merging src/main/java/es/upm/miw/iwvg/ecosystem/git/C1.java
CONFLICT (content): Merge conflict in src/main/java/es/upm/miw/iwvg/ecosystem/git/C1.java
Automatic merge failed; fix conflicts and then commit the result.
```

```
public class C1 {
    public String m1() { return "C1:m1"; }

    public String m2() { return "C1:m2"; }

    public String m3() { return "C1:m3"; }

    public String m4() {
<<<<< HEAD
        return "C1:m4 of issue#20";
=====
        return "C1:m4";
>>>>> develop
    }
}
```

→ Editar C1 y dejar su contenido correcto

```
>git add -all
>git commit
```

Merge develop into issue #20

Conflicts:

```
src/main/java/es/upm/miw/iwvg/ecosystem/git/C1.java
#
# It looks like you may be committing a merge.
# If this is not correct, please remove the file
#     .git/MERGE_HEAD
# and try again.
```



→ Establecer el mensaje final

<ESC>:wq

→ Visualizar el resultado

 Git

-  Demo in Action
 - *Sin conflictos*
 - *Con conflictos*

-  Workflow exercises: Note 1
 - Proyecto: <https://github.com/miw-upm/iwvg-git-workflow/projects>

Git. Remotos



-> origin/develop y develop sincronizados

-> origin/develop NO sincronizado
-> Permitido Fast-forward

Merge issue #21 into develop

add C1:m5 #21

Merge issue #20 into develop

develop

issue#21

origin/develop



Merge issue #20 into develop

>git push origin -all
-> GitHub realiza un merge con Fast-forward
-> Visualizar el resultado

Merge issue #21 into develop add C1:m5 #21

origin & issue#21

>git checkout issue#21
>git commit -am "add C1:m7 #21"
-> Visualizar el resultado

add C1:m7 #21

Merge issue #21 into develop

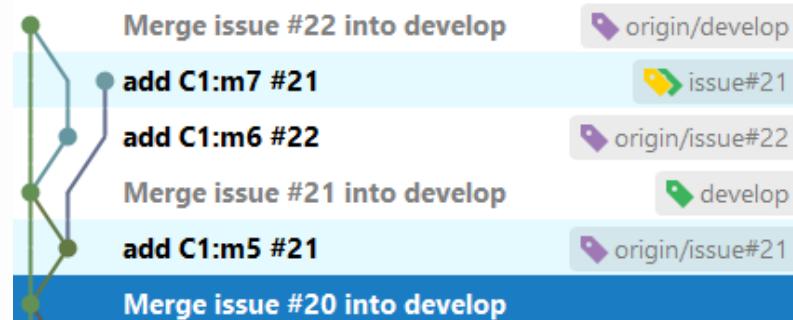
add C1:m5 #21

issue#21

origin & develop

origin/issue#21

Merge issue #20 into develop



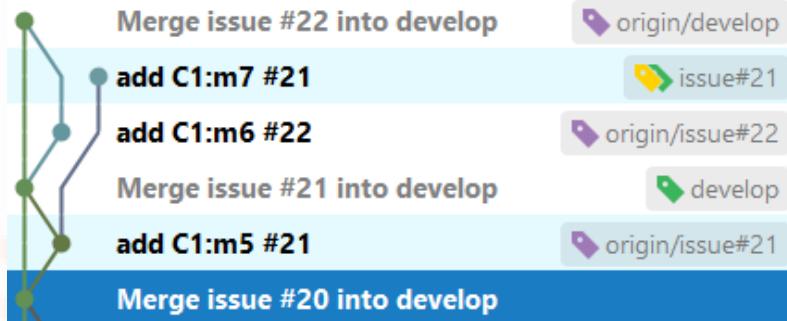
Merge issue #21 into develop

add C1:m5 #21

Merge issue #20 into develop

>git fetch origin
-> Visualizar el resultado
-> ??? siguientes pasos

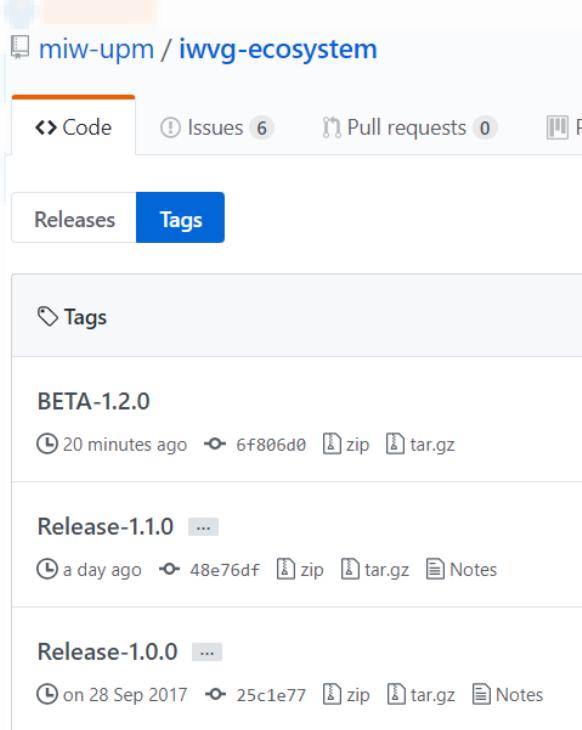
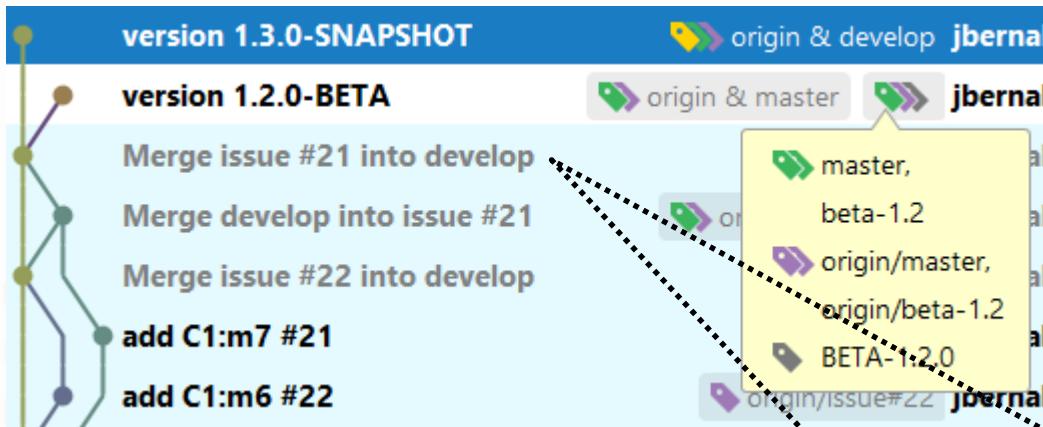


 Git

```
>git checkout develop
>git merge origin/develop

>git check issue#21
>git merge -m "Merge develop into issue #21" develop
>git checkout develop
>git merge --no-ff -m "Merge issue #21 into develop" issue#21
-> Visualizar el resultado
>git push origin --all
```



 Git

miw-upm / iwg-ecosystem

Code Issues 6 Pull requests 0 P

Releases Tags

Tags

BETA-1.2.0

20 minutes ago 6f806d0 zip tar.gz

Release-1.1.0 ...

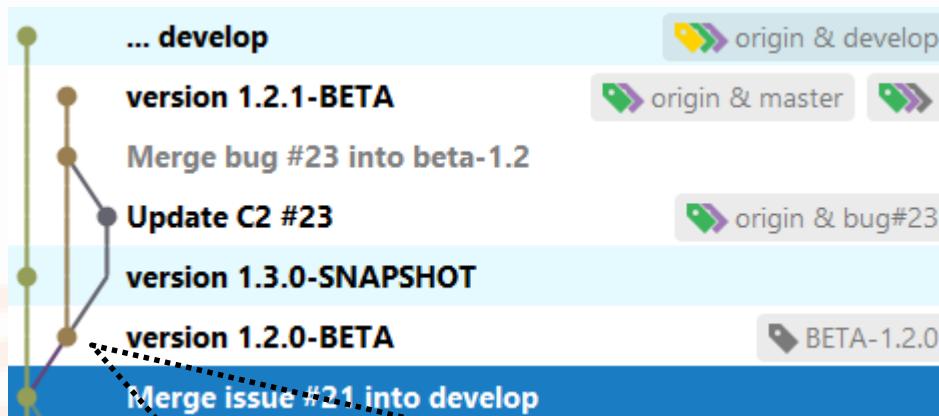
a day ago 48e76df zip tar.gz Notes

Release-1.0.0 ...

on 28 Sep 2017 25c1e77 zip tar.gz Notes

```
>git checkout -b beta-1.2
>git commit -am "version 1.2.0-BETA"
>git push origin beta-1.2
>git tag -a BETA-1.2.0 -m "BETA-1.2.0"
>git push origin BETA-1.2.0
>git checkout master
>git reset --hard 6f806d0
-> Visualizar el resultado para ver si es el esperado
>git push origin master --force

>git checkout develop
>git commit -am "version 1.3.0-SNAPSHOT"
>git push origin develop
-> Visualizar el resultado
```

 Git

```
>git checkout beta-1.2
>git checkout -b bug#23
>git commit -am "Update C2 #23"
>git checkout beta-1.2
>git merge --no-ff -m "Merge bug #23 into beta-1.2" bug#23
>git commit -am "version 1.2.1-BETA"
>git push origin beta-1.2
>git tag -a BETA-1.2.1 -m "BETA-1.2.1"
>git push origin BETA-1.2.1
>git checkout master
>git reset --hard d877efe
-> Visualizar el resultado para ver si es el esperado
>git push origin master --force

>git checkout develop
>git commit -am "... develop"
>git push origin develop
-> Visualizar el resultado
```

miw-upm / iwg-ecosystem

Code

Issues 7

Pull requests 0

Releases

Tags

Tags

BETA-1.2.1

5 minutes ago · d877efc · zip · tar.gz

BETA-1.2.0

38 minutes ago · 6f806d0 · zip · tar.gz

Release-1.1.0 ...

a day ago · 48e76df · zip · tar.gz · Notes

Release-1.0.0 ...

on 28 Sep 2017 · 25c1e77 · zip · tar.gz · Notes

 Git

-> Issue#24 ha realizado su mejora y la da por finalizada

- Errores de issue#24 (5 errors cometidos)



-> Issue#25, al intentar sincronizar origin/develop, github lo rechaza, realiza un >git fetch origin, y se encuentra con esta situación

- Errores de issue#25



-> Issue#26 ha realizado su mejora y la da por finalizada

- Errores de issue#26



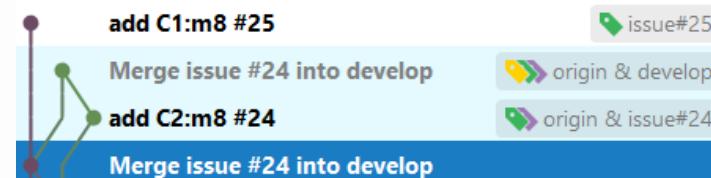
■ Errores

▫ Issue#24

- En los mensajes de los commits no añadir #24
- Realizar *Merge* con *develop* con *fast-forward* (*add C2:m5*)
- En el mensaje, juntar issue#24
- El *Merge* de *devolp into issue#24* es innecesario
- *origin/develop* no sincronizada

▫ Issue#25

- Erróneamente, se pensaba que su *develop* local estaba sincronizado
- Una solución, sería borrar el *develop* local y situarlo en *origin/develop*, y a partir de ahí... hacerlo bien:



▫ Issue#26

- Ha hecho *Merge* con *develop*, sin incorporar previamente los cambios de *develop* a su *issue*

 Git

-  **Workflow exercises: Note 2**
 - Proyecto: <https://github.com/miw-upm/iwvg-git-workflow/projects>

GitHub. Wiki

- Una es un sitio web cuyas páginas pueden ser editadas directamente desde el navegador, donde los usuarios crean, modifican o eliminan contenidos compartidos
- GitHub dispone de Wiki para la generación de documentación rápida y compartida a partir de lenguajes de marcado ligeros
- Dispone de varios lenguajes de marcado... Markdown

/miw-upm/iwvg-ecosystem/wiki/Markdown#lenguaje-markdown

es Marketplace Explore

miw-upm / iwvg-ecosystem

Code Issues 9 Pull requests 0 Projects 1 Wiki Insights Settings

Unwatch

Markdown

Máster en Ingeniería Web edited this page 2 days ago · 13 revisions

IWVG. Ecosistema. wiki!

Lenguaje Markdown

Varios

Línea en blanco para separar estructuras

```
\# carácter de escape para los caracteres especiales de Markdown
```

```
# carácter de escape para los caracteres especiales de Markdown
```

Hitos

- *Milestone*: Hito (título, descripción y fecha). Representa una de fecha de referencia, normalmente asociada a un lanzamiento o finalización de un módulo. Puede estar abierto-cerrado
- Se les puede asociar tickets
- Tiene marcado un nivel de finalización, obtenido por los tickets asociados que se encuentran cerrados

The screenshot shows a GitHub repository page for 'miw-upm / iwg-ecosystem'. The 'Milestones' tab is selected, displaying two milestones:

- Curso 2018-19**: Last updated 2 days ago. Progress bar: 66% complete (1 open, 2 closed). Actions: Edit, Reopen, Delete.
- Curso 2017-18**: Last updated 2 days ago. Progress bar: 100% complete (0 open, 5 closed). Actions: Edit, Reopen, Delete.

At the bottom of the screenshot, there is a footer with links: © 2019 GitHub, Inc., Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, About.

Issues

- Issues: disputa, asunto, tarea, error, ticket... (título, asignado a..., asociado a un hito..., comentario, etiquetas) (abierto-cerrado)
 - Se abre un foro de comentarios
- Etiquetas: facilitan su identificación y organización
 - Etiquetas de prioridad: priority: high, priority: medium...
 - Etiquetas de tipo: type: test, type: documentation...
 - Etiquetas de estimación: points: 8

The screenshot shows a GitHub repository interface for 'miw-upm / iwg-ecosystem'. The 'Issues' tab is selected, displaying 9 open issues. The issues are:

Issue #	Title	Type	Assignee
#26	Git 11 error	type: documentation	MIW
#25	Git 10 error	type: documentation	MIW
#24	Git 9 error	type: documentation	MIW
#14	Story 2	points: 8, priority: low, type: test	MIW
#13	Story 3	points: 3, priority: medium, type: enhancement	Scrum: sprint 1, MIW
#12	Story 4	points: 2, priority: medium, type: bug	Scrum: sprint 1, MIW

Organización de un Issue

- Sólo tiene una breve descripción. En la wiki se establece toda la documentación

Git 11 #26

The screenshot shows a GitHub issue page for issue #26. The page includes the following sections:

- Issue Details:** Shows the title "Ejercicios básicos de GIT. Situaciones anómalas (@miw-upm)", the owner "miw-upm", assignees "miw-upm", labels "error" and "type: documentation", and projects and milestones.
- Comments:** A single comment from "miw-upm" linking to a detailed description in a wiki page.
- Commits:** Two commits by "miw-upm" referencing the issue, and a merge commit "Merge issue #26 into develop".
- Notifications:** A notification bar indicating the user is receiving notifications for this repository.
- Bottom Navigation:** Buttons for "Write" and "Preview" and a rich text editor toolbar.

Callouts provide the following annotations:

- Título General, el identificador es único lo crea Github (#26)** (General title, the identifier is unique created by Github (#26))
- @cuenta se referencia cuentas de usuarios** (@account refers to user accounts)
- Se referencia en la wiki una descripción detallada** (Referenced in the wiki a detailed description)
- Indica al proyecto que pertenece** (Indicates which project it belongs to)
- Hito asociado** (Associated commit)
- Con #26 en el mensaje, el commit se asocia al issue#26** (With #26 in the message, the commit is associated with issue #26)
- Commit asociado** (Associated commit)
- Test OK** (Test OK)

Proyectos: ≈ Scrum

Estimación **Prioridad**

Sprints: issues, start, due, team velocity...

The screenshot shows a Git-based project management interface. At the top, there are three boxes: 'Estimación' (Estimation), 'Prioridad' (Priority), and 'Sprints: issues, start, due, team velocity...'. Below this is a navigation bar with links for 'Code', 'Issues 9', 'Pull requests 0', 'Projects 1', 'Wiki', 'Insights', and 'Settings'. A progress bar indicates 'IWVG Ecosistema Software 2019/20' is 25% complete. The main area features a Kanban board with four columns: 'Story' (2 items), 'Backlog' (2 items), 'In progress' (1 item), and 'Done' (1 item). Each card contains details like title, author, points, priority, type, and a 'MW' icon. Above the board, a 'Labels' and 'Milestones' section shows '3 Open' and '2 Closed' issues. A detailed view of a sprint is shown on the right, titled 'Scrum: sprint 1' and 'Scrum: sprint 2', with information about due dates, last update times, start dates, team velocity, and a progress bar.

Template: Automated kanban, adaptado a Scrum

Cuando se mueve de columna, aparece en el historial del issue

Se asocian los issues a un Hito, que limita el tiempo del Sprint

Cuando se cierra un issue se mueve automáticamente a la columna "Done"

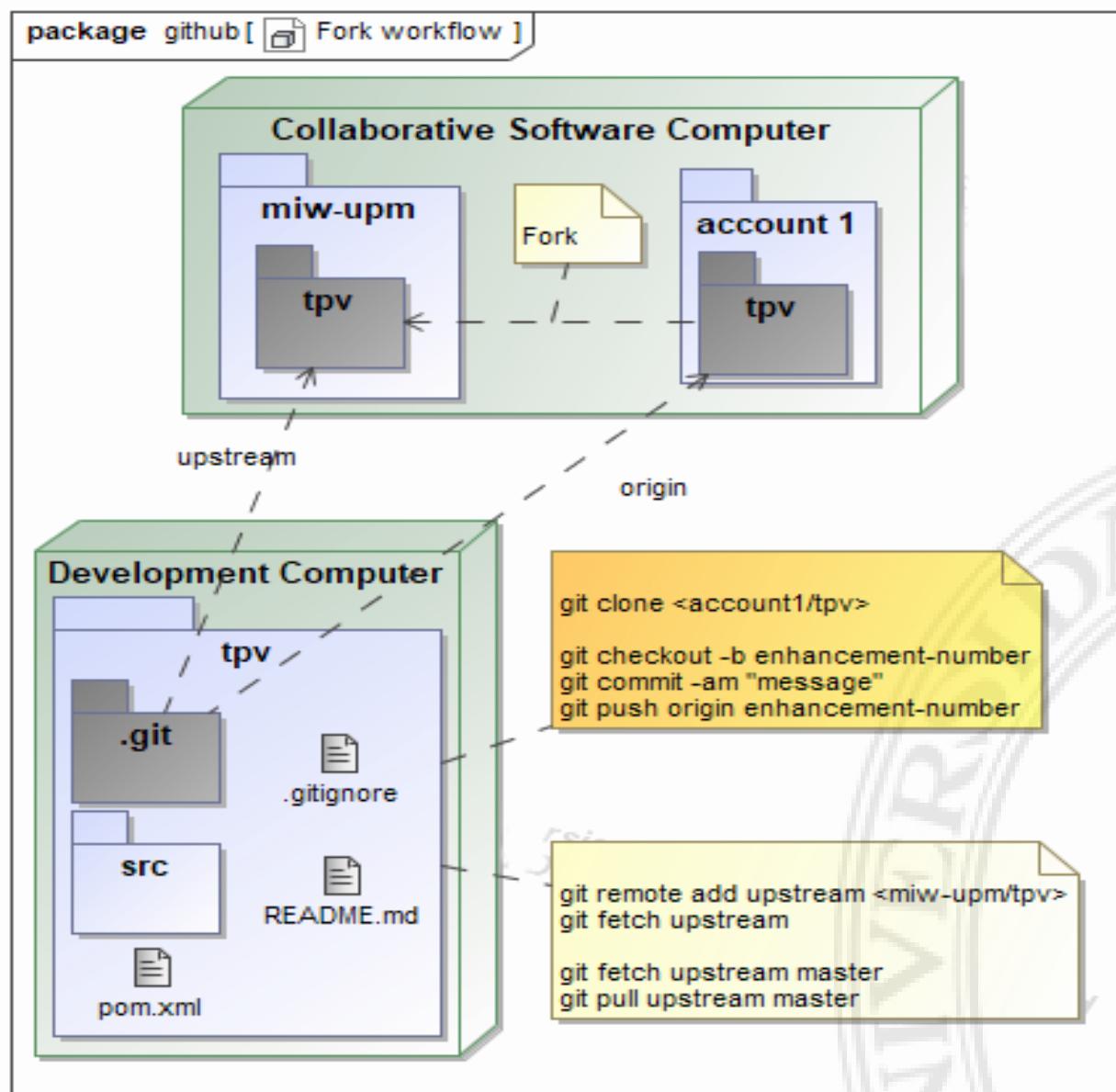
✍ Gestión con GitHub: ≈ Scrum

1. Crear las historias (*issues*) en el proyecto. Asociarle *prioridad, estimación, tipo*...
2. Mover las historias a la columna *backlog*, son las que desarrollan en el próximo *sprint*, asociarles el *sprint* (hito).
3. Cuando alguien inicia una historia, tarea (*issue*), se lo asigna y lo mueve a la columna *In progress*.
4. Se crea la rama ‘*issue#xx*’ y se programa la historia, tarea... un *commit* debe ser reflejado en el historial del *issue*, se añade en el mensaje ‘ #xx’
5. Cuando la historia se termina, se fusiona con *develop*:
 - git merge -no-ff -m “Merge branch issue #xx into develop” issue#xx
6. Se cierra la historia (*issue#xx*) y se abandona la rama

 Git

-  Workflow exercises with Scrum: Note 3
 - Proyecto: <https://github.com/miw-upm/iwvg-git-workflow/projects>

Flujo de Trabajo con bifurcación (Forking Workflow)



Flujo de Trabajo con bifurcación (Forking Workflow)

The screenshot illustrates the Forking Workflow across two main interfaces: a local Git commit history and a GitHub repository page.

Local Git Commit History: On the left, a terminal window shows a local repository named "iwvg-fork1 [iwvg-fork-workflow]". It displays four commits: "add C2", "add C1", "add core", and "Initial". The "add core" commit is highlighted. To the right of the commit details, two status indicators are shown: "origin & enhancement-number-one" (green arrow icon) and "origin & master upstream/master" (green arrow icon and pink diamond icon).

GitHub Repository Page: On the right, the GitHub page for the forked repository "jesusBernalBermudez / iwvg-fork-workflow" is displayed. The repository was forked from "miw-upm/iwvg-fork-workflow". The page includes navigation tabs for "Code", "Pull requests 0", "Projects 0", "Wiki", "Insights", and "Settings".

Annotations:

- A callout box labeled "Nombres de ramas descriptivos" points to the descriptive branch names in the local commit history (e.g., "enhancement-number-one").
- A callout box labeled "git fetch upstream / git pull upstream ...con frecuencia" points to the GitHub repository page, specifically to the "pull requests" section where a new pull request is visible.
- A large callout box at the bottom labeled "Con login del fork, al subir la rama al remoto, aparece la petición de pull request" points to the "Compare & pull request" button in the GitHub interface.

Flujo de Trabajo con bifurcación (Forking Workflow)

Enhancement number one #1

[Open](#) jesusBernalBermu... wants to merge 2 commits into [miw-upm:master](#) from [jesusBernalBermudez:enhancement-number-one](#)

Conversation 1 Commits 2 Checks 0 Files changed 2

jesusBernalBermu... commented 4 minutes ago
Enhancement number one Description

miw-upm added some commits 19 minutes ago
miw add C1
miw add C2

miw-upm requested changes just now
miw-upm left a comment

Add more methods

Add more commits by pushing to the [enhancement-number-one](#) branch on [jesusBernalBermudez/iwvg-fork-workflow](#).

Changes requested
1 review requesting changes [Learn more](#).
[Unsubscribe](#) [Notifications](#)

miw-upm requested changes [Approve changes](#) [Dismiss review](#) [Re-request review](#)

Continuous integration has not been set up
Several apps are available to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

[Merge pull request](#) You can also open this in GitHub Desktop or view command line instructions.

- Comment
Submit general feedback without explicit approval.
- Approve
Submit feedback and approve merging these changes.
- Request changes
Submit feedback that must be addressed before merging.

Login del oficial
Se puede configurar
y administrar la
petición

Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge
The 3 commits from this branch will be combined into one commit in the base branch.

Rebase and merge
The 3 commits from this branch will be rebased and added to the base branch.

Flujo de Trabajo con bifurcación (Forking Workflow)

Enhancement number one #1

Merged miw-upm merged 3 commits into miw-upm:master from jesusBernalBermudez:enhancement-number-one 10 minutes ago

Conversation 1 Commits 3 Checks 0 Files changed 2 +24 -0

jesusBernalBermudez commented 26 minutes ago

Enhancement number one Description

Contributor + ...

Reviewers miw-upm

miw-upm added some commits 41 minutes ago

0c4b1d6
0ac3bb0

MIW add C1
MIW add C2

miw-upm requested changes 22 minutes ago

View changes

miw-upm left a comment

Owner + ...

Add more methods

MIW update C1&C2

b764964

miw-upm merged commit 08096c2 into miw-upm:master 10 minutes ago

Revert

Pull request successfully merged and closed

You're all set — the jesusBernalBermudez:enhancement-number-one branch can be safely deleted.

If you wish, you can also delete your fork of miw-upm/iwvg-fork-workflow.

Delete branch

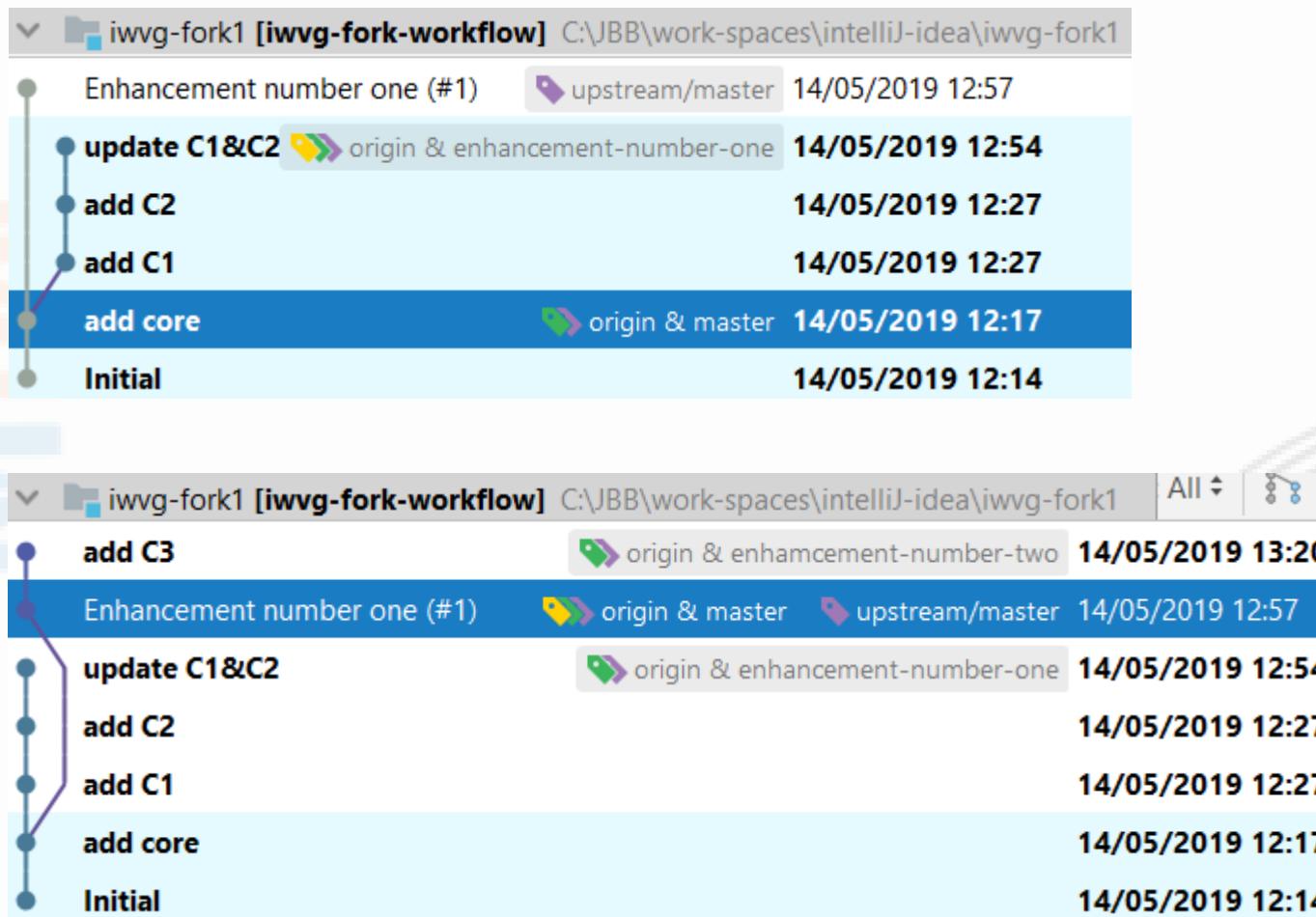
Notifications

Unsubscribe

2 participants

MIW

Flujo de Trabajo con bifurcación (Forking Workflow)



 Ejercicio

- Los alumnos deben hacer fork al repositorio *iwvg-fork-workflow*
- Se debe crear una clase con un método y realizar un *pull request*
- Observar como evoluciona el código

Integración Continua (CI)

- La integración continua es una práctica de desarrollo de software, propuesto inicialmente por Martin Fowler, la cuál los miembros de un equipo integran su trabajo frecuentemente (diariamente) y se revisa automáticamente.
- Filosofía muy relacionada con las metodologías ágiles y la programación extrema (XP).
- Cada integración se verifica mediante una herramienta de construcción para detectar los errores de integración tan pronto como sea posible.
- Para ello se pueden utilizar un software especializado en automatizar tareas y que estas se ejecuten de forma automática. Las tareas a realizar pueden ser:
 - Compilación de los componentes.
 - Obtener métricas de calidad de código.
 - Ejecución de pruebas unitarias.
 - Ejecución de pruebas de integración, de aceptación.
- Buenos principios.
 - Contribuir a menudo.
 - No contribuir con código roto, todos los *tests* deben pasar.
 - Soluciona los *builds* rotos inmediatamente.
 - Escribe *tests* automáticos.

Integración Continua (CI). Procedimiento

- El equipo modifica el código, y una vez testeado, se realiza un *commit + push* al repositorio.
- La herramienta de CI monitoriza el repositorio y se dispara cada vez que detecta un cambio.
- CI ejecuta la construcción del todo el código fuente, aplicando los testeos de control de calidad, pruebas unitarias, pruebas de integración...
- CI envía correos de los resultados de la integración del nuevo código.
- Incluir *Badges*.

Estado del código

build passing



maintainability

A

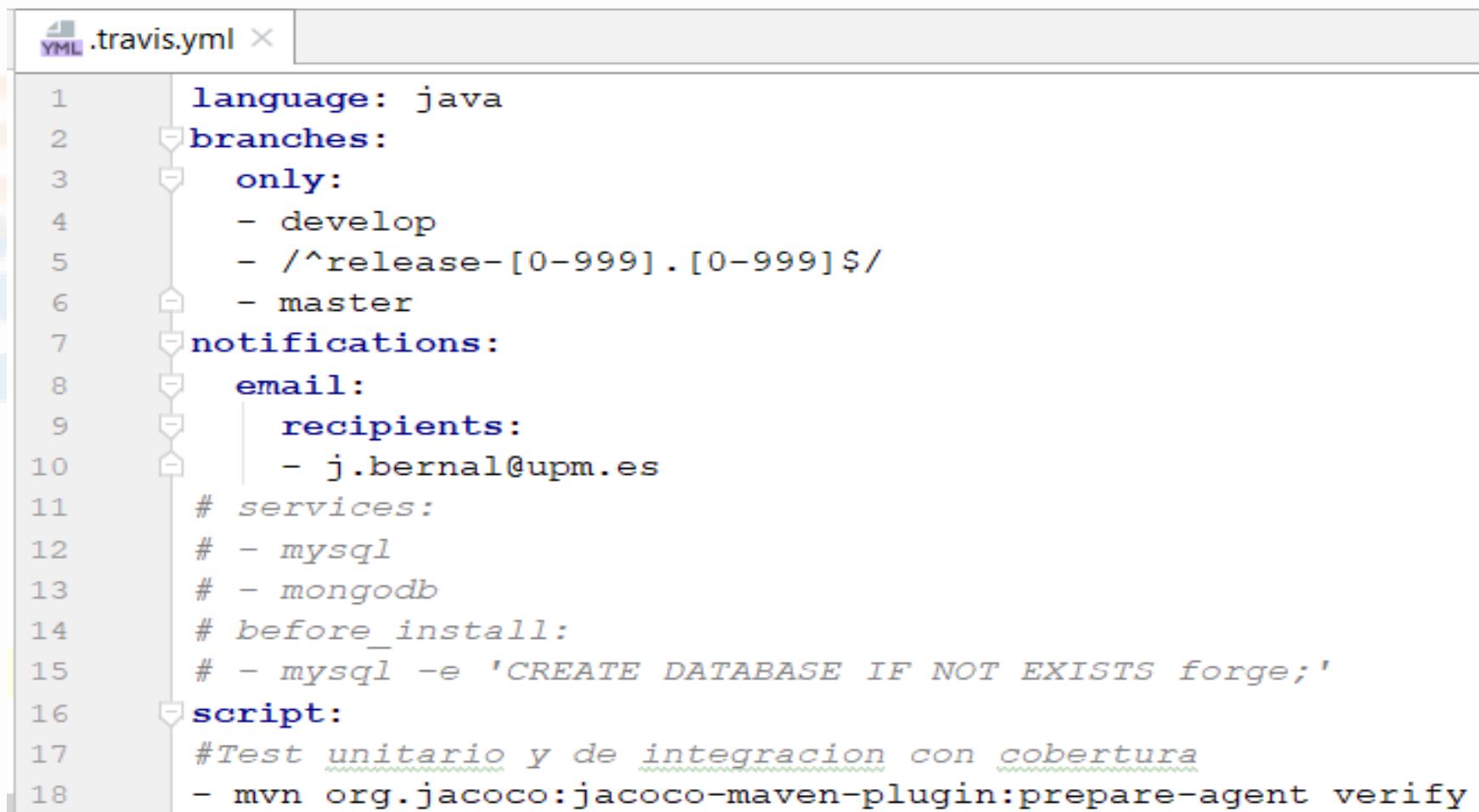
Better Code 10 / 10

Integración Continua. Herramientas

- Travis CI
 - [https://travis-ci.org/.](https://travis-ci.org/)
 - Documentación: [http://about.travis-ci.org/docs/.](http://about.travis-ci.org/docs/)
 - Travis CI es un sistema de integración continua gratuito en la nube para la comunidad OpenSource. Está integrado con GitHub y ofrece soporte para: Java, Groovy, Haskell, Node.js, PHP, Python, Ruby...
- Jenkins
 - [http://jenkins-ci.org/.](http://jenkins-ci.org/)
 - Jenkins es un software de integración continua de código abierto escrito en Java, evolución de Hudson.
 - Jenkins tiene soporte para sistemas de control de versiones, algunas como SVN, CVS, Git y corre en un servidor de aplicaciones como por ejemplo Tomcat o Jboss permitiendo la ejecución de proyectos Ant, Maven...

Travis CI. Configuración

- Travis-CI se configura con el fichero: [*.travis.yml*](#) en la raíz del proyecto.



```
YML .travis.yml ×

1   language: java
2   branches:
3     only:
4       - develop
5       - ^release-[0-999].[0-999]$/ 
6       - master
7   notifications:
8     email:
9       recipients:
10      - j.bernal@upm.es
11   # services:
12   # - mysql
13   # - mongodb
14   # before_install:
15   # - mysql -e 'CREATE DATABASE IF NOT EXISTS forge;'
16   script:
17     #Test unitario y de integracion con cobertura
18     - mvn org.jacoco:jacoco-maven-plugin:prepare-agent verify
```

✍ GitHub y Travis CI

- Para empezar a Travis CI, acceder a través de GitHub OAuth. Ir a Travis CI y seguir el vínculo de iniciar sesión, situado en la parte superior: *Sign in GitHub*.
- Activar en GitHub el Servicio Hook#, se realiza desde la Web de Travis-CI. En la parte superior derecha, sobre nuestra cuenta, elegir la opción *Accounts*, activar el interruptor para cada repositorio que desea conectar en Travis CI.
- A partir de ahora, el repositorio estará monitorizado por Travis CI. Cada vez que se realice un commit, ejecuta uno de los siguientes constructores en sus servidores, esta operación puede ir despacio (varios minutos).

Travis CI

Builds - miw-upm/iwvg-ecosyste x +

https://travis-ci.org/miw-upm/iwvg-ecosystem/builds En pausa

Travis CI Dashboard Changelog Documentation Help MIW

miw-upm / iwvg-ecosystem

build passing

Current Branches Build History Pull Requests More options

Branch	Pull Request	Time	Actions
✓ develop	Merge issue #26 into develop	1 min 53 sec	↻
✓ develop	Máster en Ingeniería Web	d380985 ↗	about 4 hours ago
✓ develop	Merge issue #25 into develop	2 min 11 sec	↻
✓ develop	Máster en Ingeniería Web	3c328d4 ↗	about 4 hours ago
✓ develop	Merge issue #24 into develop	1 min 47 sec	↻
✓ develop	Máster en Ingeniería Web	5cc28cc ↗	about 4 hours ago

Herramientas de Análisis Estático

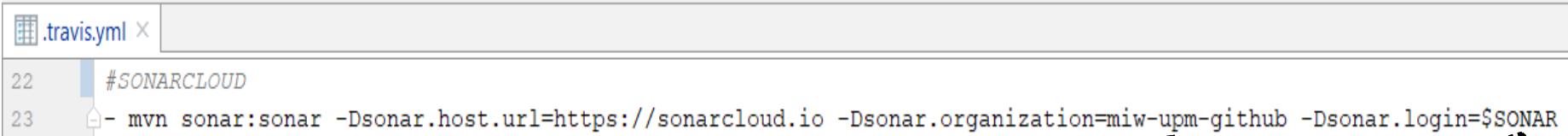
- Son herramientas que nos ayudan a detectar problemas del código fuente, buscando errores de seguridad, código duplicado, problemas potenciales por mala práctica...
- Se pueden conectar directamente al repositorio y realizar el análisis cada vez que este cambie.
- También se pueden integrar en el proceso de la Integración Continua.
- Existe una gran variedad en el mercado... pero se han elegido dos:
 - *Sonarcloud*. Se ofrece en la nube *Sonarqube* (*GNU LGPL 3*), y se integra GitHub, Maven...
 - *BetterCodeHub*. Se ofrece en la nube (*Creative Commons Attribution 3.0 License*), y se integra con GitHub...

Sonar

- Sonarqube es una plataforma abierta para gestionar la calidad del código.
 - <http://www.sonarqube.org/>.
- Se ofrece servicio en la nube
 - <https://sonarcloud.io/>.
- Abarca las siguientes características:
 - Estándares de Codificación.
 - Bugs y errores potenciales .
 - Duplicaciones de código.
 - Pruebas Unitarias.
 - Cobertura de pruebas.
 - Complejidad ciclomática.
 - Control de comentarios.
- Ofrece soporte para más de 20 lenguajes: Java...

 Sonarcloud

- Servicio on-line de *sonarqube*.
- Para crear un nuevo proyecto.
 - >>>*Help > Tutorials > Analyze a new Project.*
- Para generar una clave de acceso:
 - <https://sonarcloud.io/account/security/>.
- Ejecutar el análisis desde Travis-CI.
 - Seguridad: la clave de acceso es mejor definirla como variable en *Travis-CI*.
 - En *Travis-CI*, en el Proyecto, ir a *Settings (More options)*, y en el apartado *Environment Variables*, definir una variable con la clave, por ejemplo, **SONAR**.
 - Para referenciarla: **\$SONAR**.
 - Definir en el fichero *.travis.yml*:



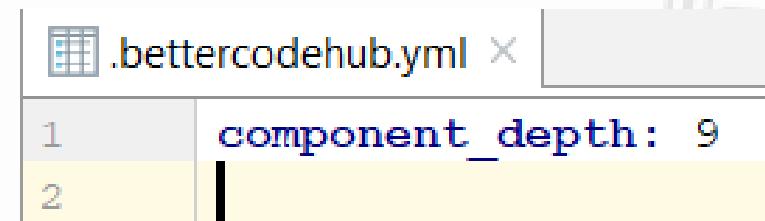
```
.travis.yml x
22 #SONARCLOUD
23 - mvn sonar:sonar -Dsonar.host.url=https://sonarcloud.io -Dsonar.organization=miw-upm-github -Dsonar.login=$SONAR
```

Nombre de la organización

Variable de entorno de Travis-CI

Better Code Hub

- Better Code Hub verifica que su código cumpla con 10 pautas de ingeniería de software de referencia.
- Soporta Java, Kotlin, TypeScript, Python...
- Sigue los principios de:
 - Building Maintainable Software, Java Edition. Ten Guidelines for Future-Proof Code.
 - Building Software Teams: Ten Best Practices for Effective Software Development.
- Se conecta con GitHub, y tiene una configuración por defecto, pero se puede configurar con un fichero *.bettercodehub.yml* en la raíz del proyecto.



A screenshot of a code editor showing a file named ".bettercodehub.yml". The file contains the following content:

```
component_depth: 9
```

Better Code Hub

- Write Short Units of Code (*method*)
 - Limit the length of code units to 15 lines of code. Do this by splitting long units into multiple smaller units until each unit has at most 15 lines of code.
- Write Simple Units of Code (*method*).
 - Limit the number of branch points per unit to 4. Do this by splitting complex units into simpler ones and avoiding complex units altogether.
- Write Code Once.
 - Do not copy code. Do this by writing reusable, generic code and/or calling existing methods instead.
- Keep Unit Interfaces Small (*parameters*).
 - Limit the number of parameters per unit to at most 4. Do this by extracting parameters into objects.
- Separate Concerns in Modules (*class*).
 - Avoid large modules in order to achieve loose coupling between them. Do this by assigning responsibilities to separate modules and hiding implementation details behind interfaces.

Better Code Hub

- Couple Architecture Components Loosely (*class-package*)
 - Achieve loose coupling between top-level components. Do this by minimizing the relative amount of code within modules that is exposed to (i.e., can receive calls from) modules in other components.
- Keep Architecture Components Balanced (*class-package*).
 - Balance the number and relative size of top-level components in your code. Do this by organizing source code in a way that the number of components is close to 9 (i.e., between 6 and 12) and that the components are of approximately equal size.
- Keep Your Codebase Small (*project < 175.000 lines*).
 - Keep your codebase as small as feasible. Do this by avoiding codebase growth and actively reducing system size.
- Automate Tests.
 - Automate tests for your codebase (>80%). Do this by writing automated tests using a test framework.
- Write Clean Code.
 - Write clean code. Do this by not leaving code smells behind after development work.

Better Code Hub

Compliance 9 of 10

miw-upm/apaw-microservice-themes-user

Last analysis: 3 days ago
Previous analysis: 3 days ago

Branch: develop (default)

Write Short Units of Code ✓

Write Simple Units of Code ✓

Write Code Once ✗

Refactoring candidates

- Duplicate
 - 12 lines occurring 2 times in 2 files: Address.java, UserCreationDto.java
 - 8 lines occurring 2 times in 2 files: User.java, UserCreationDto.java

Show snoozed Lines of Code 12 8

non-duplicated code duplicated code

Keep Unit Interfaces Small ✓

Separate Concerns in Modules ✓

Couple Architecture Components Loosely ✓

Keep Architecture Components Balanced ↗✓

Keep Your Codebase Small ✓

Automate Tests ✓

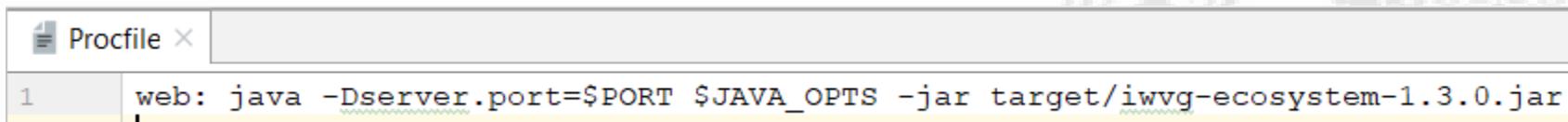
Write Clean Code ✓

HEROKU

- Heroku (www.heroku.com) es una plataforma *PaaS* (*Plataforma como Servicio*) comercializada por *Salesforce*.
- Es posible desarrollar prácticamente con cualquier lenguaje de programación: *Java*, *Ruby*, *PHP*, *NodeJS*, etc.
- Las aplicaciones Heroku.
 - Están preparadas para poder tener una muy alta escalabilidad. Esto se consigue utilizando los procesos *dyno*, que son instancias independientes que *Heroku* puede levantar y bajar en cualquier momento (en función de la carga de trabajo).
 - Cada instancia contiene una copia del código fuente de la aplicación.
 - Es la plataforma Heroku la que se ocupa de colocar un servidor web delante de los *dyno*, de hacer las comunicaciones SSL, etc.
 - Los procesos *dyno* pueden tener una vida muy efímera (por ejemplo, para atender tan solo una petición) con lo cual no es recomendable almacenar información en memoria, ni tampoco escribir ningún fichero en disco.
- Cliente *Heroku CLI* (<https://devcenter.heroku.com/articles/heroku-cli>).
 - **>heroku -versión**
 - **>heroku login**
 - **>heroku logs --app=<proyecto> -n 100** //logs pasados, los 100 últimos
 - **>heroku logs --tail -app=<proyecto>** //logs en tiempo real

HEROKU (despliegue)

- En la Web de Heroku: <https://dashboard.heroku.com>
 1. Crear una aplicación en Heroku (Botón **>>New**).
 - *name: proyecto.*
 - *zone: Europe.*
 2. Configurar Heroku.
 - **>>Personal>\${proyecto}>Settings**
 - Es automático. Se puede elegir dependencias (**>>Add buildpack**): **heroku/java**
 3. Obtener el API-key: **>>Personal>Account settings**
- En el proyecto, añadir el comando *deploy* en el fichero *.travis.yml*
- En la web de TRAVIS-CI, crear una variable de entorno (*HEROKU*) con la *ApiKey*.
- Crear el fichero *Procfile*, en la raíz del proyecto.



```
Procfile x
1 web: java -Dserver.port=$PORT $JAVA_OPTS -jar target/iwvg-ecosystem-1.3.0.jar
```

- Subir a **Git-hub** la rama *master*, ello dispara la integración continua con *Travis-CI*, y si no existen errores, se despliega automáticamente en *Heroku*, la ruta del despliegue será:
 - <https://<proyecto>.herokuapp.com/>.

deploy:

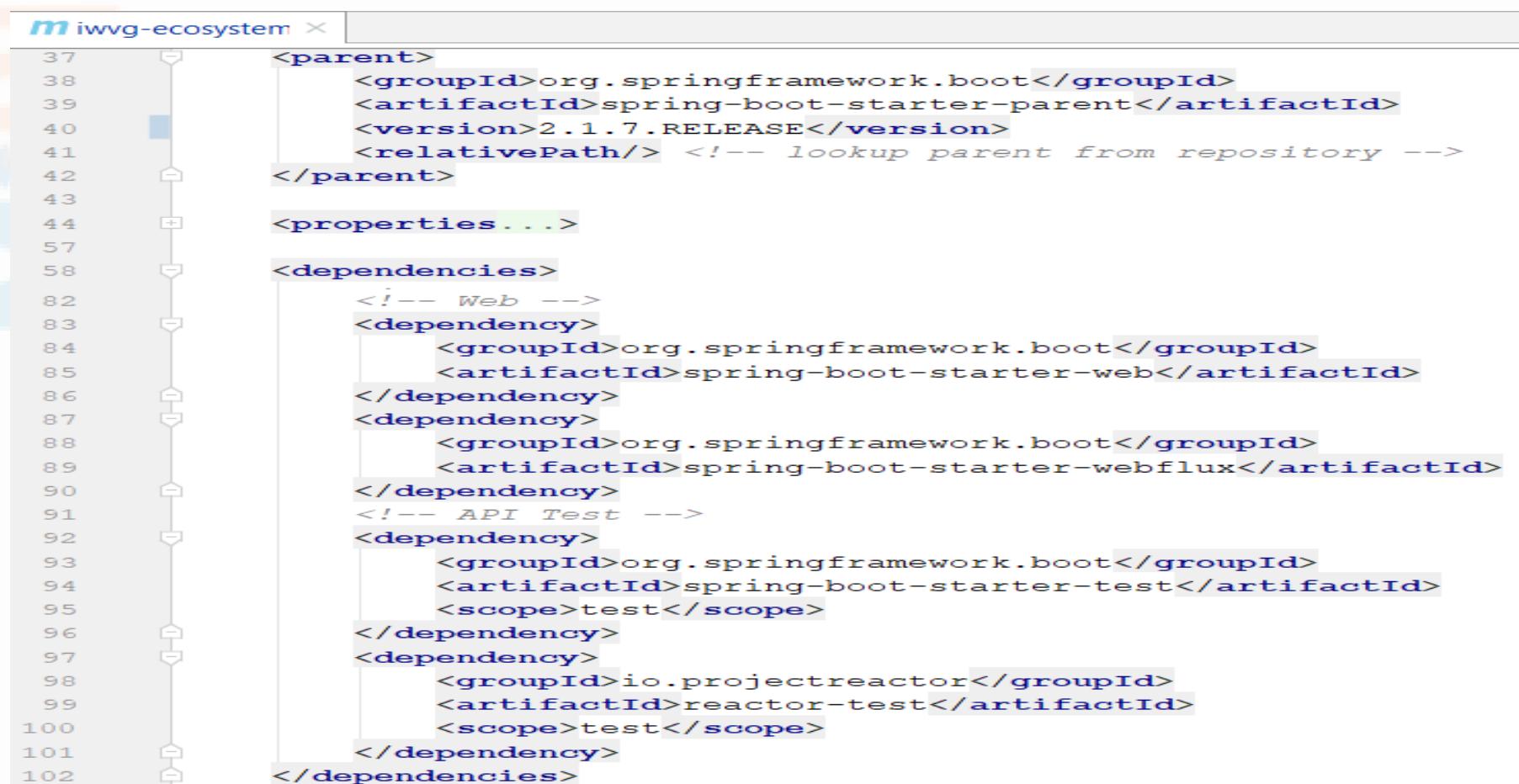
provider: heroku

api_key:

secure: \$HEROKU

Spring Boot: introducción

- Spring es un framework ligero de código abierto para facilitar el desarrollo de Aplicaciones Empresariales modernas mediante Java: <https://spring.io/>
- Dependencias: POM



The screenshot shows a code editor displaying a Maven Project Object Model (POM) file. The file is titled 'iwvg-ecosystem' and contains XML code for defining dependencies and properties. The code is color-coded for syntax highlighting, with blue for tags and green for attributes. The editor interface includes a left sidebar with navigation icons and a top bar with the project name.

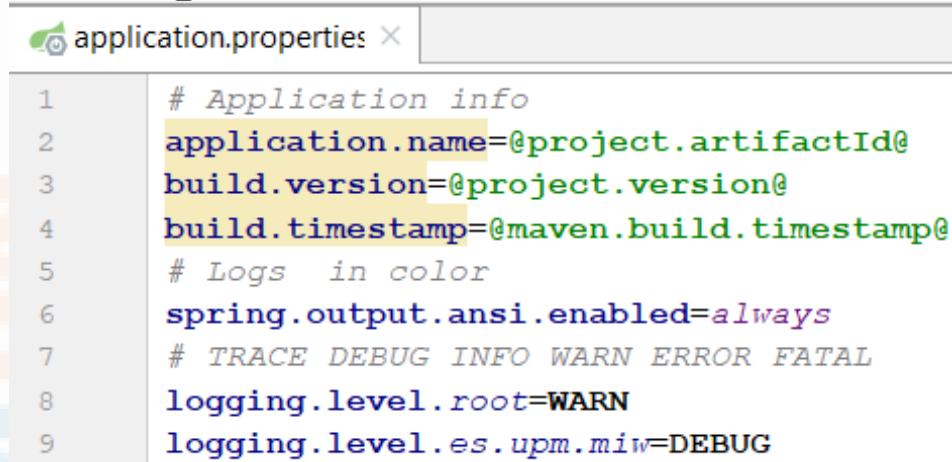
```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.1.7.RELEASE</version>
    <relativePath/> <!-- lookup parent from repository -->
</parent>

<properties...>

<dependencies>
    <!-- Web -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-webflux</artifactId>
    </dependency>
    <!-- API Test -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>io.projectreactor</groupId>
        <artifactId>reactor-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Spring Boot: introducción

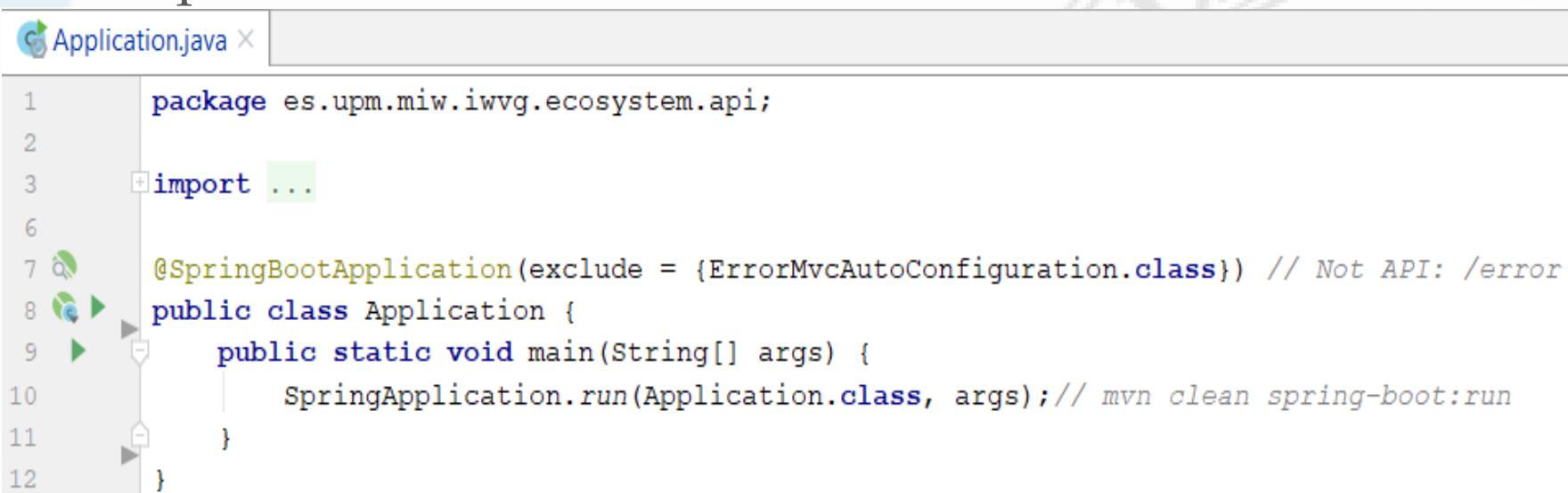
- Configuración: *src/main/resources/application.properties*



```
application.name=@project.artifactId@
build.version=@project.version@
build.timestamp=@maven.build.timestamp@

spring.output.ansi.enabled=always
logging.level.root=WARN
logging.level.es.upm.miw=DEBUG
```

- Arranque

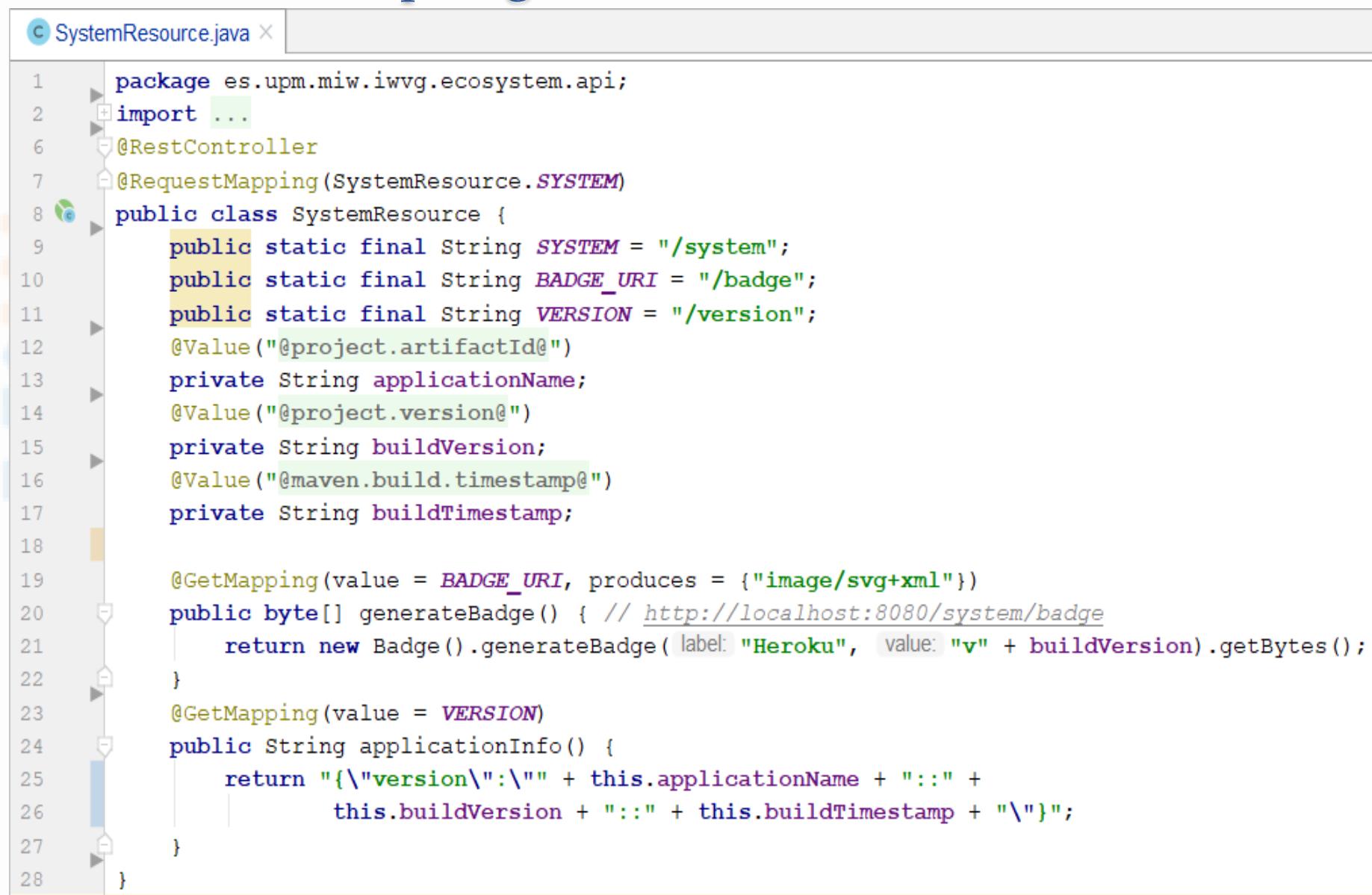


```
package es.upm.miw.iwvg.ecosystem.api;

import ...

@SpringBootApplication(exclude = {ErrorMvcAutoConfiguration.class}) // Not API: /error
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args); // mvn clean spring-boot:run
    }
}
```

Spring Boot: introducción



The screenshot shows a code editor window with the file `SystemResource.java` open. The code is annotated with various colors: blue for packages, imports, and annotations; green for strings and class names; purple for variables and methods; and red for comments. The code defines a `SystemResource` class with static final variables for system paths and badge/version endpoints, and private fields for application name, build version, and timestamp. It contains two `@GetMapping` methods: one for generating a badge and another for returning application info.

```
1 package es.upm.miw.iwvg.ecosystem.api;
2 import ...
3 @RestController
4 @RequestMapping(SystemResource.SYSTEM)
5 public class SystemResource {
6     public static final String SYSTEM = "/system";
7     public static final String BADGE_URI = "/badge";
8     public static final String VERSION = "/version";
9     @Value("${project.artifactId}")
10    private String applicationName;
11    @Value("${project.version}")
12    private String buildVersion;
13    @Value("${maven.build.timestamp}")
14    private String buildTimestamp;
15
16    @GetMapping(value = BADGE_URI, produces = {"image/svg+xml"})
17    public byte[] generateBadge() { // http://localhost:8080/system/badge
18        return new Badge().generateBadge(label: "Heroku", value: "v" + buildVersion).getBytes();
19    }
20
21    @GetMapping(value = VERSION)
22    public String applicationInfo() {
23        return "{\"version\": \"" + this.applicationName + ":" + this.buildVersion + ":" + this.buildTimestamp + "\"}";
24    }
25}
```

Spring Boot: introducción

```
1 package es.upm.miw.iwvg.ecosystem.api;
2 import ...
3 @ExtendWith(SpringExtension.class)
4 @SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
5 @AutoConfigureWebTestClient
6 @TestPropertySource(locations = "classpath:test.properties")
7 class SystemResourceIT {
8     @Autowired
9     private WebTestClient webTestClient;
10
11    @Test
12    void testReadBadge() {
13        byte[] badge =
14            this.webTestClient
15                .get().uri( $: SystemResource.SYSTEM + SystemResource.BADGE_URI) capture of ?
16                .exchange() ResponseSpec
17                .expectStatus().isOk() ResponseSpec
18                .expectBody(byte[].class) BodySpec<byte[], capture of ?>
19                .returnResult().getResponseBody();
20
21        assertNotNull(badge);
22        assertEquals( expected: "<svg", new String(badge).substring(0, 4));
23    }
24
25    @Test
26    void testReadInfo() {
27        String body = this.webTestClient
28            .get().uri( $: SystemResource.SYSTEM + SystemResource.VERSION) capture of ?
29            .exchange() ResponseSpec
30            .expectStatus().isOk() ResponseSpec
31            .expectBody(String.class) BodySpec<String, capture of ?>
32            .returnResult().getResponseBody();
33
34        assertEquals( expected: 3, body.split( regex: ":" : ).length);
35    }
36}
```