



ARQUITECTURA DEL SOFTWARE 2021-22

Jose Emilio Labra Gayo
Pablo González
Irene Cid
Hugo Lebrede



Escuela de
Ingeniería
Informática



Universidad de Oviedo

Laboratorio 1

- Introducción a la práctica
- Organización de equipos
- Git
- GitHub

● Introducción a la práctica

¿Que vamos a hacer?

Desarrollar un sistema de venta online llamado **Descentralized Delivery** (DeDe) que sea capaz de respetar la privacidad de los clientes siguiendo SOLID

¿Qué recursos debemos utilizar?

- El [website](#) de Arquitectura del Software donde se encontrará toda la documentación de la asignatura.
- [Campus virtual](#).
- La [Especificación](#) de DeDe.
- Github del proyecto [repositorios](#) .





¿Cómo se va a **evaluar** la práctica?

70% - Trabajo en grupo 

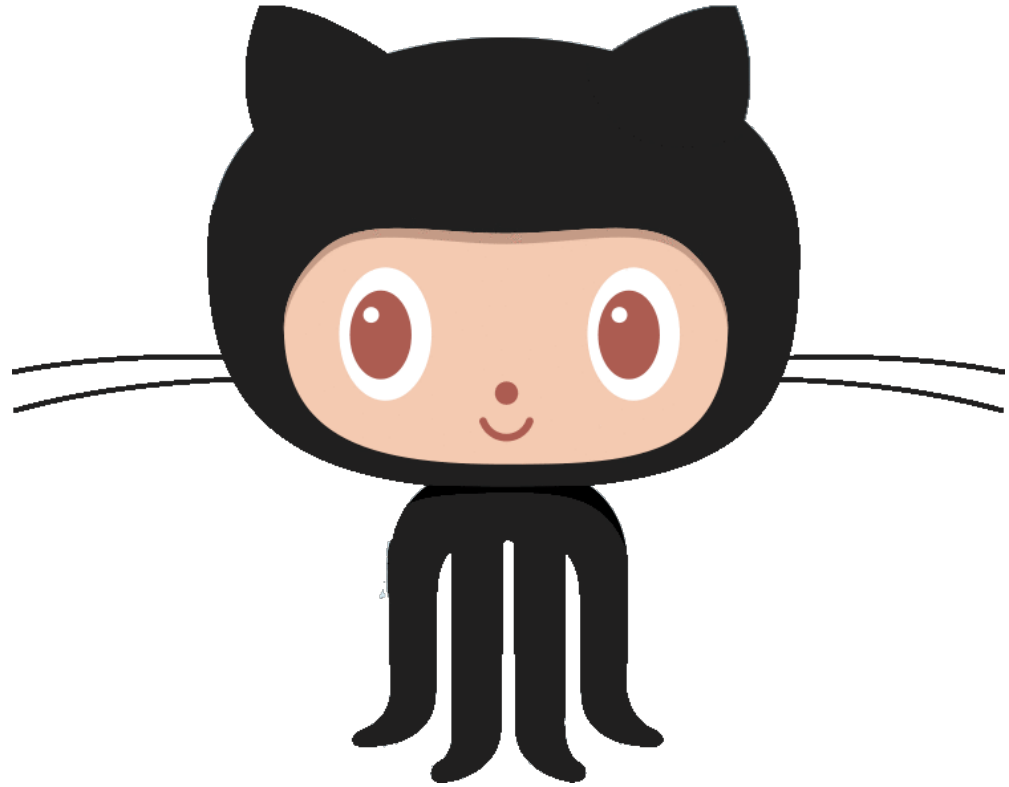
30% - Trabajo individual 

● Organización de equipos

Actas de reuniones

- Clases de práctica == reunión.
 - Se pueden hacer otras reuniones aparte de las clases de laboratorio
- **Obligatorio** tomar acta de las reuniones.
- Se utilizará la sección “wiki” del repositorio para las actas.
- Formato mínimo **obligatorio**:
 - ☐ Fecha 
 - ☐ Lista de participantes 
 - ☐ Acuerdo en el Reparto de trabajo adoptados para la próxima sesión (issues abiertos) 
 - ☐ Revisión de estado de tareas en reuniones anteriores ☒
 - Enlaces a **Issues** y **Pull requests** 
 - ☐ Breve descripción decisiones tomadas
 - Preferible enlazar a registros de decisiones arquitectónicos (<https://adr.github.io/>)

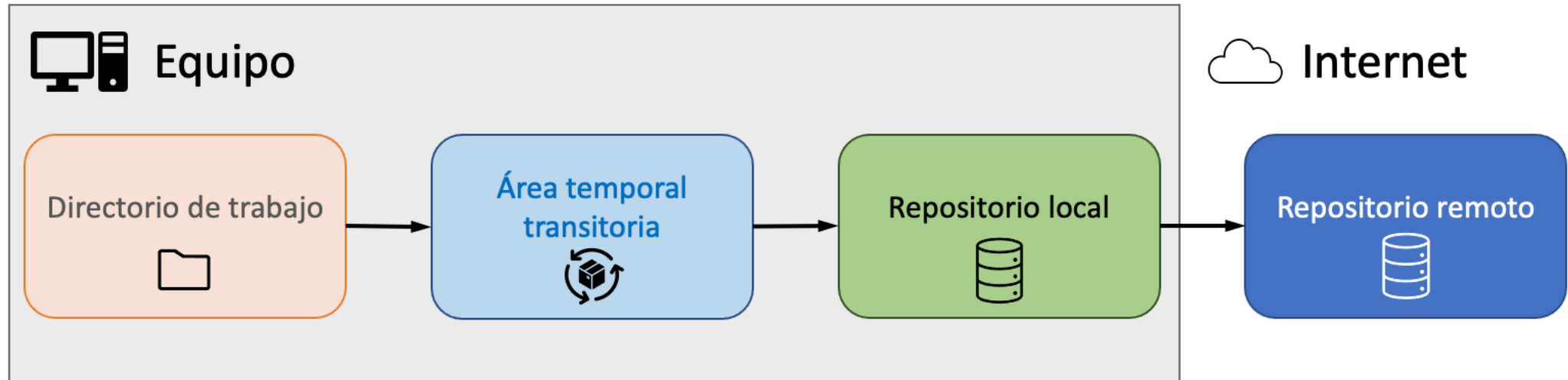
● Git



● Git

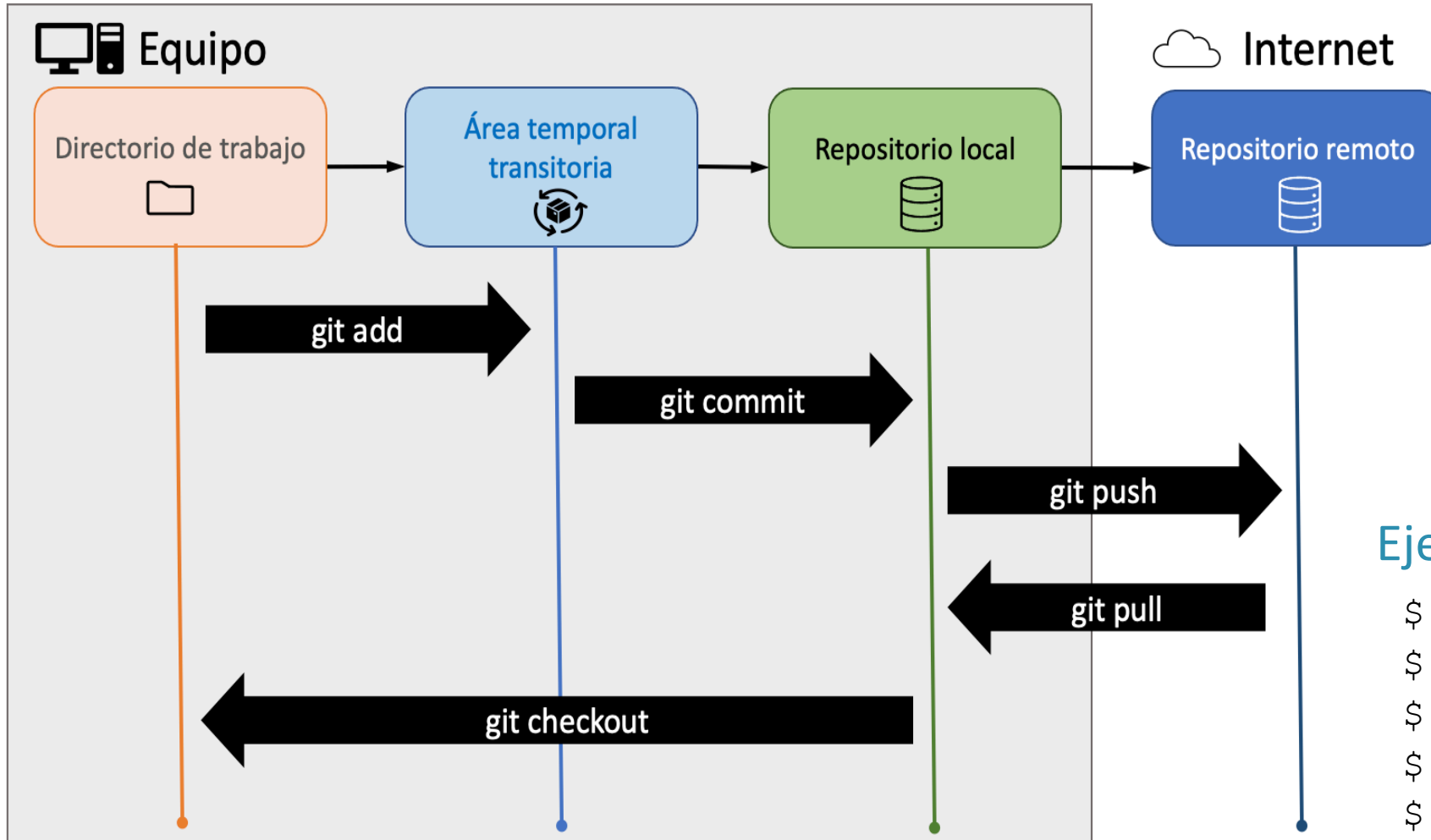


Git Workflow





Git Comandos comunes

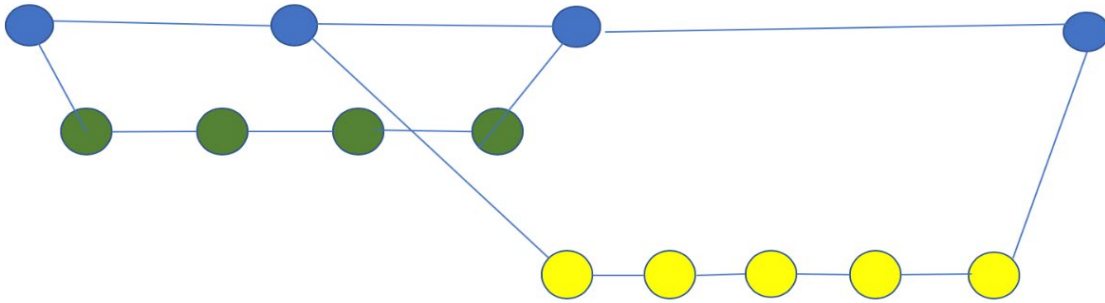


Ejemplo básico

```
$ git init
$ git clone urlRepositorio
$ git add .
$ git commit -m "Mensaje"
$ git push origin master
```

● Git

Trabajar con ramas



- Crear una rama:
`$ git checkout -b rama1`
- Ver en qué rama estamos
`$ git branch`
- Cambiar de rama
`$ git checkout master`
- Ver los cambios entre ramas
`$ git diff --stat master rama1`
- Fusionar ramas
`$ git checkout master`
`$ git merge --no-ff rama1`
- Eliminar la rama
`$ git branch -d rama1`
- Crear rama develop
`$ git checkout -b develop`
- Añadir la rama a nuestro repositorio
`$ git push origin develop`

● Git

Estrategias de ramificación

- Todas las estrategias se basan esencialmente en la forma en la que van a crear o no ramas y fusionarlas a la rama principal.
- **No existe** una estrategia perfecta 😞. Depende de muchos factores desde la veteranía del equipo hasta las tecnologías utilizadas o las políticas de la organización.
 - <https://martinfowler.com/articles/branching-patterns.html#Trunk-basedDevelopment>
- Varias estrategias:
 - Git Flow. Vincent Driessen en 2010: [A successful Git branching model](#)” (Tiene código comandos git).
 - [GitHub Flow](#).
 - Trunk-based development: <https://trunkbaseddevelopment.com/>



Pull Request

Pull Request es la aportación más importante que ha hecho GitHub: Poder integrar a un repositorio código hecho por otros usuarios.

Pasos

- Crear la rama

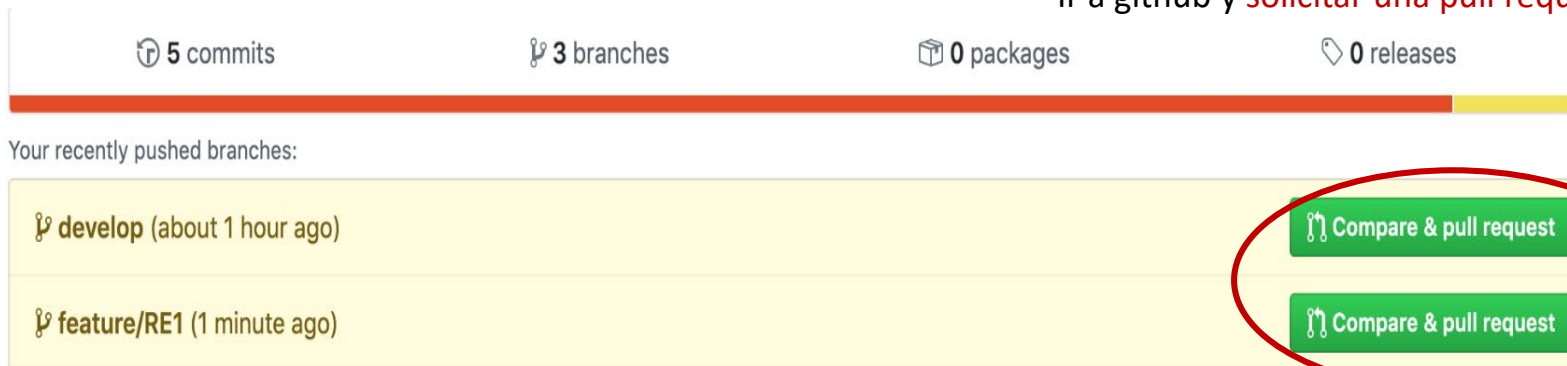
```
$ git flow feature start RE1 develop
```

```
$ git checkout -b feature-RE1 develop
```
- Añade tu nombre en **README.md** en el apartado *Colaboradores*
- Subir los cambios en local

```
$ git add .
```

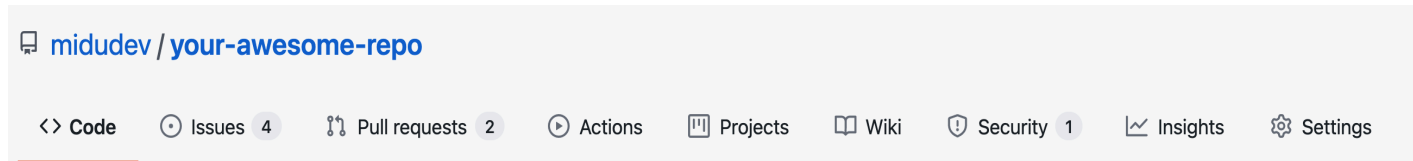
```
$ git commit
```
- Subir los cambios

```
$ git push --set-upstream origin feature-RE1
```
- Ir a github y **solicitar una pull request**





Pull request



Open a pull request

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

Rama destino. → 1 base: master

← 2 compare: feature-new-cool-thing → **Rama que queremos fusionar**

3 Fix problem when creating an user

4 **Añadir contexto adicional.**

Crear pull request → 5

Crear pull request

Changes

Use uuid v4 instead a manual generated id with Math.random to avoid problems when using the id in some places.

Fixed issues

Fixed #30 on generating new users

Attach files by dragging & dropping, selecting or pasting them.

● GitHub

¿Por qué es necesario?

- Organizar el trabajo de un grupo
- Trabajamos con software no con entidades del mundo físico
 - ✓ Este sector ajusta muy bien al teletrabajo.
 - ✗ El software no es algo físico. Hay que gestionar los proyectos de forma diferente a otros sectores.
- Seguridad 🛡️

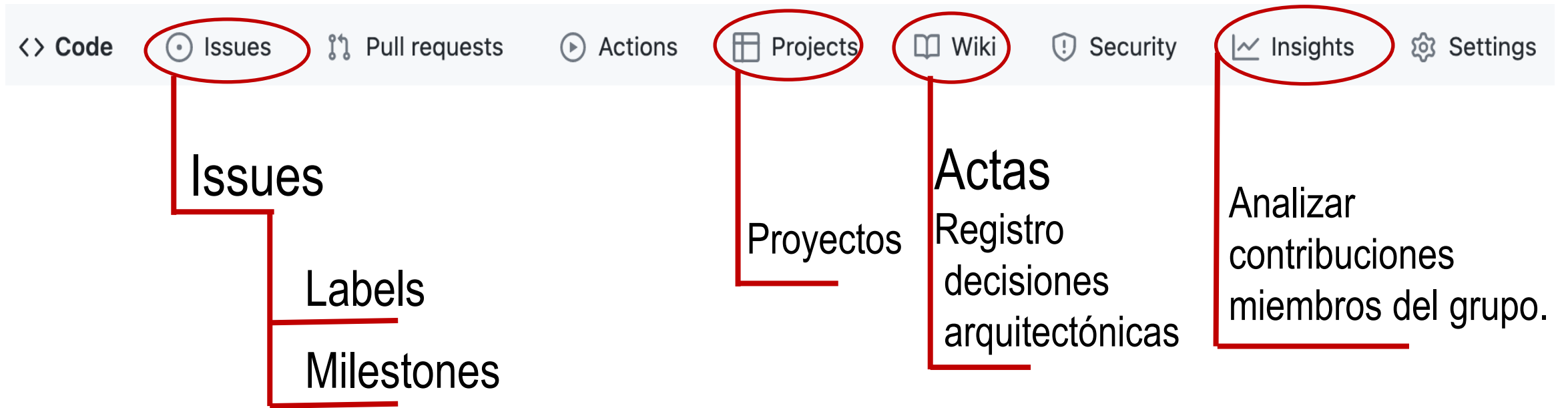
GitHub

Ventajas de la gestión de proyectos

- Planificación del proyecto (Futuro).
- Controlar el estado actual del proyecto.
 - Detectar cuellos de botella.
 - Carga de trabajo de los miembros del equipo.
 - Problemas actuales.
- Reporting del desempeño (Pasado).
 - Evaluación de la contribución de los miembros del equipo.

● GitHub

¿Qué necesitamos de GitHub para gestionar proyectos?



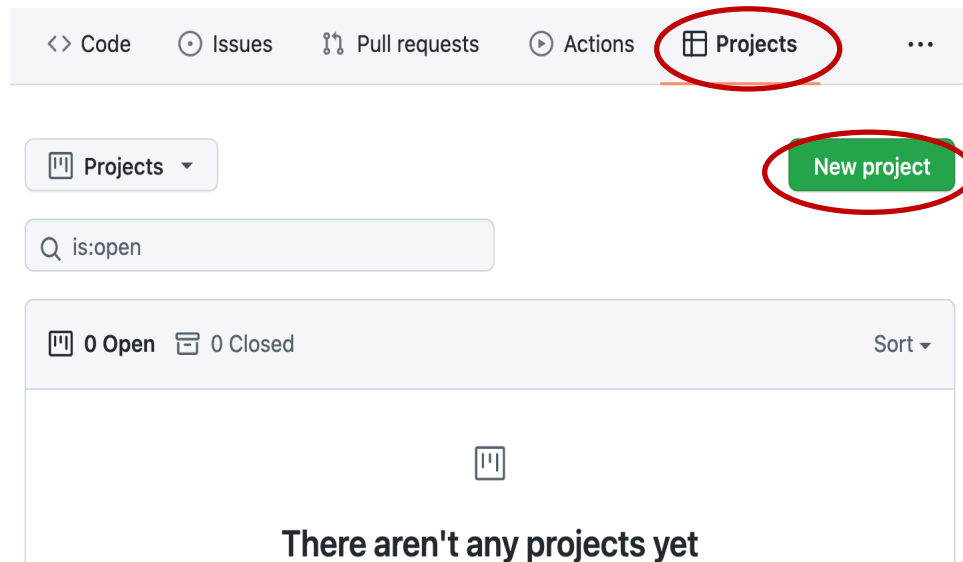
TABLERO KANBAN



● GitHub

Tablero de proyecto

- Podemos crear un tablero KanBan.
- Muy Flexible. Podemos hacer un tablero muy simple (y de gestión manual)
- GitHub permite automatizar el flujo de trabajo a través de la gestión de Issues y pull requests.
- Podemos crear **diferentes** tableros (documentación, backend,...)



Create a new project

Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name

Project board name

Description (optional)

Project template

Save yourself time with a pre-configured project board template

Template: None

Create project

Opciones de automatización



Issues

Puede entenderse como un post-it en el tablero **Kanban**.




👉 Cualquier problema o tarea que nos encontremos con el proyecto debe tener un Issue.

Utilizan **Markdown**.

Escribir solo lo necesario para que se entienda el problema. Se puede acompañar imágenes o hiperenlaces en la explicación.

Es el eje la gestión de un proyecto. Cuantos más Issues, mayor información acerca del estado del proyecto.

Un Issue puede relacionarse con otras entidades del repositorio de GitHub. Esta interacción potencia enormemente la información.

- Responsables 
- Etiquetas 
- Milestones 



SOLO SE EVALUA LA COMUNICACIÓN QUE ESTÉ EN EL REPOSITORIO DE GITHUB



Investigate to what extent HTTPS should be mandatory #1091

[New issue](#)

Estado Issue

Open

RubenVerborgh opened this issue on 9 Dec 2021 · 2 comments

Explicación
del
problema

Comentarios



RubenVerborgh commented
on 9 Dec 2021

Member



In some previous testing, I have come across preliminary evidence that some Solid-related functionality only works over HTTPS. In particular, when running Mashlib as the on-server UI, authentication seems to break because the server is not running over HTTPS.

Whereas this is actually a question for the bigger Solid ecosystem, we can test some assumptions on CSS and turn them into recommendations.

If the answer is that some functionality only works over HTTPS, then we might want to make CSS start over HTTPS out of the box (e.g., by auto-generating `localhost` certificates etc.).

Assignees



RubenVerborgh

Asignado a



Labels



task

Etiquetas



Projects

None yet

Tablero de Proyecto

Milestone

No milestone

Milestones



Linked pull requests

Successfully merging a pull request may close this issue.

Pull request
linkado

None yet

Log del Issue



RubenVerborgh added the



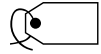
task

label on 9 Dec 2021

GitHub

ETIQUETAS

- Sirven para catalogar los issues.
- GitHub provee unas etiquetas genéricas.
- Podéis crear las etiquetas que queráis.(backend, frontend, bbdd, jerarquías)
- Cuanto más organizado mejor.
- Cuidado con no pasarse. Si se crean demasiadas pierden su función que es categorizar.
- GitHub permite personalizar el color y utilizar emojis para dar estilo y ser de ayuda visual 📁.



<> Code Issues 105 Pull requests 13 Discussions Actions Projects 1

Labels Milestones Search all labels

32 labels

worker threads

Making the server multithreaded

5

bug

Something isn't working

17

dependencies

Pull requests that update a dependency file

developer experience

5

difficulty:high

2

difficulty:low

13

difficulty:medium

9

Nombre
etiqueta

Pequeña descripción

Numero de Issues
etiquetados

Uso común en jerarquías
dificultad o prioridad
suelen tener distintos
niveles

Uso de FAQ de arquisoft para preguntas

<https://github.com/Arquisoft/faq/issues>

Compartir preguntas del curso

- Se permite añadir cualquier issue que describa alguna pregunta relacionada con el curso en inglés o español
 - Cualquiera puede contribuir respondiendo la pregunta o añadiendo comentarios
 - Los que contribuyan deben seguir un código de conducta que respite las consideraciones éticas de un curso de la Universidad de Oviedo
- Los profesores podrán borrar cualquier issue o comentario que consideren inapropiado para el curso

Doc adicional

- Diapositivas de Labra sobre [Git](#).
- [Guía rápida](#) de Pablo Gonzalez sobre Git
- Una pequeña [introducción a git](#) de Hugo Lebreo
- Esta [guía](#) esta muy chula es un buen sitio para aprender y buscar comandos frecuentes, el diseño es muy limpio. Si estas buscando cosas como por ejemplo, sincronizar tu repositorio local con el de GitHub puede ayudarte un buen lugar 👍.
- Increíble la web learngitbranching.js.org. Explica Git de manera interactiva, con animaciones y lo más importante paso a paso
★ ★ ★ ★ ★ .

Doc adicional

- Recomendable todo contenido de [Miguel Angel Durán](#). Uno de los mejores canales sobre Git y desarrollo web.
- Libro de Miguel Angel [Aprendiendo Git](#).

Youtube

- [Explicación conceptos de Git en 15 minutos](#)
- [Curso/tutorial](#) desde 0
- [GitHub vs GitLab](#) Este video explica la importancia actual de estas empresas.