## Use-Cases:

Detail use cases for job/user stories.

A rule of thumb: function/user level Use case <=> user/job story in product (business) requirements.

1. Create a new pipeline.
   1. As a pipeline’s editor I want to create a new empty pipeline so that I will have a ready empty environment for a new pipeline.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user clicks on the “Create new” button.
      4. The authorized user chooses the “Create empty” option.
      5. The authorized user gets notified that the pipeline was created
      6. The authorized user gets redirected to pipeline’s configuration editing page.
   2. As a pipeline’s editor I want to create a new pipeline out of a git repository so that I will have a new pipeline with configurations loaded from the git repository.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user clicks on the “Create new” button.
      4. The authorized user chooses the “Create from git” option.
      5. The authorized user gets notified that the pipeline was created
      6. The authorized user gets redirected to pipeline’s configuration editing page where all his files from the git repository are presented.
2. View all pipelines.
   1. As an authorized user I want to view all pipelines available to me so that I can do CRUD operations with them.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user sees a table consisting of pipeline rows. Each row has a name, description and the buttons: “Show info”, “Edit”.
3. View pipeline’s information
   1. As an authorized user I want to view the pipeline's information so that I can know its name, description, owner, status, storage location.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user selects any pipeline by clicking on the “Show info” button near the corresponding pipeline’s name and description.
      4. The authorized user sees the name, description, owner, status, storage location of the pipeline.
4. Edit Pipeline
   1. As a pipeline's editor I want to edit my pipeline’s meta information (name, description) so that it would make my user experience better.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user selects any pipeline by clicking on the “Edit” button near the corresponding pipeline.
      4. The authorized user sees the name, description of the pipeline.
      5. The authorized user clicks on the pipeline’s name, description.
      6. The authorized user enters the new pipeline’s name and description.
      7. The authorized user clicks on the “Save” button.
      8. The authorized user sees a notification that the pipeline was successfully changed.
   2. As a pipeline's editor I want to edit my pipeline’s configuration so that it would better suit business requirements.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user selects any pipeline by clicking on the “Edit” button near the corresponding pipeline.
      4. The authorized user sees the configuration of the pipeline shown in a text editor with syntax highlighting of the yaml file.
      5. The authorized user edits the configuration by writing a yaml declarative model corresponding to their business requirements.
      6. The authorized user clicks on the “Save” button.
      7. The authorized user sees a notification that the pipeline was successfully changed.
   3. As a pipeline's editor I want to apply git operations to my pipeline’s configuration so that it will collaborate with my remote git repository.
      1. The authorized user loads the web application.
      2. The authorized user clicks on the “Pipelines” button.
      3. The authorized user selects any pipeline by clicking on the “Edit” button near the corresponding pipeline.
      4. The authorized user sees the configuration of the pipeline shown in a text editor with syntax highlighting of the yaml file.
      5. If the pipeline is stored in the git repository, then the authorized user sees the git operations buttons. Otherwise, the authorized user sees the button “Clone git repository”. By clicking on these buttons, the user does corresponding git actions.

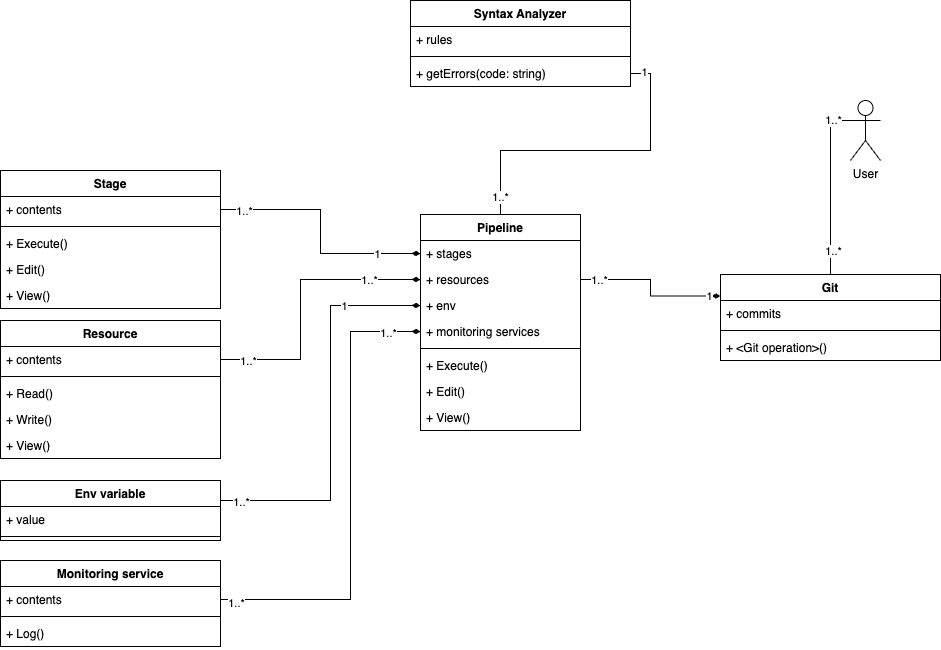
## Glossary

* 1. The framework:
     1. Automates development and maintenance of AI-services
     2. Executes, monitors, stores, updates, and secures ML applications
     3. Runs pipelines stage by stage
     4. Provides with authentication data
  2. The editor:
     1. Saves pipelines
     2. Verifies the pipelines syntax
     3. Validates the pipelines
     4. Gets authentication data from the Framework
     5. Incorporates Git
  3. A pipeline:
     1. Consists of stages
     2. Gets data from input resources
     3. Processes data
     4. Writes results to output storages (consumers)
     5. Has env variables

## Applying SIAOUT

|  | S (several instances) | I (interface change) | A (a state has many attributes) | O (several operations in interface) | U (all the attributes and operations apply to all instances) | T (crucial requirement) |
| --- | --- | --- | --- | --- | --- | --- |
| User | + | + | + | + | + | + |
| Application | - | - | - | - | - | + |
| Web-editor | + | + | + | + | + | + |
| Pipeline | + | - | - | - | + | + |
| Stage | - | - | - | - | + | + |
| Resource | + | + | + | + | + | + |
| Env variable | - | - | - | - | + | + |
| Authentication | + | + | - | - | + | - |
| Git | + | + | + | + | + | - |

## Draft class model



## Assigning DDD stereotypes

