# Trabajo Práctico: Commits y branches en git

## **Enunciado**

Desarrollar en c++ una librería que permita crear Branches y commit, la misma debe estar basada en el modelo de datos del conocido programa git.

Una librería es un conjunto de módulos que cooperan dando soporte para un fin común. El enfoque que utilizaran para construir la librería es el de TDA.

La construcción de esta librería se realizará en 4 entregas incrementales.

**Motivacion**

Git es una de las tantas herramientas conocidas cómo sistemas de control de versiones y sin dudas es la más utilizada.

En clase pesentaremos lo siguiente a modo de presentacion y motivacion:

* Describiremos en que consisten estas herramientas y sobre todo porque son importantes
* Describiremos el modelo de datos (tipos de datos) presentes en git porque es lo que nos importa desarrollar en el TP

[Video de presentacion](https://www.youtube.com/watch?v=Vz1f_7LCvjQ&t=2s)

**Descripción del Dominio para la primer entrega**

**Commit**

Un commit es un punto en la historia de cambios que tiene la responsabilidad de conocer:

* El autor: La persona que realizo ese cambio
* El tiempo: Fecha y hora de cuando se realizo el cambio
* Un mensaje: El mensaje tiene cómo objetivo describir las razones del autor del cambio producido por el commit.
* El Commit predecesor: Todo commit puede tener un Commit predecesor

Se trata de un tipo de dato ce instancias inmutable

**DateTime**

Es objetivo de la primer entrega implementar el tipo de dato DateTime que representa la fecha y hora. Este tipo de dato sera utilizado para:

* Obtener la marca de tiempo asociado a la creacion de cada Commit

Si bien se podría decidir representar otros formatos de marca de tiempo (cómo [unix epoch time](https://es.wikipedia.org/wiki/Tiempo_Unix)) y almacenarla en estructuras de datos más compactas, obtamos por crear un tipo de dato con la interfaz dada en DateTime.h

Para implementar este TDA puede utilizar librerias auxiliares

Otra alternativa hubiera sido utilizar directamente la libreria de c: [*<ctime>*](https://es.wikipedia.org/wiki/Time.h) sin embargo la interfaz que encontramos en esa libreria no es muy buena para nuestros objetivos por lo que nuestro TDA DateTime sera un [adapter/wrapper](https://es.wikipedia.org/wiki/Adaptador_(patr%C3%B3n_de_dise%C3%B1o)) de *<ctime>*

**Context**

El tipo de dato Context ser utilizado por todos los modulos que quieran obtener la Hora y Fecha actual y el usuario actual.

Usualmente en este tipo de tipo de datos (Context, EnvironmentContext) se suele encontrar informacion del ambiente (environment) que normalmente es el usuario autenticado y autorizado, datos del servidor y la horar y fecha actual del servidor. En nuestro caso

Este TDA debe devolver cómo usuario una unica instancia siempre, lo que se conoce cómo un Singleton.

**User**

En git no se realizan autenticaciones ni autorizaciones de usuario. Existe el concepto de author de un commit que puede ser cualquier usuario. Modelamos un usuario con el objetivo de asociarlo cómo autor de cada commit

**CommitSymbolTable**

Se pide que agregue una coleccion de tipo diccionario, con la implementacion que desee, donde la clave sea el hash code del commit y el valor la instancia de commit.

**Descripción del Dominio para la segunda entrega**

[Publicacion Pendiente]

**Descripción del Dominio para la tercera entrega**

[Publicacion Pendiente]

## **Criterio de aprobación de las entregas**

Para que una entrega esté aprobada debe cumplir con todos los requerimientos y restricciones descriptos en cada entrega, sin ninguna excepción y en la fecha de entrega de la misma.

Cada entrega es una instancia de evaluación y se puede recuperar. La recuperación de una entrega se realiza a la fecha de entrega de la próxima entrega (no se realizan entregas en fechas intermedias)

Cada entrega consta del codigo fuente (SOLAMENTE los archivos .h y .cpp) comprimidos en un archivo (.zip, .rar) y se debe entregar por slack de la siguiente forma: Crear un chat con todos los integrantes del grupo y el docente, adjuntar la entrega en ese chat.

## **Fechas de entrega**

**Primera Entrega: 22 junio**

## **Material de referencia**

[[git-internals](https://github.com/pluralsight/git-internals-pdf/releases)] - Git Internals de Scott Chacon