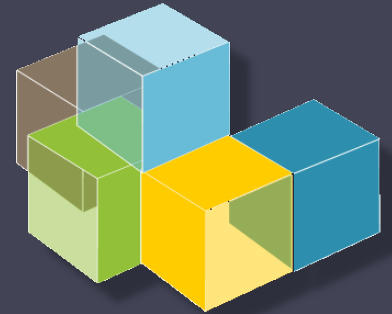




Universidad de Oviedo



Escuela de  
Ingeniería  
Informática



ARQUITECTURA  
DEL SOFTWARE

# Arquitectura del Software

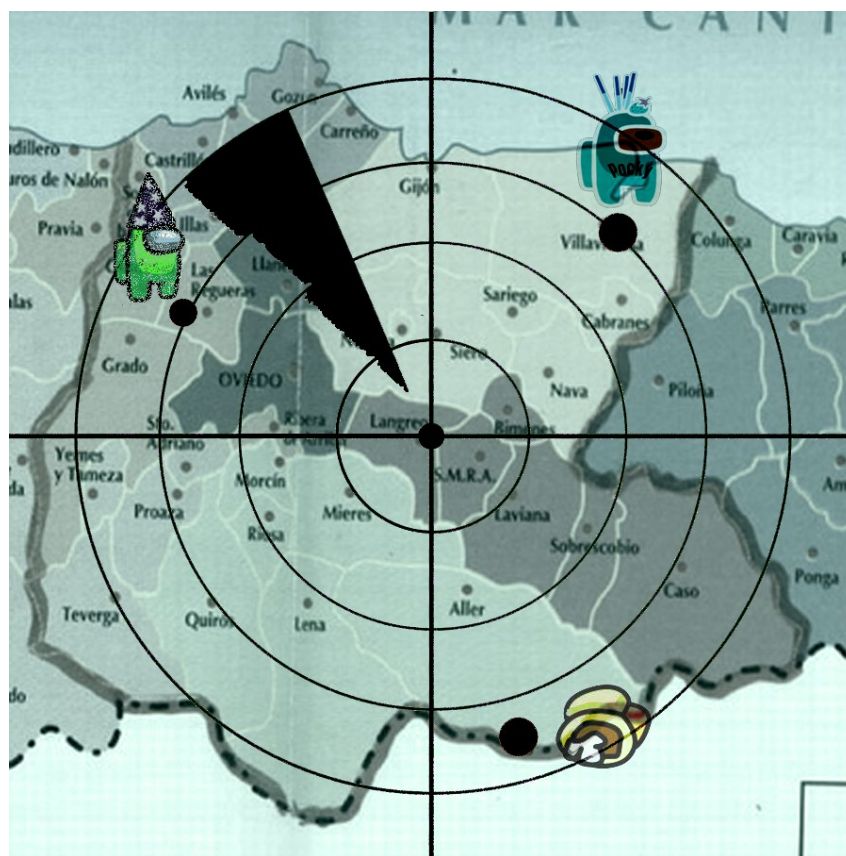
## Lab. 1

- ✓ Introducción a la práctica
- ✓ Organización de Equipos
- ✓ Git
- ✓ Github

2020-21

Jose Emilio Labra Gayo  
Pablo González  
Irene Cid  
Paulino Álvarez

# Introducción a la práctica - Radarín



- ¿ En que trabajaremos en nuestras sesiones prácticas?

## Radarín

- Evaluación:
  - 70% - Trabajo en grupo
  - 30% - Trabajo individual

# Introducción a la práctica - Recursos

- Curso de Arquitectura del Software [website](#)
- [Campus virtual](#)
- Radarin. [Especificación](#)
- Github del proyecto [repositorios](#)

# Organización Equipos

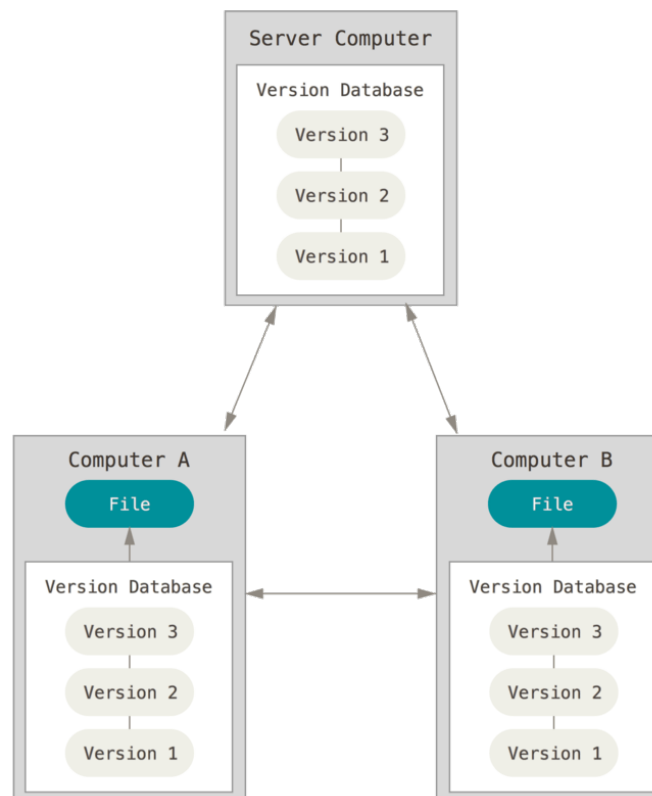
- Cuentas Github RadarinESXA/B
- Reuniones, wiki e incidencias en Github
  - Clases de práctica = reunión
  - Obligatorio tomar actas de reuniones
    - Mínimo obligatorio
      - Fecha
      - Lista de participantes
      - Acuerdos adoptados para la próxima sesión
      - Revisión acuerdos adoptados en la sesión anterior (enlaces a issues y pull request)

# Git - Funcionamiento básico

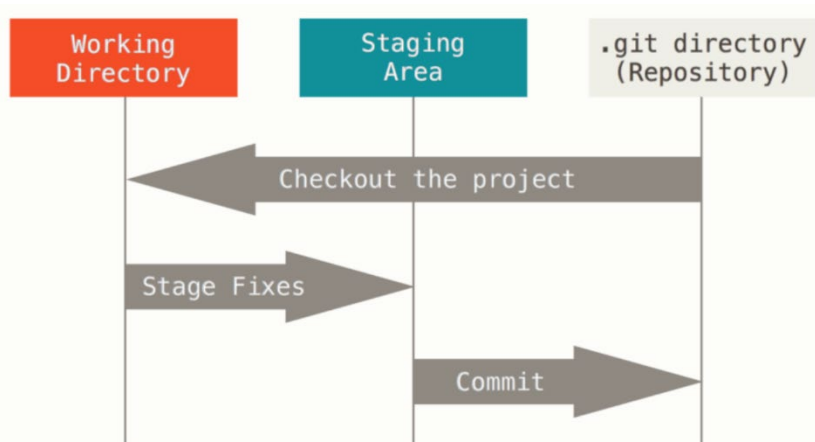
## ¿Por qué un Sistema de Control de Versiones?

- ✓ No trabajamos solos.
- ✓ Si modificamos un proyecto directamente, no tenemos constancia de cómo era antes de los cambios.
- ✓ Seguridad.

## Estructura de Git



# Los tres estados de Git en local



FUENTE <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Fundamentos-de-Git>

Traemos una copia de los ficheros del repositorio: checkout (sobreescribimos) o pull (merge)

Preparas los archivos, añadiéndolos a tu área de preparación : add

Confirmas los cambios tal y como están en el área de preparación y guarda un snapshot : commit

# Primeros pasos con Git y Github

- Cada equipo tendrá un repositorio **Github**
  - Vamos a asignar a cada usuario a su repositorio correspondiente
- Vamos a clonar el repositorio
  - Paso 1: inicializamos nuestro repositorio local
    - >> `git init`
  - Paso 2: Nos traemos el directorio remoto
    - >> `git clone https://github.com/Arquisoft/radarinESXAB.git`
  - Paso 3: Añadimos los cambios necesario
    - <<. Modificar enlaces a `radarin_o`>>
  - Paso 4: Añadimos los cambios al stage
    - >> `git add .`

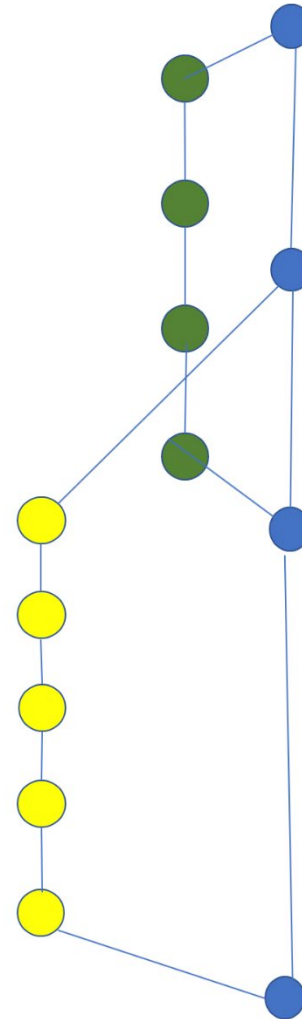
# Primeros pasos con Git y Github

- Paso 5: Registramos los cambios
  - >> `git commit -m "Configuración inicial"`
- Paso 6: Subimos nuestros cambios al remoto
  - >> `git push origin master`
- Saber más...
  - [Git cheatsheet](#) para los comandos más usados
  - Slides: Introduction to git  
<https://www.slideshare.net/jelabra/introduction-to-git-44244608>
  - Conferencia y Talleres de Luis Velasco en el TechFest



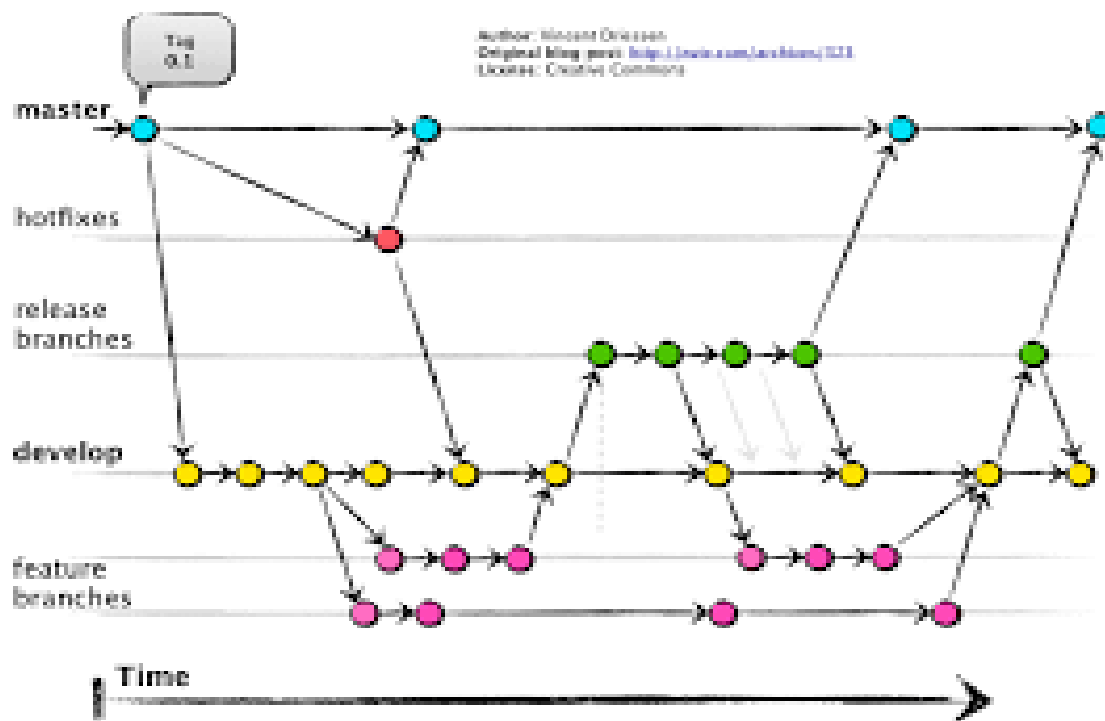
# Creando ramas en Git

- Crear una rama:  
`$ git checkout -b ramal`
- Ver en que rama estamos  
`$ git branch`
- Cambiar de rama  
`$ git checkout master`
- Ver los cambios entre ramas  
`$ git diff --stat master ramal`
- Fusionar ramas  
`$ git checkout master`  
`$ git merge --no-ff ramal`
- Eliminar la rama  
`$ git branch -d ramal`
  
- Paso 1: Vamos a crear una rama develop en nuestro remote
  - >> `git checkout -b develop`
  - >> `git push origin develop`

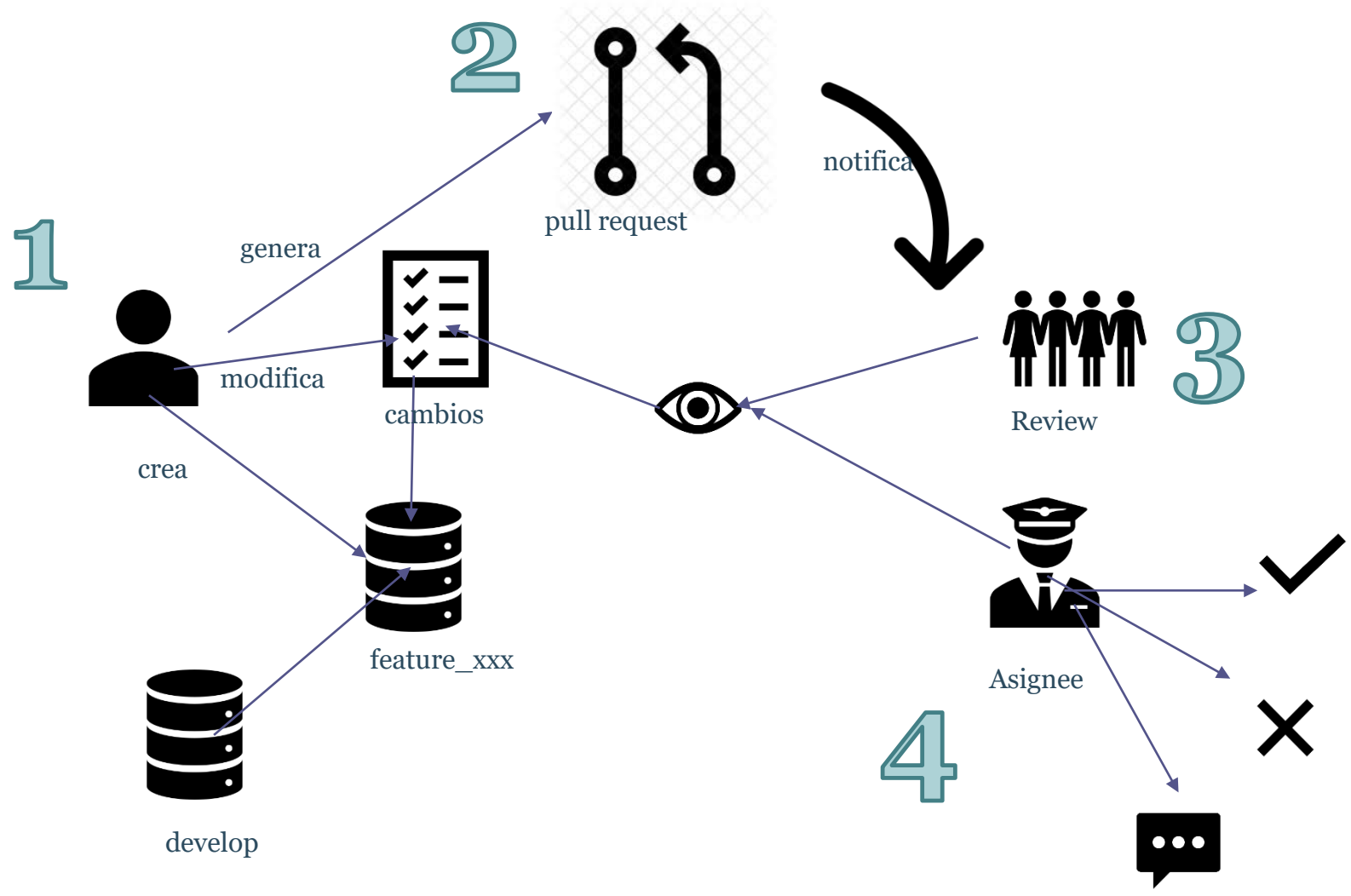


# Git-Flow

- El trabajo en un equipo de desarrollo : iteraciones, los evolutivos, solución a errores.
- Nuestro repositorio debe estar preparado para ello. Git-flow promueve la siguiente jerarquía de ramas:



# Pull request



# Pull request - Pasos

- Crear la rama
  - `$ git flow feature start RE1 develop`
  - `$ git checkout -b feature-RE1 develop`
- Añade tu nombre en **README.md** en el apartado *Colaboradores*
- Subir los cambios en local
  - `$ git add .`
  - `$ git commit`
- Subir los cambios
  - `$ git push --set-upstream origin feature-RE1`
- Ir a github y solicitar una pull request



5 commits      3 branches      0 packages      0 releases

Your recently pushed branches:

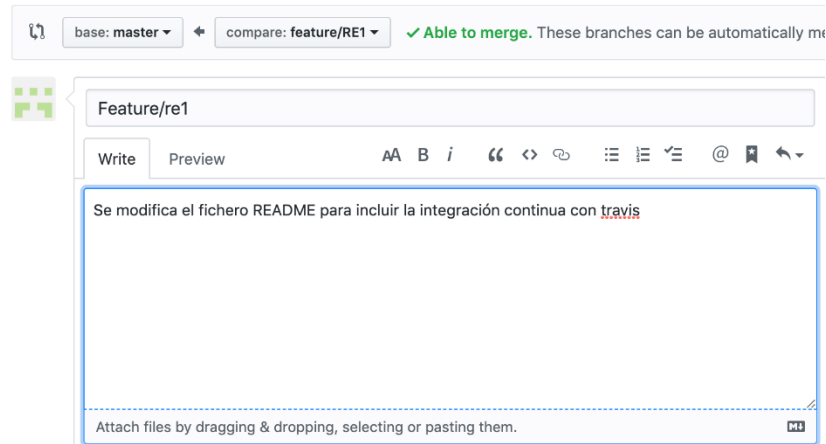
 <b>develop</b> (about 1 hour ago)	<a href="#">Compare &amp; pull request</a>
 <b>feature/RE1</b> (1 minute ago)	<a href="#">Compare &amp; pull request</a>

# Pull request - Pasos

- Añadir los comentarios: (también se pueden añadir revisores)

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across](#)



The screenshot shows the GitHub interface for opening a pull request. At the top, there are two dropdown menus: 'base: master' and 'compare: feature/RE1'. To the right of these is a green checkmark and the text 'Able to merge. These branches can be automatically merged'. Below this is a text input field containing 'Feature/re1'. Underneath the input field are two tabs: 'Write' (selected) and 'Preview'. To the right of the tabs is a rich text editor toolbar with icons for bold, italic, quote, code, link, list, table, linkify, mention, and undo. The main text area contains the text: 'Se modifica el fichero README para incluir la integración continua con [travis](#)'. At the bottom of the text area, there is a dashed line and the text: 'Attach files by dragging & dropping, selecting or pasting them.' with a small icon to the right.

- Los revisores de código tienen tres opciones: añadir comentarios, aceptar los cambios, rechazarlos