

Introducción a la programación

Recomendaciones a tener en cuenta antes de enviar el TP

Creando repositorio

A continuamos veremos una opción para agregar los archivos del TP a un nuevo repositorio. Usaremos como ejemplo git.exactas pero son los mismos pasos para gitlab.com:

- ▶ Crear un repositorio (o proyecto) en `git.exactas.uba.ar` con nombre `<nombre-repo>`
Esto crea un `<url-repo>` en `https://git.exactas.uba.ar/<usuario>/<nombre-repo>.git`
Para este ejemplo vamos a usar `<usuario>=docentes.ip` y `<nombre-repo>=1c2024tphaskell`
- ▶ Abrir la terminal en alguna carpeta y clonar el repositorio con el comando `git clone <url-repo>`

```
~$ git clone https://git.exactas.uba.ar/docentes.ip/1c2024-tphaskell.git
```
- ▶ Ir a la carpeta `<nombre-repo>` y mover los archivos del TP
- ▶ Importante:
 - ▶ El archivo con la solución debe llamarse `Solucion.hs`
 - ▶ No debe haber carpetas dentro del repositorio

Agregando archivos al repositorio

Supongamos que movimos los archivos del template descargados del campus al la carpeta <nombre-repo>

- En la terminal, parados en la carpeta <nombre-repo>, ejecutamos `git status`:

```
On branch master

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
Solucion.hs
test-catedra.hs
{nombre_grupo}.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- En este momento los archivos no están “marcados” para rastrear los cambios. Para “marcarlos”¹ a todos, ejecutamos ‘`git add .`’

¹Debemos moverlos a lo que se conoce como *staging area*

Agregando archivos al repositorio


- Confirmamos los cambios con `git commit -am <mensaje>` ²

```
~/1c2024-tphaskell$ git commit -am "subimos template original"
[master (root-commit) 5b88c50] subimos template original
3 files changed, 224 insertions(+)
create mode 100644 Solucion.hs
create mode 100644 test-catedra.hs
create mode 100644 {nombre_grupo}.txt
```

- Todavía falta subirlo al repositorio remoto para que el resto del grupo pueda acceder a los cambios. Para esto ejecutamos `git push`

```
~/1c2024-tphaskell$ git push
Username for 'https://git.exactas.uba.ar': docentes.ip
Password for 'https://docentes.ip@git.exactas.uba.ar':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 2.77 KiB | 2.77 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://git.exactas.uba.ar/docentes.ip/1c2024-tphaskell.git
* [new branch]      master -> master
```

- Podemos ir a la interfaz web y vamos a ver los archivos recién subidos.

²`commit -am`: La opción `-m` es para indicar el mensaje a continuación, y la opción `-a` es para confirmar todos los archivos en la staging area. 

Trabajando en grupo en el TP

- ▶ Recuerden agregar a cada uno de los integrantes del grupo con rol developer.
- ▶ Para agregar colaboradores en git.exactas, ver este video:
<https://www.youtube.com/watch?v=bI12oUvEcoI>.
- ▶ Para agregar colaboradores en gitlab.com, ver este video:
<https://youtu.be/qIcpDdk6-SQ>.

Trabajando en grupo en el TP

Luego de muchos commits y push entre todos los integrantes del grupo, estamos en condiciones de hacer la entrega del tp. Para esto necesitamos la url del repositorio y el último commit.

- ▶ URL: algo similar a
`https://git.exactas.uba.ar/<usuario>/<nombre-repo>.git`
- ▶ commit: es lo que identifica unívocamente cada cambio confirmado. Podemos verlo ejecutando `git log`

```
commit 4c4311256c2cb21ebae1a65adf15a5a73ebf503b (HEAD -> master, origin/master)
Author: Docentes IP <j.godoy277@gmail.com>
Date:   Fri May 17 20:30:29 2024 -0300
```

TP finalizado

//...Listado de todos los commits realizados, ordenados de más reciente a más antiguo ...

```
commit 5b88c5050b028744e8d91c21e64f3e76fb915d93
Author: Javi Godoy <j.godoy277@gmail.com>
Date:   Fri May 17 20:14:02 2024 -0300
```

subimos template original

- ▶ Por ejemplo, el último commit aparece primero y es
'4c4311256c2cb21ebae1a65adf15a5a73ebf503b'

Revisando que el TP compila y todos los test pasan

Luego de finalizar el TP y completar el archivo de texto a entregar, para validar que todo está bien, se deben realizar los siguientes pasos (usando su propio url y commit):

- ▶ En una nueva carpeta, abrir una terminal y clonar el repositorio:

```
~$ git clone https://git.exactas.uba.ar/docentes.ip/1c2024-tphaskell.git
```

- ▶ Moverse a la carpeta clonada: `cd 1c2024-tphaskell`

- ▶ Pararse sobre el commit indicado con el comando
`git checkout <commit>`

```
~$ git checkout 4c4311256c2cb21ebae1a65adf15a5a73ebf503b
```

- ▶ Ejecutar `ghci`, cargar el archivo `test-catedra.hs` y ejecutarlo. Revisar que compila y pasa todos los tests.

```
~/1c2024-tphaskell$ ghci
Loaded package environment from /home/jgodoy/.ghc/x86_64-linux-9.2.8/environments/default
GHCi, version 9.2.8: https://www.haskell.org/ghc/  :? for help
ghci> :l test-catedra.hs
[1 of 2] Compiling Solucion          ( Solucion.hs, interpreted )
[2 of 2] Compiling Main              ( test-catedra.hs, interpreted )
Ok, two modules loaded.
ghci> runCatedraTests
Cases: 15  Tried: 15  Errors: 0  Failures: 0
Counts {cases = 15, tried = 15, errors = 0, failures = 0}
ghci>
```

Revisando que el TP compila y todos los test pasan

- ▶ Luego, ejecutar sus propios tests (pueden estar en el mismo archivo o, mejor, en un nuevo archivo, por ejemplo: test-propios.hs)
- ▶ Recordar que:
 - ▶ El archivo con la solución debe llamarse `Solucion.hs`
 - ▶ No debe haber carpetas dentro del repositorio
- ▶ Si realizando estos pasos el proyecto no compila, el TP está desaprobado.
- ▶ Si luego de realizar estos pasos el proyecto no compila o algún test no pasa, corregirlo, subirlo al repositorio y actualizar el commit.
REPETIR TODOS LOS PASOS NUEVAMENTE CON EL NUEVO COMMIT.
- ▶ Pueden probar los pasos anteriores ya que el repositorio <https://git.exactas.uba.ar/docentes.ip/1c2024-tphaskell.git> es público.

- ▶ En cuanto a los tests en test-catedra.hs, recordar que pueden usar las funciones presentadas en la clase de HUnit. Por ejemplo, si un test falla porque el resultado esperado es '33.333336' y el resultado obtenido es '33.333337' ¿Qué función deberían usar?