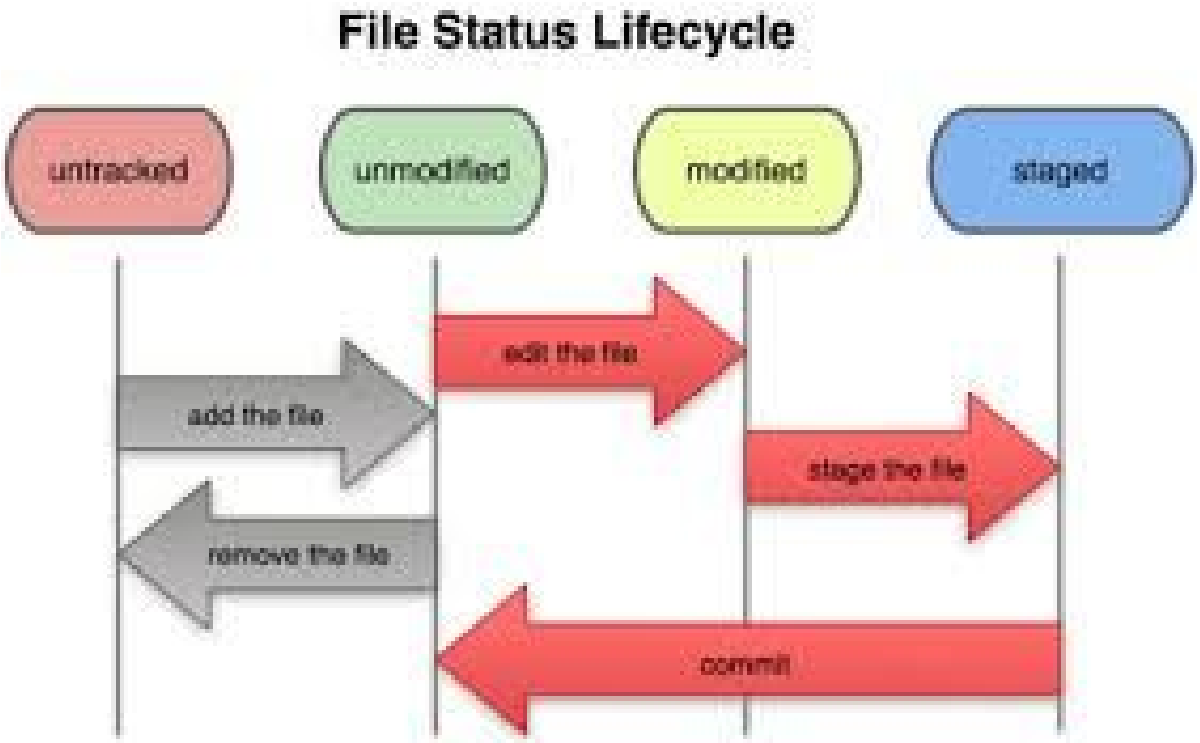
sgit

A Scala-based git-like code source manager

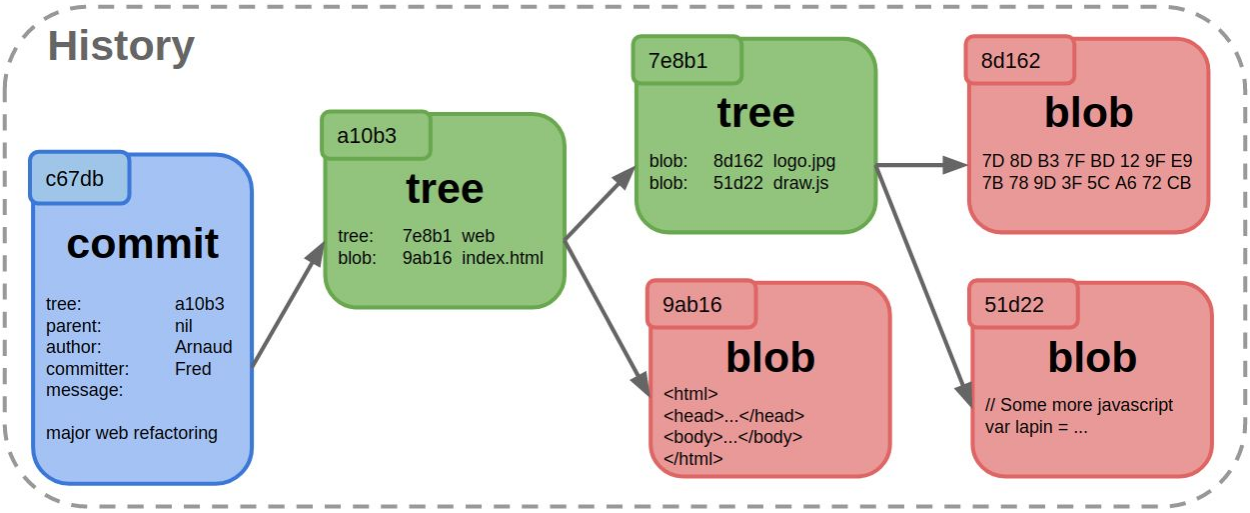
Git (/ɡɪt/)[7] is a distributed version-control system for tracking changes in source code during software development.[8] It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed,[9] data integrity,[10] and support for distributed, non-linear workflows.[11]

Description

Git File status Lifecycle



Git Anatomy



HomeWork

Each student has to provide an implementation of sgit as a full packaged command-line tool. Here are the requirements:

R1) SGit required functionalities:

- Create:
  - sgit init
- Local Changes:
  - sgit status
  - sgit diff
  - sgit add <filename/filenames or . or regexp>
  - git commit

- Commit History:
  - sgit log
    Show all commits started with newest
  - sgit log -p
    Show changes overtime
  - sgit log --stat
    Show stats about changes overtime

- Branches and Tags
  - sgit branch <branch name>
    Create a new branch

    - sgit branch -av
      List all existing branches and tags
    - sgit checkout <branch or tag or commit hash>
    - sgit tag <tag name>

- Merge & Rebase
    - sgit merge <branch>
    - sgit rebase <branch>
    - sgit rebase -i <commit hash or banch name>

What is not mandatory:

- No compression is asked
- No remote (no pull/push)
- No binary files to handle only text files (easier for diff)

R2) Tests:

You should add tests to test each of your functionalities.

R3) Report:

A short report (3 to 5 pages max), written in English. This report should be a synthetic report with 4 sections

1) Instructions
   You should explain what to do to compile and run your program (the list of command from your root source folder).
   Be careful, it should not need to modify your code in any way.
2) Architecture
   In this section, you should
    a) explain what is the architecture of your application and your conception choices.
    b) explicitly have two subsections talking about the pros and the cons of your architecture
    c) add at least one figure to describe your application
3) Explain your test strategy and show one or two idiomatic examples

4) Post Mortem
   A **project post-mortem** is a process, usually performed at the conclusion of a project, to determine and analyze elements of the project that were successful or unsuccessful.
    a) What went right
    b) What went wrong
    c) Lessons learned
       // In this subsection at least try to answer this question: What will you do to not reproduce the same mistakes?

Delivery:

You have to deliver your work before the **October, 20 11:42 pm**[1].

You will have to send an email with “[IG5][FP-2019] SGIT” as subject
In the body of the email, you have to provide:

- The url to a github or gitlab repository with your source code
    - please upload only source code (do not upload binary code, e.g .class files),
    - If your project is private
        - add aca-polytech on github
        - Add @arnaud.castelltort on gitlab
- A report (3 to 5 pages max) **in pdf format**

[1] Be careful, this is a **hard deadline**.