Conceptos básicos de Gitlab

Proyectos Integrados
3º Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Hipólito Guzmán Miranda
Universidad de Sevilla

Contenido

- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

Contenido

- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

¿Qué es gitlab?

Gitlab es una plataforma de desarrollo (DevOps = development operations).

Como tal, ofrece distintas funcionalidades:

- Repositorios git
- Issue tracker
- Integración contínua
- Interfaz web
- Gestión de usuarios y grupos
- … y muchas más…

¿Por qué usar gitlab?

- Una buena forma de aprender y aplicar las distintas buenas prácticas en desarrollo de proyectos
- A diferencia de github, no sólo tienes la instancia pública, sino que puedes instalarlo gratuitamente en tus propios servidores
- Gitlab Community Edition es 100% open source y gratuita
- (también existe una Enterprise Edition, de pago, tanto de gitlab como de github)

Organización de Gitlab

- En gitlab existen usuarios y grupos
- Los grupos a su vez pueden tener otros grupos dentro (subgrupos)
  - https://gitlab.com/pigierm tiene varios grupos dentro, que a su vez tienen más grupos
- Los proyectos pueden pertenecer a un usuario o grupo
  - Los proyectos de un usuario estarán bajo https://gitlab.com/nombreusuario
- Los permisos se heredan jerárquicamente
  - Si eres miembro de un grupo, tienes acceso a sus proyectos
  - También puedes ser miembro de un proyecto

Proyectos de Gitlab

- En gitlab, un proyecto tiene:
  - Un repositorio git
  - Un Issue Tracker
  - Un sistema de Integración Continua
  - Un registro de Contenedores (imágenes Docker)
  - Y más cosas …
- No es obligatorio usarlo todo!
  - Lo mejor es aprenderlo poco a poco

Contenido

- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

Markdown

- Tanto en gitlab como en github veréis ficheros llamados `README.md`
- La extensión `.md` significa que es un fichero escrito en lenguaje Markdown
- Es un lenguaje de 'markup' (marcado) muy ligero y sencillo de usar
  - Es casi como escribir texto plano
- Si existe un fichero `README.md` en una carpeta, el interfaz web de gitlab lo renderiza automáticamente

- **Stash**: es un área temporal donde se pueden esconder temporalmente cambios mientras se trabaja en otra cosa.

A continuación, veremos unos comandos básicos para manejarnos con ``git``.

Configurando ``git``

La configuración de ``git`` puede editarse en tres ficheros diferentes:

  - A nivel de sistema: ``/etc/gitconfig (en Linux)``
  - A nivel de usuario: ``~/.gitconfig``
  - A nivel de repositorio: ``<path_al_repo>/.git/config``

La configuración específica de repositorio tiene prioridad sobre la de usuario, que a su vez tiene prioridad sobre la de sistema.

Editamos ``~/.gitconfig``, que es la configuración a nivel de usuario, con el editor de texto que prefiramos (por ejemplo ``nano`` o ``gedit``).

Un fichero de configuración de ejemplo sería:

```
[user]
    name = Nombre Apellido
    email = usuario@correo.dominio
[core]
    editor = nano
[color]
    ui = auto
[push]
    default = matching
```

Para comprobar la configuración que está viendo ``git``, puedes hacer:

```
git config --list
```

Git help

En esta sesión no va a dar tiempo a que veamos todo ``git`` en profundidad. Algunos comandos pueden recibir más argumentos de los que veremos aquí. En caso de duda, puedes utilizar ``git help`` para obtener más información sobre ``git`` o sobre algún comando concreto:

```
git help [comando]
```

Git básico

```
- **Stash**: es un área temporal dond
otra cosa.

A continuación, veremos unos comandos

# Configurando ``git``

La configuración de ``git`` puede edi

  - A nivel de sistema:          ``/e
  - A nivel de usuario:          ``~/
  - A nivel de repositorio:      ``<p

La configuración específica de reposi
prioridad sobre la de sistema.

Editamos ``~/.gitconfig``, que es la
prefiramos (por ejemplo ``nano`` o ``

Un fichero de configuración de ejempl

    [user]
        name = Nombre Apellido
        email = usuario@correo.domini
    [core]
        editor = nano
    [color]
        ui = auto
    [push]
        default = matching

Para comprobar la configuración que e

    git config --list

# Git help

En esta sesión no va a dar tiempo a q
recibir más argumentos de los que ver
obtener más información sobre ``git``

    git help [comando]

# Git básico
```

https://gitlab.com/hgpub/gitintro

GitLab Menu Search GitLab 14 13

- **Stash**: es un área temporal donde se pueden esconder temporalmente cambios mientras se trabaja en otra cosa.

A continuación, veremos unos comandos básicos para manejarnos con `git`.

Configurando `git`

La configuración de `git` puede editarse en tres ficheros diferentes:

- A nivel de sistema: `/etc/gitconfig (en Linux)`
- A nivel de usuario: `~/.gitconfig`
- A nivel de repositorio: `<path_al_repo>/.git/config`

La configuración específica de repositorio tiene prioridad sobre la de usuario, que a su vez tiene prioridad sobre la de sistema.

Editamos `~/.gitconfig`, que es la configuración a nivel de usuario, con el editor de texto que prefiramos (por ejemplo `nano` o `gedit`).

Un fichero de configuración de ejemplo sería:

```
[user]
    name = Nombre Apellido
    email = usuario@correo.dominio
[core]
    editor = nano
[color]
    ui = auto
[push]
    default = matching
```

Para comprobar la configuración que está viendo `git`, puedes hacer:

```
git config --list
```

Git help

En esta sesión no va a dar tiempo a que veamos todo `git` en profundidad. Algunos comandos pueden recibir más argumentos de los que veremos aquí. En caso de duda, puedes utilizar `git help` para obtener más información sobre `git` o sobre algún comando concreto:

```
git help [comando]
```

Git básico

Markdown

- El interfaz web de gitlab siempre te ofrece una pestaña de ‘preview’ para ver cómo va a quedar el Markdown
- Muy útil para documentar rápidamente
- Sintaxis: https://docs.gitlab.com/ee/user/markdown.html
  - Podéis buscar para encontrar cómo se hace cada cosa (buscad por table, image, link, etc…)

Markdown

- Los proyectos deben tener un README.md que indique qué son, cómo se implementan/compilan, y cómo se utilizan
  - Pueden contener enlaces a otra documentación, por ejemplo documentación técnica o manuales de usuario
- Markdown también se usa en el issue tracker

Contenido

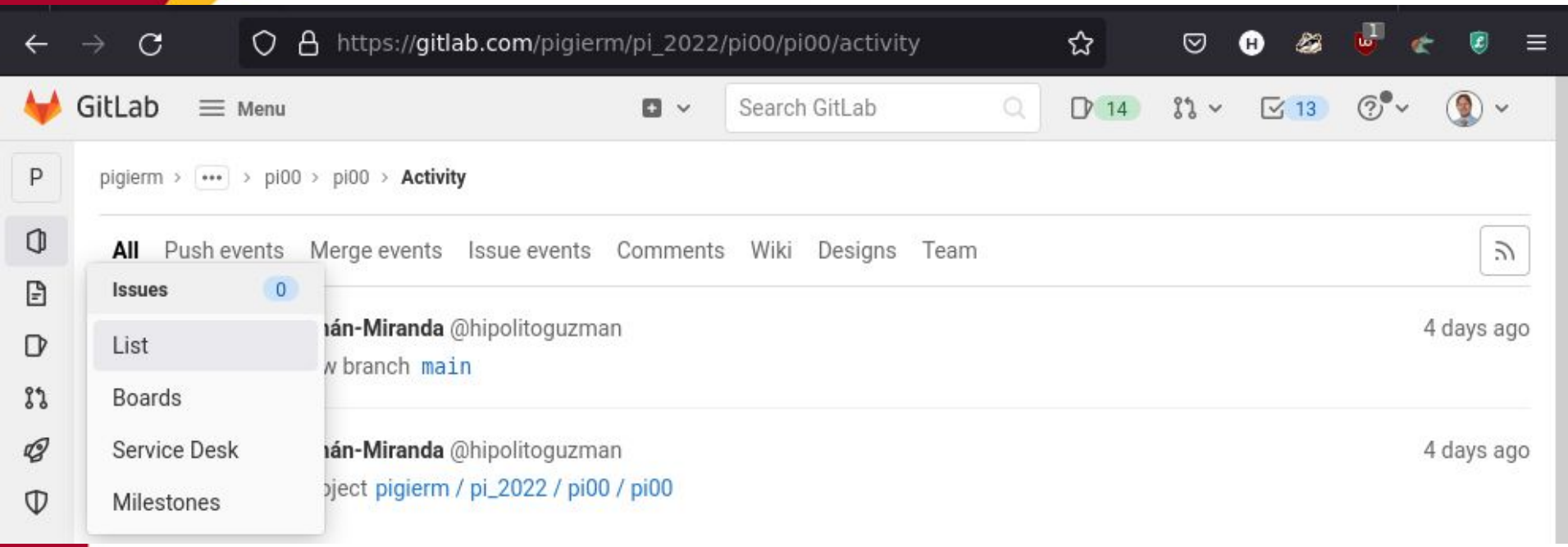
- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

Issue tracking con gitlab

Introducción al issue tracker de gitlab

Issue tracking con gitlab

Crear un issue

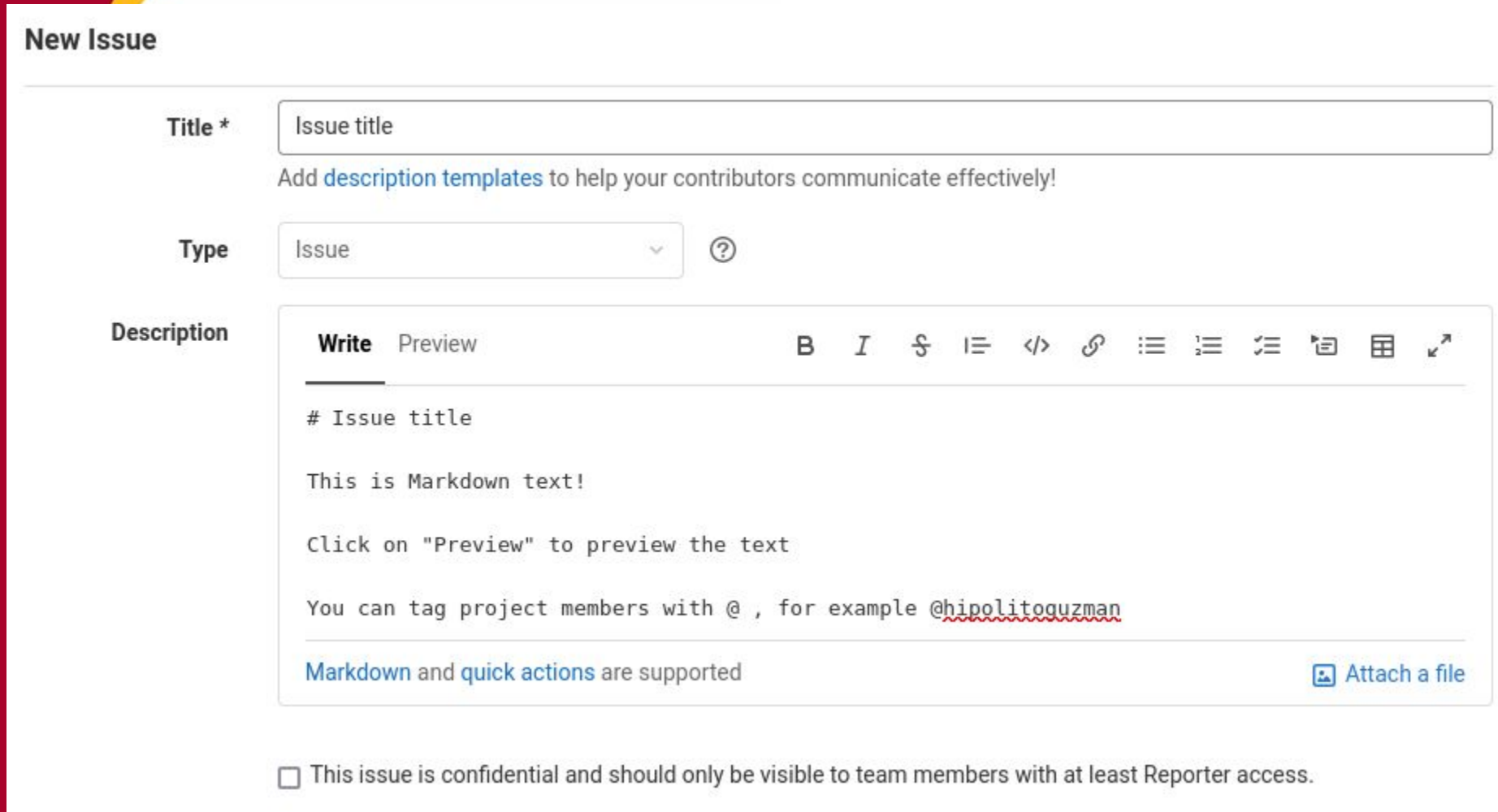


The Issue Tracker is the place to add things that need to be improved or solved in a project

Issues can be bugs, tasks or ideas to be discussed. Also, issues are searchable and filterable.

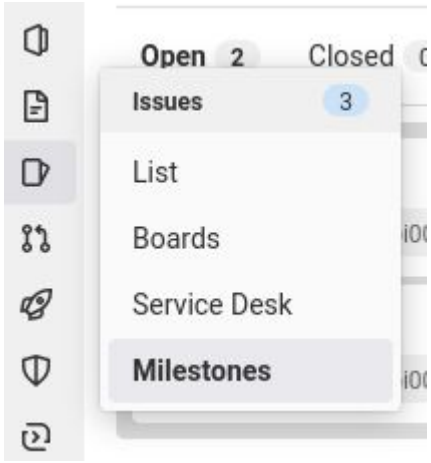
New issue | Import issues

Crear un issue



Crear un hito

Los esfuerzos de desarrollo se pueden organizar en hitos (como los hitos en gestión de proyectos)



Crear un hito

New Milestone

Title: Version 0.1

Start Date: Select start date — Clear start date

Due Date: Select due date — Clear due date

Description

Write | Preview

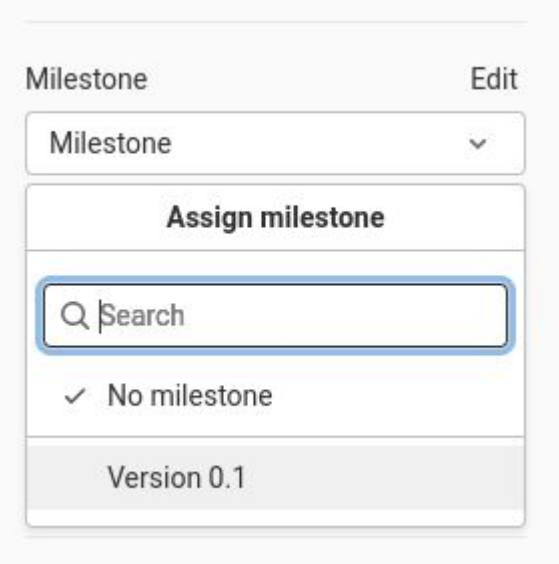
```
# Version 0.1

For this version, we want to have (for example) a release that compiles
```

Markdown is supported — Attach a file

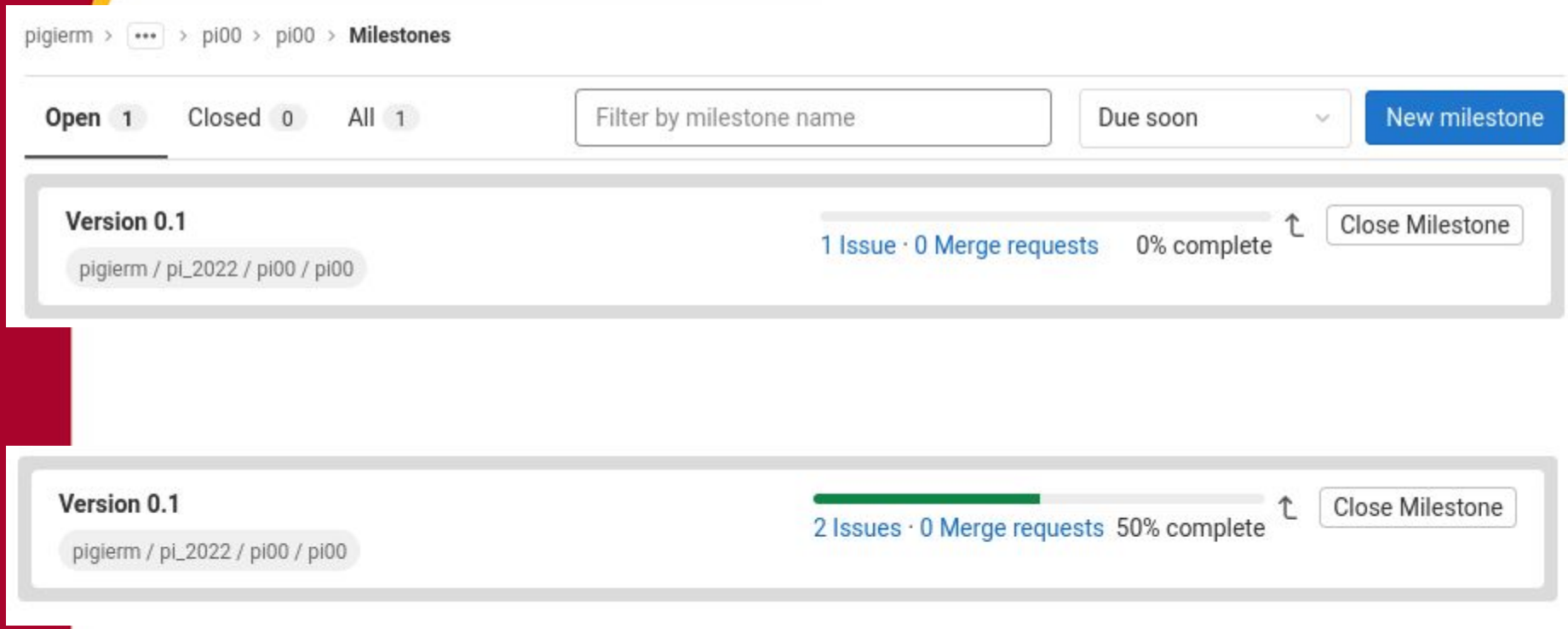
Create milestone | Cancel

Asignar un issue a un hito

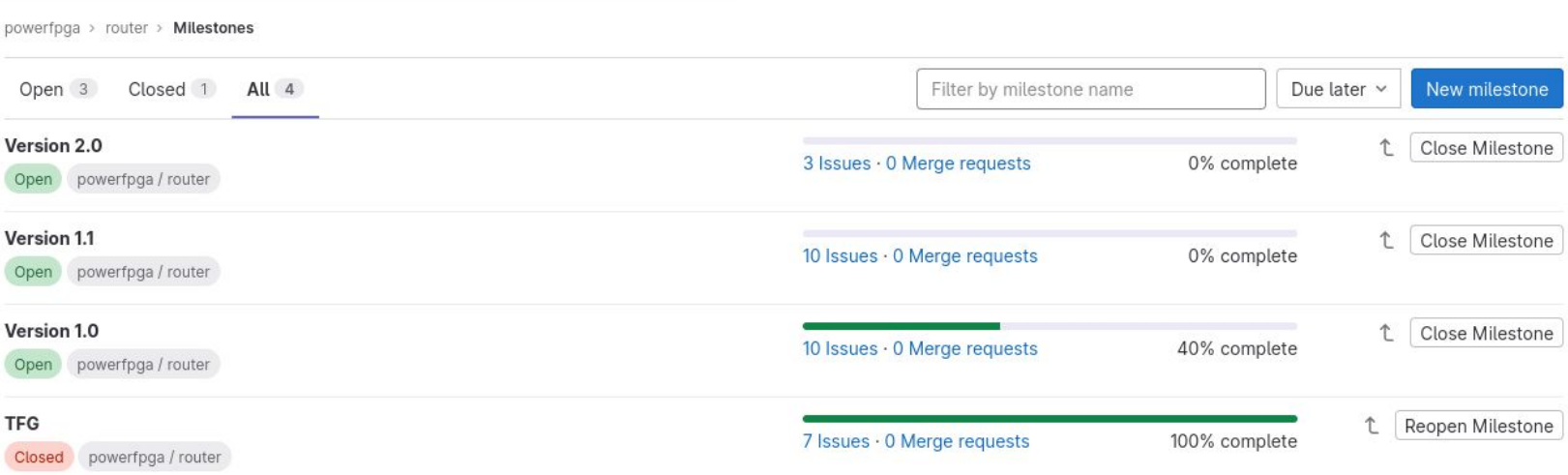
Entramos en el issue y, a la derecha:



Así se ven los hitos



Ejemplo hitos



Ejemplo hito

powerfpga › router › Milestones › **Version 1.0**

Open **Milestone**

Edit | Promote | Close milestone | Delete

Version 1.0

Milestone ID: 5

Fixing all that is broken or doesn't work as expected

Issues 10 | Merge requests 0 | Participants 2 | Labels 0

Unstarted Issues (open and unassigned) 1

- Bug: Test vehicle 4 bis (SPI RX) loses data #27

Ongoing Issues (open and assigned) 5

- Wanted: Error and Warning filtering for Xilinx ISE #30
- Testing: big python script to test the switch with the simulation results never finishes #25
- Bug: modified FIFO doesn't infer block RAMs when synthesized #24
- Bug: Switch messes up headers in FPGA implementation but not in simulation (when synthesized with GHDL) #19
- Documentation: Definiton of 'jump' and update of 'chain of jumps' definition in README.md #14

Completed Issues (closed) 4

- Xilinx implementation inside the CI fails (actually, anything that uses fusesoc will fail) #26
- Improvement: implementation scripts for LX9 Microboard #23
- Testing: create csv files with input frames and expected output frames from the VUnit testbenches to use in python scripts #21
- Testing: testing with multiple router configurations #16

40% complete

Start date — Edit
No start date

Due date — Edit
No due date

Issues 10 — New issue
Open: 6 **Closed: 4**

Time tracking
No estimate or time spent

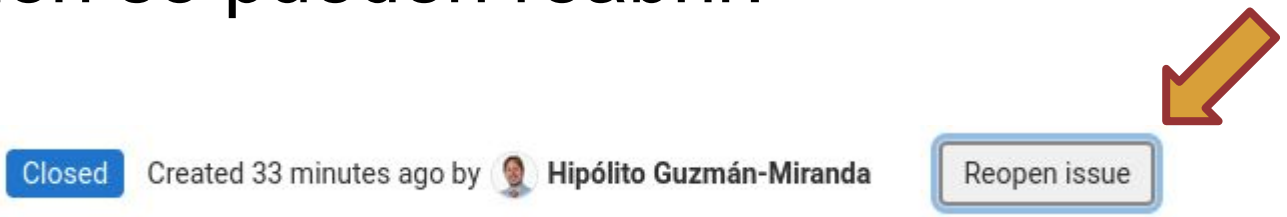
Merge requests 0
Open: 0 **Closed: 0** **Merged: 0**

Releases
None

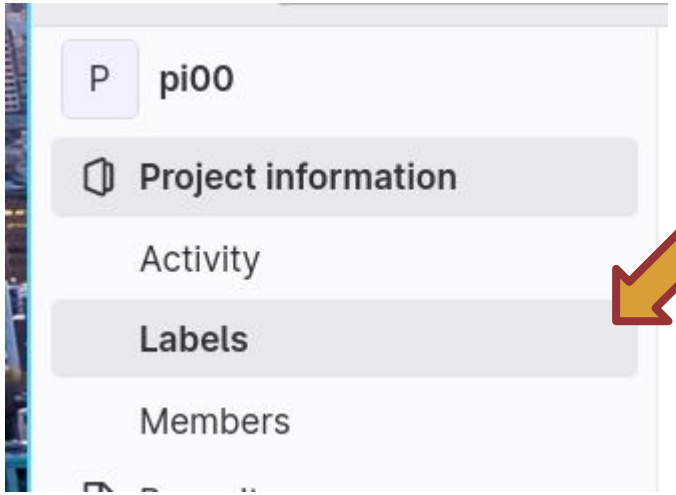
Reference: powerfpga/router%"...

Cerrar un issue

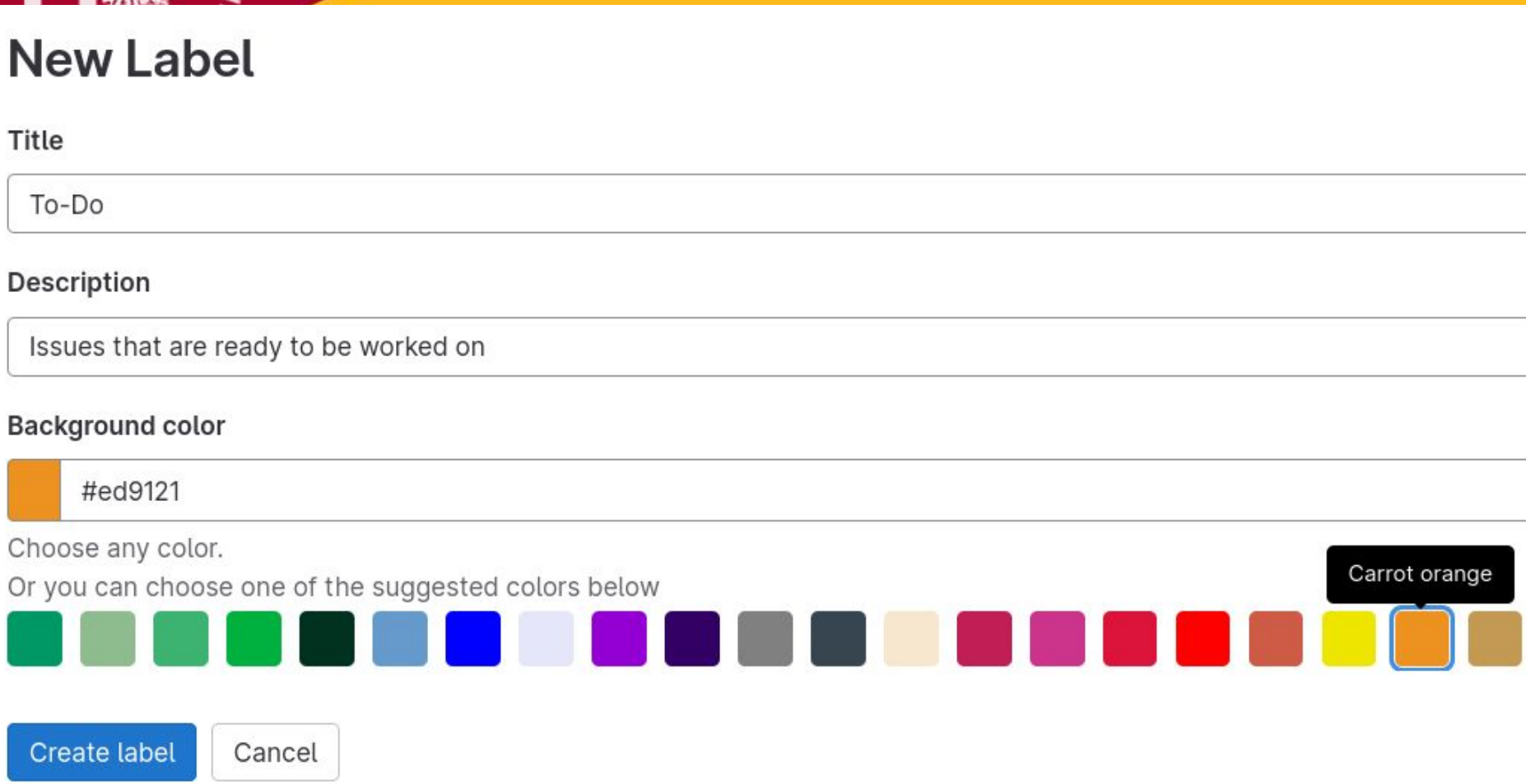
También se pueden reabrir:



Etiquetas en issues



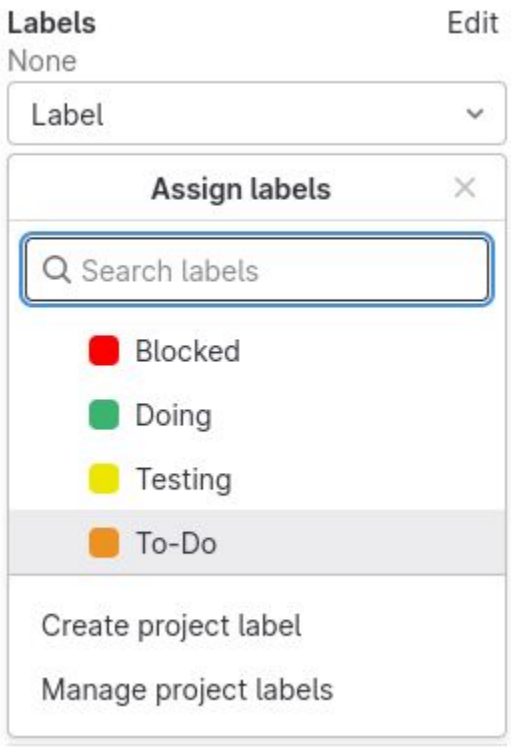
Crear etiqueta



Crear etiqueta

Etiquetas en issues

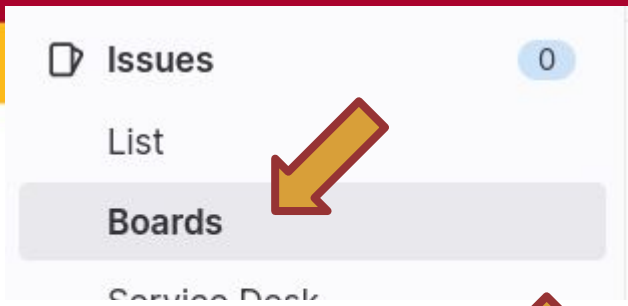
Abriendo el issue, a la derecha, podemos asignarle una o varias etiquetas

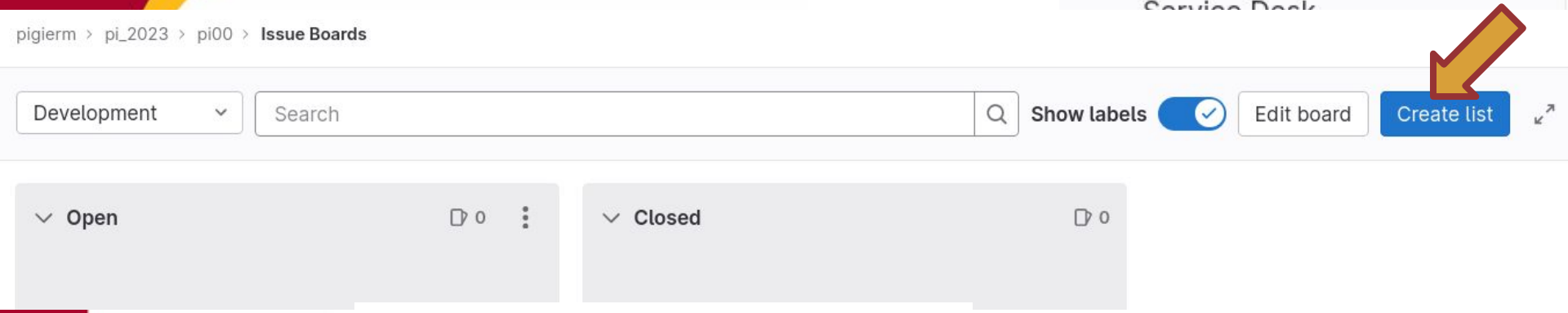


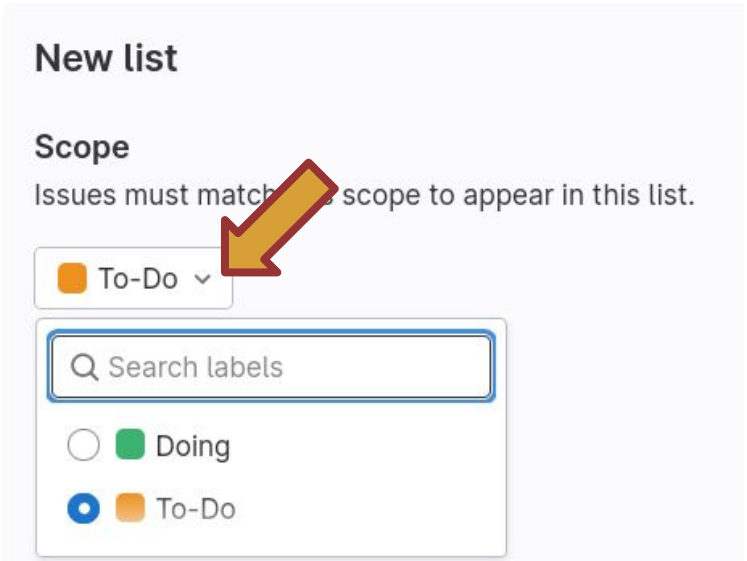
Issue Boards

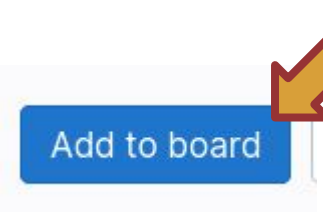
- En Issues -> Boards
- Por defecto tenemos 2 listas: open y closed
- Podemos añadir más listas (todos los issues que tengan cierta etiqueta aparecerán en esa lista)
- Podemos arrastrar issues entre listas (cambian sus etiquetas/estado automáticamente)
- Esto nos permite hacer cosas como Kanban (estilo Trello, con la forma de trabajar que prefiramos)

Issue Boards



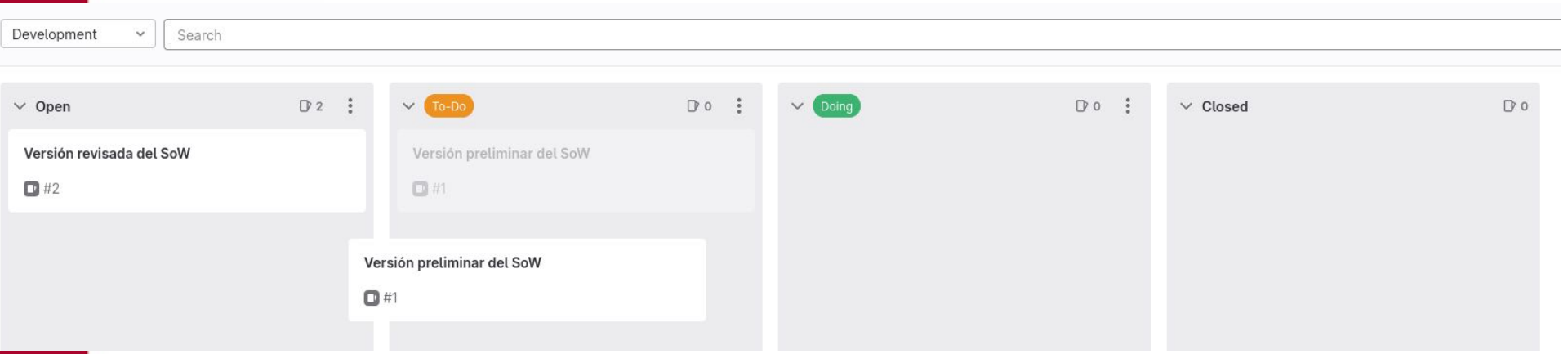






Issue Boards

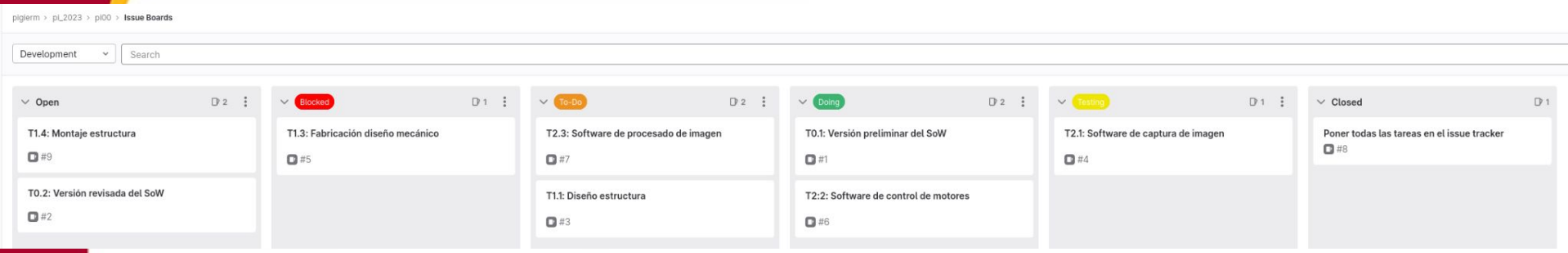
Un ejemplo de Kanban (To-Do, Doing y Done/Closed)



Podemos arrastrar issues de una lista a otra

Issue Boards

Un ejemplo con más categorías:



(Open, Blocked, To-Do, Doing, Testing, Closed)

Podéis configuraros uno o varios issue boards como os venga mejor. Los issues pueden tener más de una etiqueta

Issue Boards

Algunos ejemplos muy interesantes, algo más avanzados, de cómo usar esto para organizarse pueden verse en:

- 4 Ways to use Gitlab Issue Boards, Victor Wu: https://about.gitlab.com/blog/2018/08/02/4-ways-to-use-gitlab-issue-boards/

Cuando el issue es un bug

En un buen bug report es INDISPENSABLE que conste:

1. Pasos para reproducirlo
2. Qué esperabas que ocurriera
3. Qué ocurrió realmente

(No todos los issues son bugs, pero los bug reports deben estar bien formados)

Contenido

- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

Ejercicio manejo básico

- Entrad en el issue tracker de vuestro proyecto
- Click en ‘New issue’ -> añadid un bug llamado “usuario_testing” (cada uno con su nombre de usuario).
- Ved qué ocurre si lo vais asignando a otros miembros del equipo
- Podéis probar a ponerle etiquetas y buscar por etiquetas

Ejercicio manejo básico

- Cread un ‘Milestone’ que se llame “Pruebas issue tracker”
- Asignad todos los issues a dicho milestone
- Comentad en algún issue que haya creado algún compañero
- Podéis utilizar # para referenciar a otro issue y @ para referenciar a otro usuario
- Cuando hayáis terminado de probar cosas, cerrad el bug e id viendo cómo va quedando el Milestone

Contenido

- ¿Qué es gitlab?
- Markdown
- Issue tracking con gitlab
- Ejercicio manejo básico
- Tarea

Tarea

- Cuando las tengáis definidas, añadid las tareas de vuestro proyecto de la asignatura al issue tracker
- De esta forma las podréis asignar y seguir
- También tendría sentido que las agrupárais en hitos (Milestones) y/o que utilicéis los Issue Boards
- Tendréis una idea de cuándo habéis ido trabajando en ellas y cuándo las habéis resuelto

Tarea

Convención de nombres sugerida:

- Si es un bug simplemente la descripción
- Si es una tarea: “T2.1: ...” (tarea 1 del WP 2)
- Si es una ampliación o sugerencia, comenzad con “Wanted: …” o “Wishlist: …”

Transparencias extra

Usando etiquetas

- Usando etiquetas, se puede hacer ‘triage’ de los bugs/issues
- Esto normalmente se va adoptando poco a poco, en función de las necesidades del equipo

Estados (‘states’) de un bug

- New (nuevo)
- Feedback (para pedir información a otro miembro del equipo)
- Acknowledged (confirmado)
- Resolved (resuelto)
- Closed (cerrado)

Gravedad (‘severity’) de un bug

- feature (para añadir)
- text (texto por cambiar)
- minor (poca gravedad)
- major (bastante grave)
- crash (cuelgue)
- block (impide que el resto del equipo pueda trabajar)

Prioridad (‘priority’) de un bug

- none (usualmente significa: “aún no hemos decidido la prioridad)
- low
- normal
- high
- urgent
- immediate

Usando etiquetas

- Un ejemplo de cómo se puede hacer esto puede encontrarse en: https://about.gitlab.com/handbook/engineering/quality/issue-triage/