**Inheritance hierarchy**

Current choice of design is to use inheritance since it avoids using many different helpers in a single class. The inheritance hierarchy is as follows:

CategoriesHelper

FormatHelper -> LocalHelper -> ScoreHelper - > DataHelper-> SonarHelper Helper

GitlabHelper

The main class ProcessSonar.py delegate most of the functionalities to Helper, which inherits from other helper classes. To add new helper method, add the method to corresponding helper class based on its purpose. Then use self.helper.<new method> in ProcessSonar.py or self.<new method> in other helper classes.

**End points**

"/api/show" #get all issues

"/api/statistics" #get statistics of the project

"/api/file/xml" #get required xml file to run pipeline on gitlab

"/api/file/yml" #get required yml file to run pipeline on gitlab

"/api/duplications" #get issues regarding duplications

"/api/lmethod" #get longest method in the project

"/api/commit" #get gitlab commit information about a project

"/api/commitstat" #get gitlab commit statistics about a project

"/api/directory" #get issues by directories

"/api/project" #check whether a project has been analyze or return the history of the project analysis

"/api/author" #get issues by author’

**JSON**

Check json/example-json for JSON templates for each endpoints

**Shell**

To add additional tools to the backend, the simplest way to do it is to add a shell script to server/shell folder, and then add a caller function in LocalHelper.py. All existing shell scripts caller function are in LocalHelper.py begins with executeShell. Use subprocess.checkoutput to get all terminal printouts from executing the shell scripts.