BASES DE DATOS

Clase 6 – Manejo de repositorios

Profesora: Erika Gutiérrez Beltrán

Tema 2: Bases de datos relacionales

Objetivo

Transformar los modelos conceptuales a lógicos. Construir bases de datos para datos estructurados

Temario

- Utilización de los modelos en el mundo real
- Transformación del modelado a la base de datos
- Clases de entidades
- Arquitectura de los sistemas gestores de Bases de datos, OLTP, OLAP
- Modelo relacional, restricciones de integridad y normalización
- Lenguaje de consulta formal
- Lenguaje de consulta SQL

Logros

Capturar datos desde una pantalla de usuario real y almacenarlos en una base de datos. Manipulación de estos datos.

Resumen del tema 1

Pensemos en un escenario de la vida real. Es tu momento de prácticas, empezaste a trabajar en una aplicación nueva. Desde lo que aprendiste en el tema 1 ¿cómo puedes aportar al desarrollo de la aplicación?

Además, pensemos en que otras tablas podemos agregar al modelo y que otros atributos nos permitirían tener más información importante en nuestra base de datos de redes sociales

Manejo de repositorios

¿Qué es GIT?

Git es un sistema de control de versiones, que permite gestionar proyectos de software de manera distribuida, permitiendo acceder al historial de todos los cambios realizados sobre el código del proyecto.

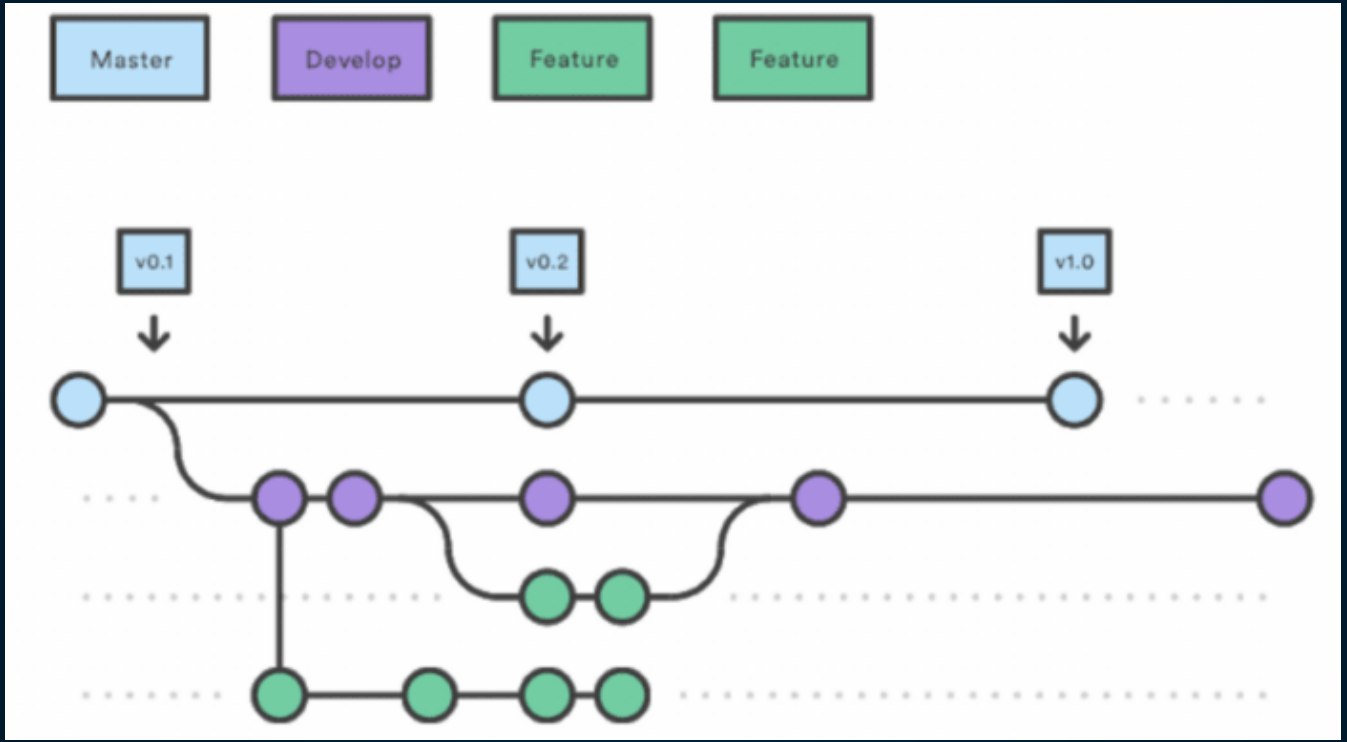
Manejo de repositorios

¿Qué es git-flow?

Es un conjunto de extensiones de git para proporcionar operaciones de repositorios de alto nivel para el modelo de ramificación.

Facilita el manejo de las ramas, el orden del proyecto y la división de los diferentes momentos por los que pasa un proyecto de software, además de desplegar y liberar cambios sobre cada rama según su ambiente.

Git-flow



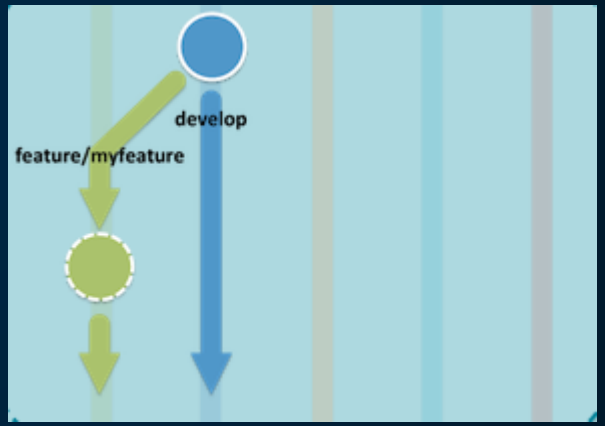
Comandos

Iniciar un proyecto con git-Flow.

1. Descargar el proyecto del repositorio, en este caso, de GITHUB
2. Ejecutar el comando **git flow init** en la consola o terminal

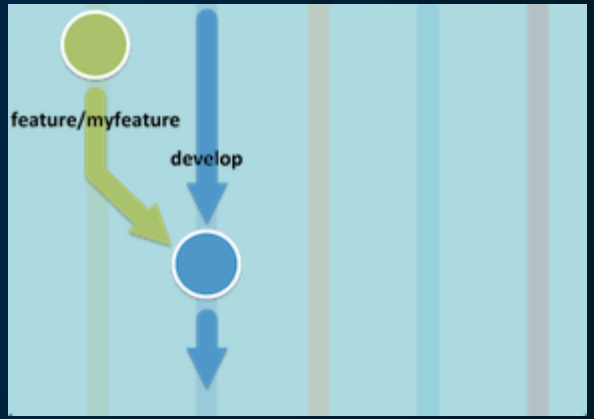
```
git flow init
```

3. Ejecuta el comando **git flow feature start MYFEATURE**
Si quiere sacar una copia del proyecto desde la rama de desarrollo

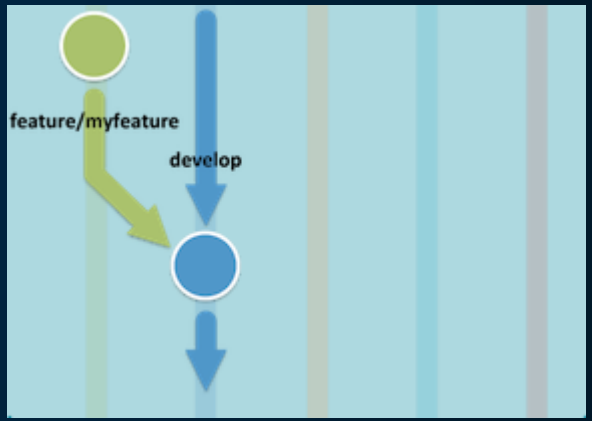


4. Para finalizar un feature ejecutar el comando **git flow feature finish MYFEATURE**

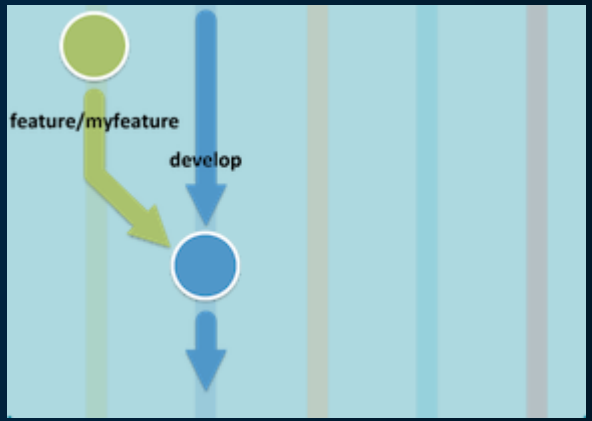
Se debe tener en cuenta que un feature es creado siempre que vamos a agregar funcionalidades nuevas a nuestro desarrollo. A nivel industria el feature se utiliza uno por cada sprint, o cada que sale un nuevo release



5. Para compartir tu rama con otros desarrolladores y programar en conjunto utiliza el comando **git flow feature publish MYFEATURE**

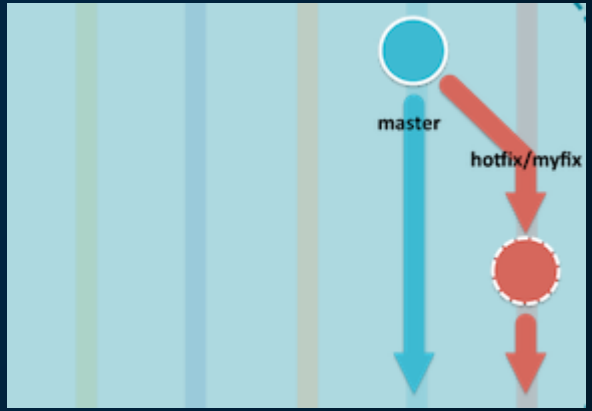


6. Para obtener los cambios creados por otro usuario sobre la rama que hemos creado usar el comando **git flow feature pull origin MYFEATURE**



7. Los hotfix son utilizados para resolver problemas presentados en el código que ocurren sobre producción
git flow hotfix start VERSION name

Importante hacer seguimiento de las versiones

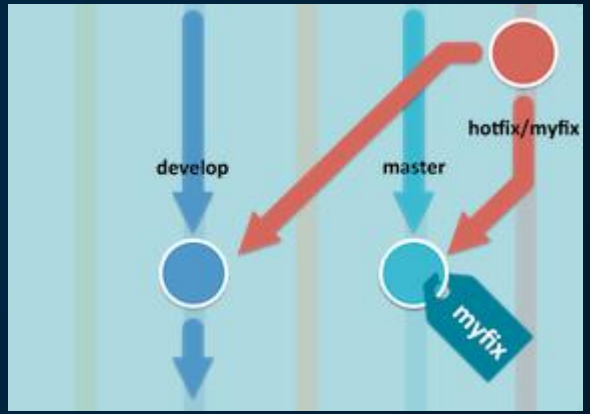


Comandos

8. Cuando se finaliza un hotifx el ajuste se libera tanto en la rama principal (master) como en desarrollo (develop)

git flow hotfix finish VERSION

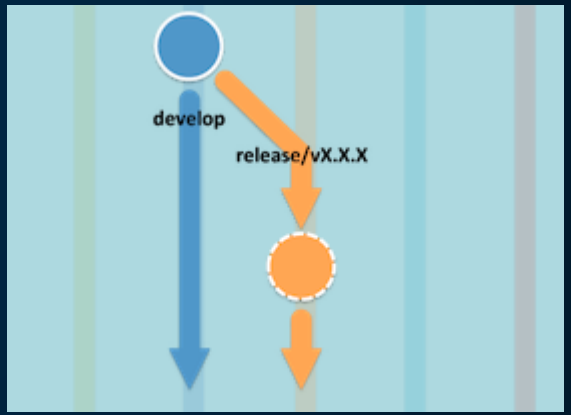
El hotfix también pude ser publicado con el comando **git flow publish** y descargar cambios con **git flow pull**



Comandos

9. El release permite crear una nueva versión de la aplicación y llevar control de las características desplegadas

git flow release start RELEASE [BASE]



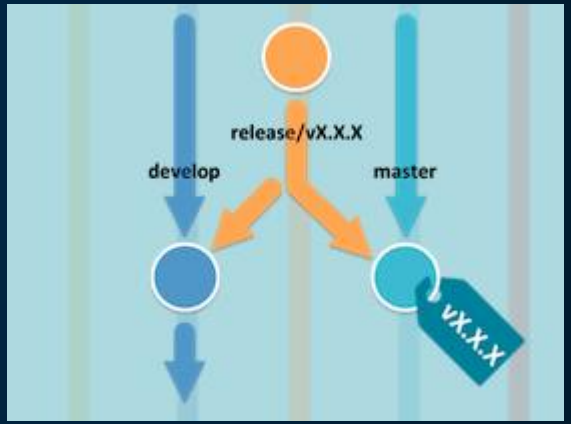
Comandos

10. El reléase debe ser publicado para garantizar por parte de todo el equipo que los ajustes a liberar para la versión o sprint son correctos, el comando es

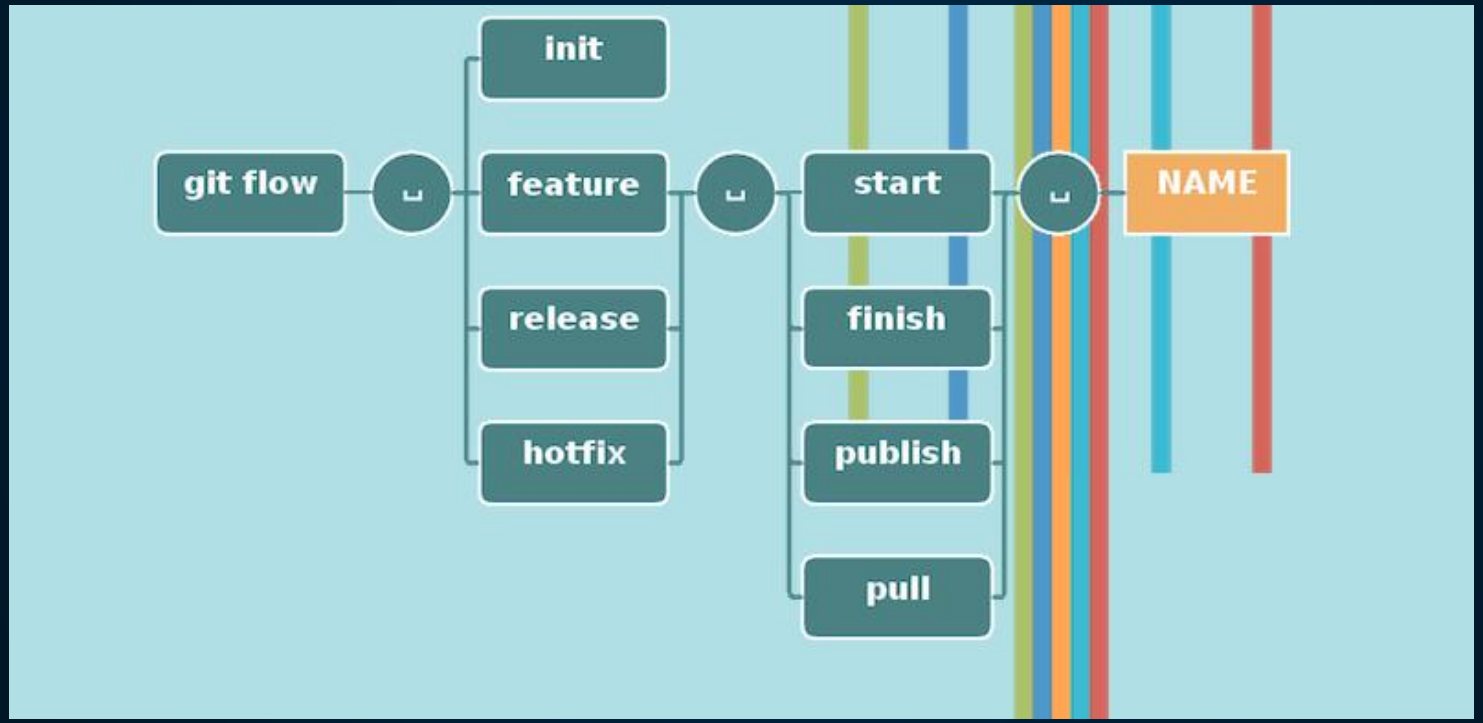
git flow release publish RELEASE

11. Para finalizar este release, utilice el comando

git flow reléase finish RELEASE



Comandos


git flow
init
feature
release
hotfix
start
finish
publish
pull
NAME

Otros Comandos de GIT

1. Para ver las ramas disponibles de un proyecto **git branch -a**
2. Para cambiar de ubicación entre ramas **git checkout nombre_rama**
3. Para crear una rama de manera tradicional **git checkout –b nombre_rama**
4. Para descartar cambios **git reset head –hard**
5. Para añadir todos los cambios a la rama **git add .**
6. Para añadir solo un archivo o un cambio de un archivo **git add ruta_archivo**
7. Para agregar un comentario sobre los cambios **git commit –m “texto”**
8. Para subir los cambios al repositorio de manera remota **git push origin nombre_rama**
9. Para descargar los cambios del repositorio de manera local **git pull origin nombre_rama**

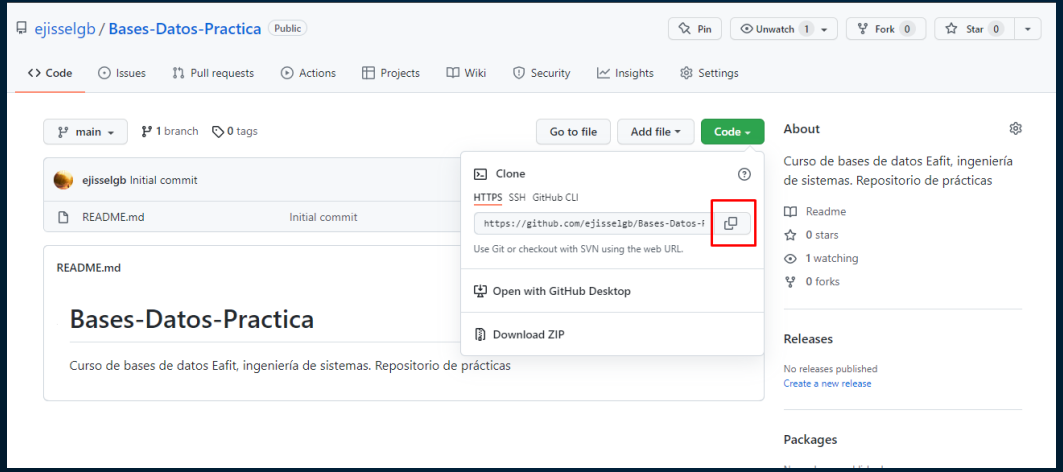
Manejo de repositorios

Práctica

Ir al repositorio

https://github.com/ejisselgb/Bases-Datos-Practica

Para descargar el proyecto ejecuta el comando **git clone + el link** que encontrarás en la página



Práctica

Ir al repositorio

https://github.com/ejisselgb/Bases-Datos-Practica

Para descargar el proyecto ejecuta el comando **git clone + el link** que encontrarás en la página

```
C:\Users\erika\OneDrive\Desktop\Clases-Eafit>git clone https://github.com/ejisselgb/Bases-Datos-Practica.git
Cloning into 'Bases-Datos-Practica'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Práctica

- Inicializar git flow

```
C:\Users\erika\OneDrive\Desktop\Clases-Eafit\Bases-Datos-Practica>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] V_
Hooks and filters directory? [C:/Users/erika/OneDrive/Desktop/Clases-Eafit/Bases-Datos-Practica/.git/hooks]
```

Práctica

- Crea una feature por equipo. Esta deberá contener los primeros apellidos de cada integrante

```
C:\Users\erika\OneDrive\Desktop\Clases-Eafit\Bases-Datos-Practica>git flow feature start gutierrez_profe
Branches 'develop' and 'origin/develop' have diverged.
And local branch 'develop' is ahead of 'origin/develop'.
Switched to a new branch 'feature/gutierrez_profe'

Summary of actions:
- A new branch 'feature/gutierrez_profe' was created, based on 'develop'
- You are now on branch 'feature/gutierrez_profe'

Now, start committing on your feature. When done, use:

     git flow feature finish gutierrez_profe
```

- Publica la rama en el repositorio para que todos los miembros del equipo tengan acceso

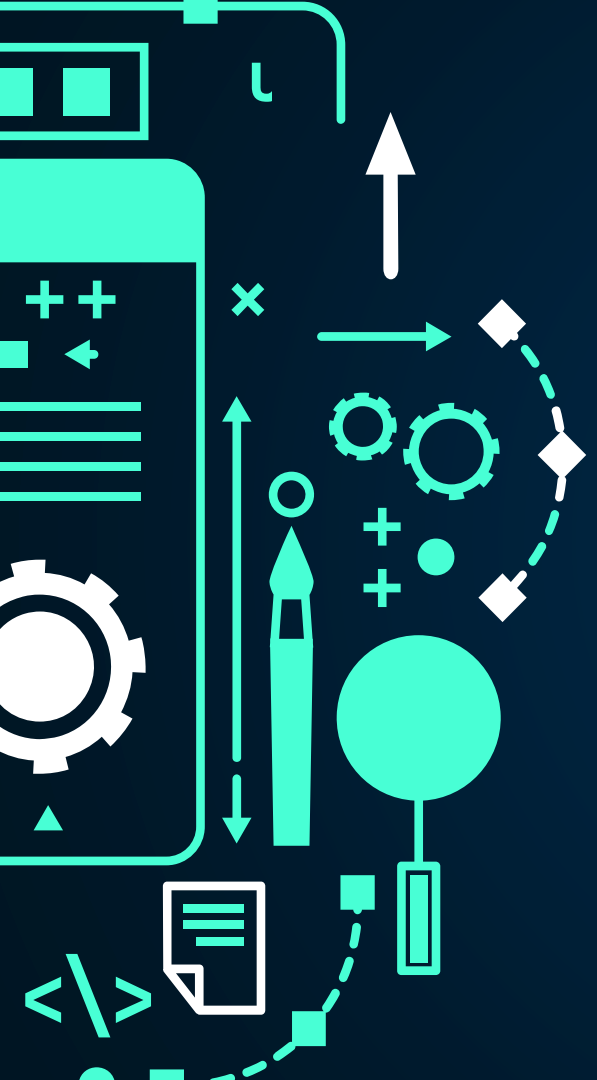
```
C:\Users\erika\OneDrive\Desktop\Clases-Eafit\Bases-Datos-Practica>git flow feature publish gutierrez_profe
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 281 bytes | 281.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature/gutierrez_profe' on GitHub by visiting:
remote:      https://github.com/ejisselgb/Bases-Datos-Practica/pull/new/feature/gutierrez_profe
remote:
To https://github.com/ejisselgb/Bases-Datos-Practica.git
 * [new branch]      feature/gutierrez_profe -> feature/gutierrez_profe
Branch 'feature/gutierrez_profe' set up to track remote branch 'feature/gutierrez_profe' from 'origin'.
Already on 'feature/gutierrez_profe'
Your branch is up to date with 'origin/feature/gutierrez_profe'.

Summary of actions:
- The remote branch 'feature/gutierrez_profe' was created or updated
- The local branch 'feature/gutierrez_profe' was configured to track the remote branch
- You are now on branch 'feature/gutierrez_profe'
```

Práctica

- Valida la rama en que te encuentras usando el comando **git branch -a**
- Luego, si estás en tu rama, crea una carpeta con los primeros apellidos de cada integrante (Esto debe hacerlo inicialmente un solo integrante)
- Agrega un archivo de texto plano indicando que aplicación quieren trabajar para el proyecto de clase
- Sube el proyecto usando los comandos explicados previamente
- Cada integrante debe clonar el repositorio, pasarse a la rama creada (en esa trabajarán durante todo el semestre)
- Cada integrante debe crear en la carpeta un archivo de texto colocando su nombre, apellido y correo electrónico
- Subir nuevamente los cambios

REFERENCIAS

- Git-Flow cheatsheet: http://danielkummer.github.io/git-flow-cheatsheet/
- Flujo de trabajo de Gitflow: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow
- ¿Qué es git?: https://www.atlassian.com/es/git/tutorials/what-is-git



Gracias!