

Repositorios de la Asignatura e Introducción a Git y GitHub

Estructuras de Datos

Convocatoria de Asignaturas en Abierto 2025-26

Universidad Rey Juan Carlos

Profesores:

Sergio Cavero

Salvador Sanchez

©2025 Autores Sergio Cavero Díaz y Salvador Sánchez Alonso. Algunos derechos reservados.

Este documento se distribuye bajo la licencia “Atribución-CompartirIgual 4.0 Internacional” de Creative Commons, disponible en:

<https://creativecommons.org/licenses/by-sa/4.0/>

Índice general

1. Repositorios de código de la asignatura	5
2. Introducción a Git e Instalación	7
2.1. Instalación de Git	7
2.1.1. Verificación Previa	7
2.1.2. Instalación en Windows	7
2.1.3. Instalación en Mac	7
2.1.4. Instalación en Linux	8
3. Inicialización de Repositorios: git init	9
3.1. ¿Qué hace git init?	9
3.2. Cómo usarlo	9
3.2.1. Escenario Común	9
4. Clonado de Repositorios: git clone	11
4.1. ¿Qué hace git clone?	11
4.2. Uso Básico	11
4.3. Opciones Avanzadas	11
5. Comprobación de Estado: git status	13
5.1. Importancia de git status	13
6. Preparación de Archivos: git add	15
6.1. El Área de Preparación	15
6.2. Comandos Comunes	15
7. Confirmación de Cambios: git commit	17
7.1. Funcionamiento	17
7.2. Buenas Prácticas	17
8. Gestión de Remotos: git remote	19
8.1. Origin y Upstream	19
8.2. Comandos Útiles	19
9. Sincronización: git pull y git push	21
9.1. git pull	21
9.2. git push	21

10.Dinámica de la Asignatura y Releases	23
10.1. Metodología de Trabajo	23
10.1.1. El Repositorio Inicial	23
10.1.2. Clonado del Repositorio	23
10.2. El Ciclo de Vida: Problema vs. Solución	23
10.2.1. Publicación de las Soluciones	24
10.2.2. El Rol de la “Release”	24
10.3. Descarga y Uso de Releases	24
10.3.1. Cómo acceder a una Release en GitHub	24
10.3.2. Descarga de Activos (Assets)	24

Capítulo 1

Repositorios de código de la asignatura

A continuación se presentan los enlaces a los repositorios de las prácticas de la asignatura. Estos repositorios servirán como plantillas para la realización de las actividades propuestas.

- **Práctica 1 - Punteros:** <https://github.com/TeachingHub/ED-Lab01-Punteros-Template>
- **Práctica 2 - TAD:** <https://github.com/TeachingHub/ED-Lab02-TAD-Template>
- **Práctica 3 - Pilas:** <https://github.com/TeachingHub/ED-Lab03-Pilas-Template>
- **Práctica 4 - Colas:** <https://github.com/TeachingHub/ED-Lab04-Colas-Template>
- **Práctica 5 - Listas:** <https://github.com/TeachingHub/ED-Lab05-Listas-Template>
- **Práctica 6 - Conjuntos:** <https://github.com/TeachingHub/ED-Lab06-Conjuntos-Template>
- **Práctica 7 - Árboles:** <https://github.com/TeachingHub/ED-Lab07-Arboles-Template>
- **Práctica 8 - Grafos:** <https://github.com/TeachingHub/ED-Lab08-Grafos-Template>

Capítulo 2

Introducción a Git e Instalación

Git es un sistema de control de versiones distribuido esencial para el desarrollo de software moderno. Este capítulo cubre cómo tenerlo listo en tu máquina.

2.1. Instalación de Git

Git puede instalarse en los sistemas operativos más comunes como Windows, Mac y Linux.

2.1.1. Verificación Previa

Para ver si ya tienes Git instalado, abre tu terminal (o “Git Bash” en Windows) y escribe:

```
git version
```

Si el comando es desconocido, necesitas instalarlo.

2.1.2. Instalación en Windows

1. Navega al instalador más reciente de Git para Windows en git-scm.com.
2. Una vez iniciado el instalador, sigue las instrucciones del asistente “Git Setup” hasta completar la instalación.
3. Abre el símbolo del sistema o Git Bash y verifica con `git version`.

2.1.3. Instalación en Mac

La mayoría de las versiones de macOS ya tienen Git instalado. Si no es así:

- **Mediante instalador:** Descarga el instalador de macOS Git desde git-scm.com.
- **Mediante Homebrew:** Si tienes Homebrew, ejecuta:

```
brew install git
```

2.1.4. Instalación en Linux

Puedes usar el gestor de paquetes de tu distribución.

- **Debian/Ubuntu:** `sudo apt-get install git-all`
- **Fedora:** `sudo dnf install git-all`

Capítulo 3

Inicialización de Repositorios: git init

El comando `git init` transforma cualquier directorio en un repositorio Git.

3.1. ¿Qué hace git init?

Es una forma de empezar un proyecto nuevo con Git. Al ejecutarlo, Git crea un directorio oculto llamado `.git`. Este directorio almacena todos los objetos y referencias que Git utiliza y crea como parte del historial de tu proyecto.

3.2. Cómo usarlo

```
git init
```

Este comando transforma el directorio actual en un repositorio Git.

3.2.1. Escenario Común

Si tienes un proyecto localmente pero aún no tiene Git:

1. Ejecuta `git init` dentro de la carpeta del proyecto.
2. Crea un repositorio remoto en GitHub.
3. Añade la URL remota: `git remote add origin <URL>`.
4. Prepara y confirma tus archivos con `git add` y `git commit`.
5. Sube los cambios con `git push -u origin main`.

Capítulo 4

Clonado de Repositorios: git clone

El comando `git clone` se utiliza para crear una copia de un repositorio específico.

4.1. ¿Qué hace git clone?

Cuando clonas un repositorio, no obtienes solo un archivo, sino el repositorio completo: todos los archivos, todas las ramas y todos los commits. Esto te permite trabajar en tu propia máquina con todo el historial del proyecto.

4.2. Uso Básico

```
git clone <url>
```

Esto descarga un repositorio que ya existe en GitHub, incluyendo todos sus archivos, ramas y commits.

Nota sobre la Asignatura

En esta asignatura, usarás este comando frecuentemente para descargar los repositorios de prácticas que te proporcionaremos.

4.3. Opciones Avanzadas

- `git clone -single-branch`: Clona solo una rama específica. Útil si el repositorio es muy grande.
- `git clone [url] [directorio]`: Clona el repositorio en una carpeta específica en lugar de crear una con el nombre del repo.

Capítulo 5

Comprobación de Estado: `git status`

`git status` muestra el estado actual de tu directorio de trabajo y del área de preparación (staging area).

5.1. Importancia de `git status`

Ante la duda, ejecuta `git status`. Siempre es una buena idea. Este comando solo muestra información, no modifica tus cambios.

Te informará sobre:

- En qué rama estás (dónde apunta HEAD).
- Si tienes archivos modificados no confirmados.
- Si los archivos están preparados (staged) o no.
- Si tu rama local está por detrás o por delante de la remota.

Capítulo 6

Preparación de Archivos: `git add`

El comando `git add` añade archivos nuevos o modificados de tu directorio de trabajo al área de preparación (staging area).

6.1. El Área de Preparación

`git add` es un paso crucial que te permite elegir qué vas a confirmar. Los commits deben ser unidades lógicas y atómicas de cambio. `git add` te permite dar forma a tus commits sistemáticamente.

6.2. Comandos Comunes

- `git add <archivo>`: Prepara un archivo específico.
- `git add .`: Prepara todos los archivos en el directorio actual (recursivamente).
- `git add -A`: Prepara todos los archivos en todo el repositorio (nuevos, modificados y eliminados).

Capítulo 7

Confirmación de Cambios: git commit

`git commit` crea una confirmación, que es como una instantánea (snapshot) de tu repositorio en un momento específico.

7.1. Funcionamiento

Los commits cuentan la historia de tu repositorio. Incluyen metadatos como el autor, la fecha y un mensaje.

```
git commit -m "mensaje descriptivo"
```

Esto inicia el proceso de commit e incluye el mensaje al mismo tiempo.

7.2. Buenas Prácticas

Los mensajes de commit deben ser cortos, descriptivos y usar el tiempo presente imperativo (ej: “añadir función de login” en lugar de “añadida función de login”).

Capítulo 8

Gestión de Remotos: git remote

`git remote` gestiona el conjunto de repositorios remotos que estás rastreando con tu repositorio local.

8.1. Origin y Upstream

- **origin:** Es el nombre amigable por defecto para la URL del repositorio remoto desde el que clonaste.
- **upstream:** Comúnmente usado para referirse al repositorio original cuando has hecho un “fork”.

8.2. Comandos Útiles

- `git remote -v`: Lista los remotos actuales.
- `git remote add [nombre] [url]`: Añade un nuevo remoto.

Capítulo 9

Sincronización: git pull y git push

9.1. git pull

Este comando actualiza tu rama de trabajo local actual con todos los nuevos commits de la rama remota correspondiente. Es una combinación de `git fetch` y `git merge`.

```
git pull
```

Es buena práctica ejecutar `git pull` antes de empezar a trabajar para evitar conflictos.

9.2. git push

Sube todos los commits de tu rama local a la rama remota correspondiente.

```
git push
```

Si es la primera vez que subes una rama, usa `git push -u origin <rama>` para establecer la relación de rastreo.

Capítulo 10

Dinámica de la Asignatura y Releases

Este capítulo detalla cómo funcionará el flujo de trabajo en clase, utilizando repositorios y el sistema de Releases de GitHub.

10.1. Metodología de Trabajo

Para cada práctica o proyecto de la asignatura, seguiremos un ciclo estricto diseñado para simular un entorno de desarrollo profesional. Este sistema facilita tanto la distribución del código base inicial como la posterior publicación de las soluciones oficiales.

10.1.1. El Repositorio Inicial

El profesorado proporcionará la URL de un repositorio en GitHub para cada lección o práctica. Inicialmente, este repositorio contendrá:

- El archivo `README.md` con el enunciado detallado de la práctica.
- El documento de la práctica en formato PDF (si aplica).
- Los archivos de código fuente en Pascal necesarios para comenzar (plantillas, unidades básicas, etc.).

10.1.2. Clonado del Repositorio

El primer paso es clonar este repositorio en tu máquina local. Esto crea una vinculación con el repositorio remoto que nos permitirá descargar actualizaciones futuras. Usa el comando visto en el Capítulo 3:

```
git clone https://github.com/asignatura/practica-x.git
```

Al clonarlo, tendrás el enunciado y la estructura de carpetas en tu ordenador.

10.2. El Ciclo de Vida: Problema vs. Solución

Una de las dudas más frecuentes al usar repositorios compartidos en clase es la gestión de las soluciones.

10.2.1. Publicación de las Soluciones

El profesor actualizará el repositorio con las soluciones de los ejercicios. Esto puede ocurrir de dos formas:

1. **Parcialmente:** Se sube la solución de un ejercicio concreto al finalizar la clase si se ha resuelto en vivo.
2. **Totalmente:** Se publican todos los ejercicios resueltos la semana siguiente a la publicación del enunciado.

El Dilema del Estudiante

Si el repositorio se actualiza con las soluciones, ¿cómo puedo acceder a los ejercicios “limpios” (sin resolver) si llego tarde o quiero repasar desde cero?

Aquí es donde entra en juego el concepto de Release.

10.2.2. El Rol de la “Release”

Para garantizar que siempre tengas acceso a la versión original de la práctica (el “enunciado” sin código resuelto), el profesor creará una Release etiquetada específicamente como *versión sin soluciones*.

- Una **Release** es una versión congelada del software en un punto específico del tiempo.
- Se basa en **etiquetas (tags)** de Git. Aunque el código principal (rama `main`) avance e incluya las soluciones, la Release se queda “quieta” en el momento inicial.

10.3. Descarga y Uso de Releases

Cuando vayas a comenzar una práctica, independientemente de si el profesor ya ha subido las soluciones al repositorio principal, debes descargar la Release inicial para intentar resolvérla por tu cuenta.

10.3.1. Cómo acceder a una Release en GitHub

1. Navega a la página principal del repositorio en GitHub. 2. Busca en la barra lateral derecha la sección llamada **Releases**. 3. Haz clic sobre ella (o sobre la etiqueta de la última versión, por ejemplo, `v1.0`).

4. Verás una pantalla con el título de la versión (ej: “Enunciado Inicial - Ejercicios sin soluciones”).

10.3.2. Descarga de Activos (Assets)

En la parte inferior de la sección de la Release, encontrarás el apartado Assets. GitHub genera automáticamente archivos comprimidos con el código exacto de esa versión.

- **Source code (zip):** Es el archivo que generalmente descargas si trabajas en Windows/Mac.
- **Source code (tar.gz):** Común en entornos Linux.

- **Binarios extra:** A veces el profesor puede adjuntar ejecutables o librerías compiladas aquí.

Lo siguiente que deberás hacer es descargar la carpeta comprimida deseada, descomprimir y abrir la carpeta resultante con el entorno de desarollo.