Instructivo para entregas

Algoritmos y Estructuras de Datos, DC, UBA.

Segundo cuatrimestre de 2023

Este documento explica cómo será el mecanismo para hacer las entregas obligatorias del laboratorio de la materia.

Formato de entrega

Todas las entregas serán digitales.

- Las entregas que incluyan informes deben incluir una carátula indicando **nombre, apellido y libreta universitaria (o documento)** de la persona o personas que entregan. En el caso de los TPs grupales, se debe incluir el **alias del grupo**, además de los datos de todes les integrantes del grupo.
- Todos los ejercicios que consistan en entregar **documentación** como especificaciones de TADs, TPs de especificación, TPs de diseño, etc., deberán entregarse **en formato** `pdf`. Sugerimos usar LaTeX para generar el documento, pero pueden usar cualquier otra herramienta que permita exportar el documento en pdf (ej. herramientas "ofimáticas" como Word, LibreOffice o Google Docs). Esperamos que la solución sea prolija y legible, pero no se pretende que sea estéticamente agradable.
- Para los ejercicios que consistan en entregar **código** como talleres del laboratorio y TPs de implementación, deberá entregarse el código fuente en Java (archivos `*.java`) que se indiquen en cada entrega.

Medio de entrega

Para la entrega utilizaremos repositorios de `git` en GitLab (`https://gitlab.com`). Abajo se detallan los pasos para crear los repositorios para entregas individuales (**Paso 1**) y grupales (**Paso 2**). Finalmente se detalla el formulario a completar para dar registro del repositorio individual, el repositorio grupal y qué integrates pertenecen al grupo.

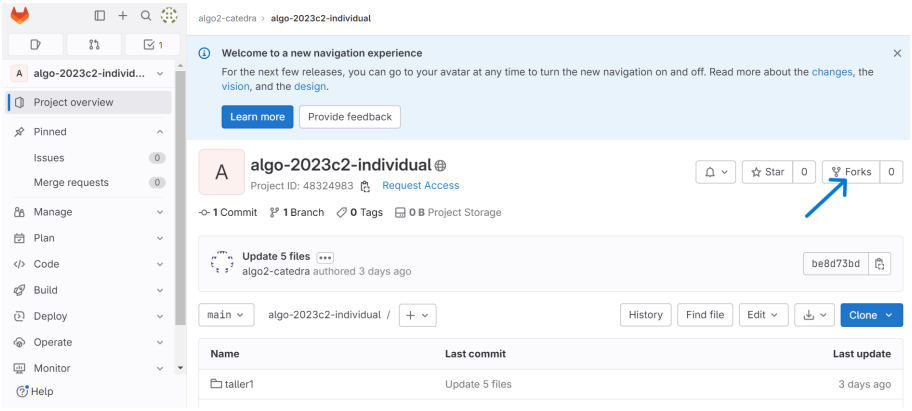
Paso 1: Crear el repositorio para entregas individuales

Para las entregas **individuales** (ejercicios obligatorios de la práctica y talleres). Supondremos que cada estudiante tiene creada una cuenta en GitLab.

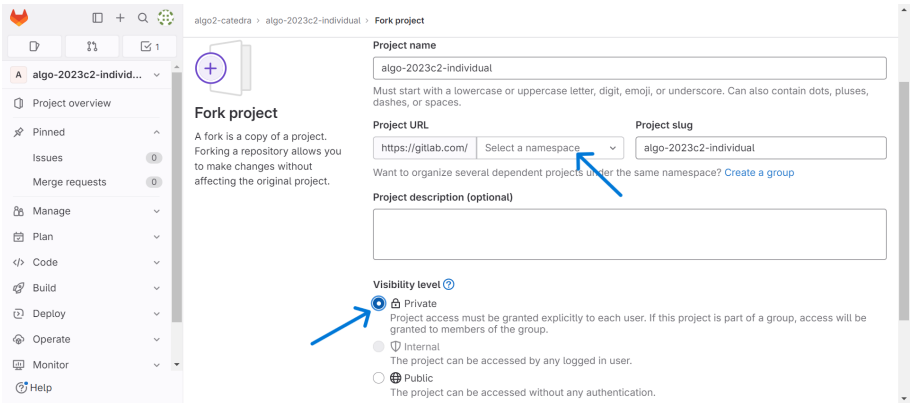
- **Paso 1.1:** Una vez loggeada la cuenta de GitLab, ingresar desde el navegador al repositorio de la cátedra que contiene un esqueleto del repositorio:

  `https://gitlab.com/algo2-catedra/algo-2023c2-individual`

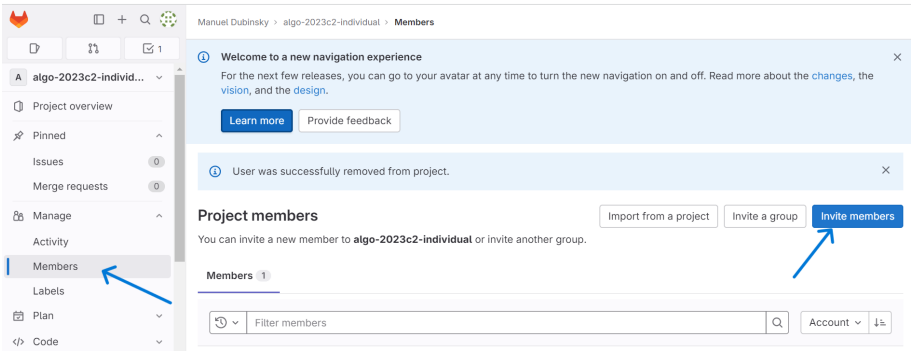
  y crear un **fork** (es decir, una "copia"):



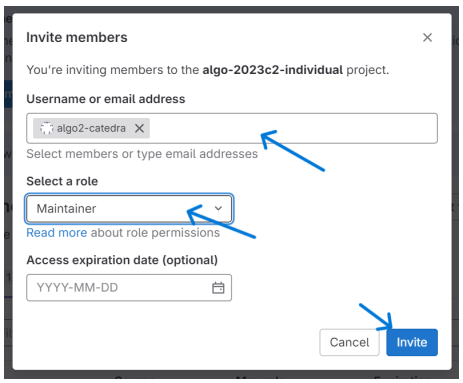
- **Paso 1.2:** Setear el namespace (usuario personal) y cambiar la visibilidad del repositorio a **privado**.



- **Paso 1.3:** Agregar al usuario "`algo2-catedra`" como miembro del proyecto.
  - Paso 1.3.1: Invitar a un usuario



  - Paso 1.3.2: Completar los datos usuario/rol con "**algo2-catedra / Maintainer**".



Hecho esto, el repositorio individual está listo para usar, por ejemplo:

```
git clone https://gitlab.com/fulano/algo-2023c2-individual.git
```

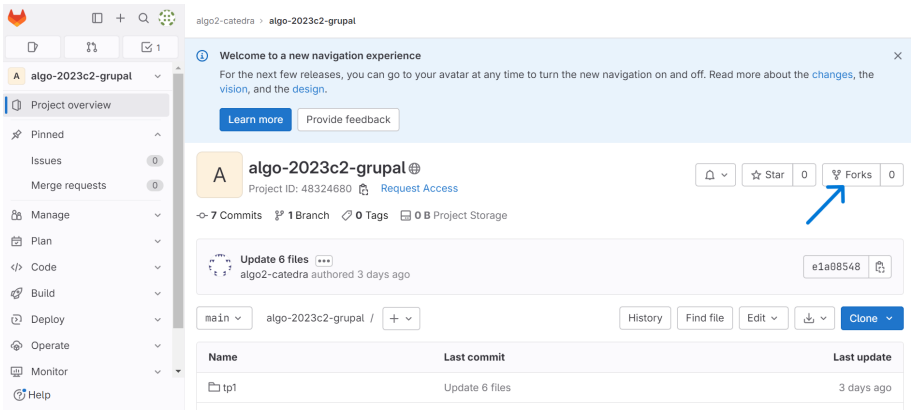
Paso 2: Crear el repositorio para entregas grupales

Para las entregas **grupales** (TPs), supondremos que cada estudiante ya tiene creada una cuenta en GitLab.

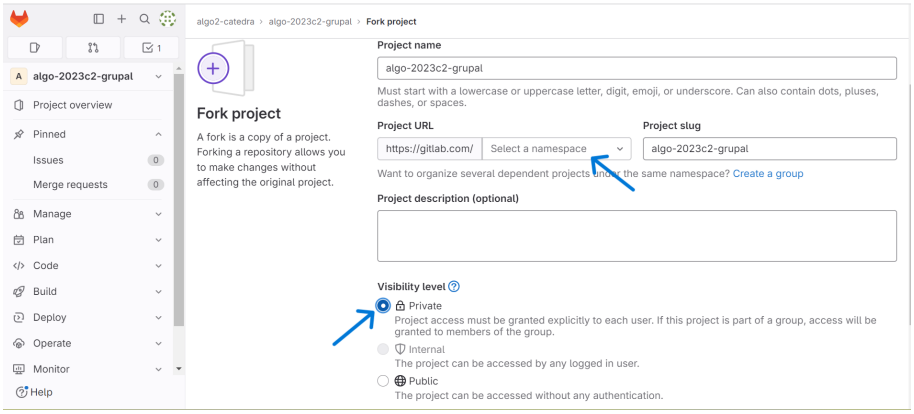
- **Paso 2.1:** Un **único integrante del grupo** deberá, estando loggeado en GitLab, ingresar desde el navegador al repositorio de la cátedra que contiene un esqueleto del repositorio:

  `https://gitlab.com/algo2-catedra/algo-2023c2-grupal`

  y crear un **fork** (es decir, una "copia"):



- **Paso 2.2:** Setear el namespace (usuario personal) y cambiar la visibilidad del repositorio a **privado**.



- **Paso 2.3:** Agregar al usuario "`algo2-catedra`" como miembro del proyecto. Importante: asegurarse de darle a dicho usuario el rol "**Maintainer**". Agregar también los restantes integrantes del grupo como miembros del proyecto, con rol "**Maintainer**".
- **Paso 2.4:** Elegir un **alias** para el grupo. El mismo se usará para completar el formulario en el Paso 3.

Paso 3:

Completar los siguientes formularios: `https://forms.gle/xv7tEZewuPbESvRE6` y `https://forms.gle/cu37EAbA5322RkNNA`.
Nota: cada integrante del grupo debe completar este formulario de manera independiente, pero todos deben tener el **mismo alias** y referir al **mismo repositorio** grupal.

. . . . . .

Estructura de carpetas

Dentro del repositorio **individual** se deberá respetar la siguiente **estructura de carpetas**, todo en minúsculas y sin tildes ni espacios:

- `taller1/` : aquí se ubica el primer **taller**.
  En este directorio deben encontrarse el/los archivo/s `*.java` correspondientes al taller.

- `taller2/` : directorio con las entregas de la guía 2, con estructura similar.
- `taller3/` : directorio con las entregas de la guía 3, con estructura similar.
- `taller4/` : directorio con las entregas de la guía 4, con estructura similar.
- `taller5/` : directorio con las entregas de la guía 5, con estructura similar.

Dentro del repositorio **grupal** se deberá respetar la siguiente **estructura de carpetas**, todo en minúsculas y sin tildes ni espacios:

- `tp1/` : directorio con la entrega del TP 1.
  Aquí se ubica el primer **trabajo práctico**.
  En este directorio deben encontrarse todos los archivos correspondientes al TP.
- `tp2/` : directorio con la entrega del TP 2, con estructura similar.

Guía absolutamente minimal de `git`

Presentamos una guía mínima de comandos para sobrevivir usando `git`, destinada a quienes nunca lo hayan usado. En caso de que no tengan experiencia usando `git`, también pueden mirar el video de la clase L03 (parte 4):

`https://www.youtube.com/watch?v=jLRvz6wonb8`

Aclaramos que esta mini-guía no es técnicamente precisa y que la herramienta `git` es mucho más compleja de lo que se describe abajo. En algunas semanas tendremos una clase especialmente dedicada a Git.

1. **Hacer una copia del repositorio.** Con el siguiente comando se crea una copia local del repositorio remoto. Por ejemplo si `fulano` quiere trabajar sobre el repositorio para el TP creado por su compañera `mengana`, debe hacer:

   ```
   git clone https://gitlab.com/mengana/algo-2023c2-grupal.git
   ```

   esto crea una carpeta `algo-2023c2-grupal/` dentro de la cual se encuentra el contenido del repositorio. Este comando se debería hacer por una única vez al principio.

2. **Actualizar la copia local.** Si el repositorio remoto sufre modificaciones, `fulano` puede actualizar su copia local con el siguiente comando, ubicándose dentro de la carpeta que contiene la copia local:

   ```
   git pull --rebase
   ```

3. **Agregar un archivo (o modificación).** Cada vez que `fulano` agrega un nuevo archivo (por ejemplo `tp1.tex`) o hace una modificación en el archivo `tp1.tex`, puede usar el siguiente comando para registrar la modificación:

   ```
   git add tp1.tex
   ```

   Notar que el comando `add` solamente tiene efecto sobre el repositorio local (no modifica el repositorio remoto).

4. **Registrar localmente un conjunto de modificaciones.** Una vez que se hayan agregado uno o varios archivos (o modificaciones), se puede registrar ese conjunto de cambios localmente a través del comando `commit`, incluyendo un mensaje que indica por ejemplo:

   ```
   git commit -m "Agregué observador coordenada en la especificación del TAD Robot."
   ```

   Los cambios que se hacen con `add` son temporales, y se vuelven definitivos a través del comando `commit`. Notar, sin embargo, que el comando `commit` solamente tiene efecto sobre el repositorio local (no modifica el repositorio remoto).

5. **Enviar los cambios al repositorio remoto.** El siguiente comando envía las modificaciones locales al repositorio remoto:

   ```
   git push
   ```

   Esto envía todos los “commits” locales.