CONTRIBUTING: FortranAnalyser

This manual will tell developers how to interact with the environment to make their contributions to the project. It consists of a series of instructions to know how to contribute correctly, in order to carry out a good management of the same and to keep informed to all the personnel of the state in which the project is located at a certain moment. This project is developed in a continuous integration environment with continuous deployment on a pre-production server. This environment is composed of a series of tools that automate the process, all led by the POM of this project.

1. TASK EXECUTION

The process to carry out a task will be the following:

- Access to Redmine and select one of the tasks that one is responsible for developing, paying special attention to the description of it.
- Open the development environment.
- Verify that you are in the develop branch and that you have the latest version of the project. If you do not meet these two requirements, solve the problem by changing the branch and cloning the latest version.
- Pull the latest changes.
- Implement the solution, including the tests.
- Make a commit with each stable part that is developed, that is, stable and tested.
- Each time a commit is made, it will be sent to the central repository to share with the rest of the development team.
- It will then be verified that the build works correctly on the Jenkins Continuous Integration Server. In case the construction fails, the steps described in section "CONSTRUCTION FAILS". If the build is correct then it will be checked that the project has been deployed and working properly on the pre-production Wildfly server and that the Maven Nexus repository has a new version of the project. Refer to the development environment section for more information.
- Once the working day is over, complete the hours in the Redmine task.

2. CONSTRUCTION FAILS

Both the master branch and the develop branch will be controlled by the integration server that will execute all the tests after a commit. In case the construction fails in Jenkins, you must undo the commit to return to a stable state of the project. This step is very important and there are several ways to do it, however the way it will be done will be through the following command:

git push origin +HEAD^:develop

What this command accomplishes is to force the remote develop branch to be placed in the previous commit to HEAD, since this is the conflicting commit. This form is the most direct way to do it. Finally, you should delete the local commit that is still wrong. To do this, the following command will be written:

git commit --amend

In case of downloading the local commit but you want to keep in the state of the files, you can use this other command:

*git reset --mixed HEAD^*