Git advanced

First workshop

by @condef5

Git clone

Clone this repository https://github.com/Bootcamp-x/recetas-selvaticas

Git log

```
git log
git log --oneline
git log -n 5
```

Git blame

```
git blame filename
```

For vscode users, use git lens extension

Git remote

```
git remote -v
```

Create a new remote

```
git remote add [remote-name] [url repository]
```

Delete a remote

```
git remote remove [remote-name]
```

Git branch

```
git branch branch-name
```

Git checkout

```
git checkout branch-name
```

Git add

```
git add .
git add -p filename
```

First Task

1. In groups of 2 people, create a new branch.
2. Each member should add a commit from their computer.
3. Upload your changes.
4. Create a unique pull request per group.
5. Merge the pull request from GitHub.
6. Update your local branches.

Good Commits

1. Brief and descriptive: The commit message should be brief yet descriptive, clearly and concisely
2. Use imperatives: Commit messages should start with an imperative verb, such as "Add", "Fix", "Update", "Remove", etc.
3. Proper capitalization and punctuation.
4. Keep it reasonably short.
5. Reference to issues or pull requests.

Commit Conventions

Fix (Corrección):

It is used when a commit resolves an error or bug in the code.
It is useful to quickly identify commits that fix specific issues in the code.
Example commit message: `Fix issue with login form validation`

Chore:

It is used for changes that are not directly related to user functionality or error correction, such as administrative tasks, refactoring, dependency updates, etc.

It helps separate functional changes from other types of changes in the repository history.

Example commit message: `Update dependencies`

Feat (Característica):

It is used to indicate the implementation of a new feature or functionality in the code.

It is useful to quickly identify commits that add new features to the project.

Example commit message: `Add user profile page`

Second Task

1. Repeat task 1 using good commits and commit conventions

Git commit

```
git commit
git commit -m 'message'
git commit --allow-empty -m "Trigger Build"
```

Git stash

```
git stash
```

“ When you run git stash, Git saves uncommitted changes to a temporary storage area called "stash." This leaves your working directory clean so you can switch branches, apply patches, merge changes, etc. ”

List all stash

```
git stash list
```

Apply last stash saved

```
git stash apply
```

Applies the changes from the stash specified by the index n in the stash list.

```
git stash apply stash@{n}
```

Third Task

1. Create a commit to references some link
2. Save some changes using stash
3. Use stash apply
4. Create 3 stash and apply the second one.

Git merge

Git rebase

Git rebase onto

```
git rebase --onto branch-name head~n

# example
git rebase --onto dev head~3
```

Git reset

```
git reset
git reset --soft head~1
git commit --amend --no-edit
```

Git reflog

```
git reflog
```