# Summary

### Domain area

The “MLOps Platform” of the HSE university provides a set of tools for Machine Learning models and ETL. Developers and analysts using the platform are provided with many capabilities, including ones for managing requests pipelines. A requests pipeline is a sequence of stages written in the “YAML” language, declaring the way data should be extracted, handled, transformed and stored.

### Problem

Pipelines may become big relatively fast. Moreover, pipelines are based on the resources description and involve an extended “YAML” syntax. Therefore, pipelines often happen to be hard to explore and modify.

### Solution

A web-editor for pipelines with auto-saving and automatic syntax and cohesion checking, embedded into the “MLOps Platform”. The editor should be based on the existing “MLOps Platform” services, including ones for Authorization and pipeline storing.

# Stakeholders

1. The “MLOps Platform” developers
2. Machine Learning specialists
3. Data Engineering Specialists
4. The Higher School of Economics

# Features

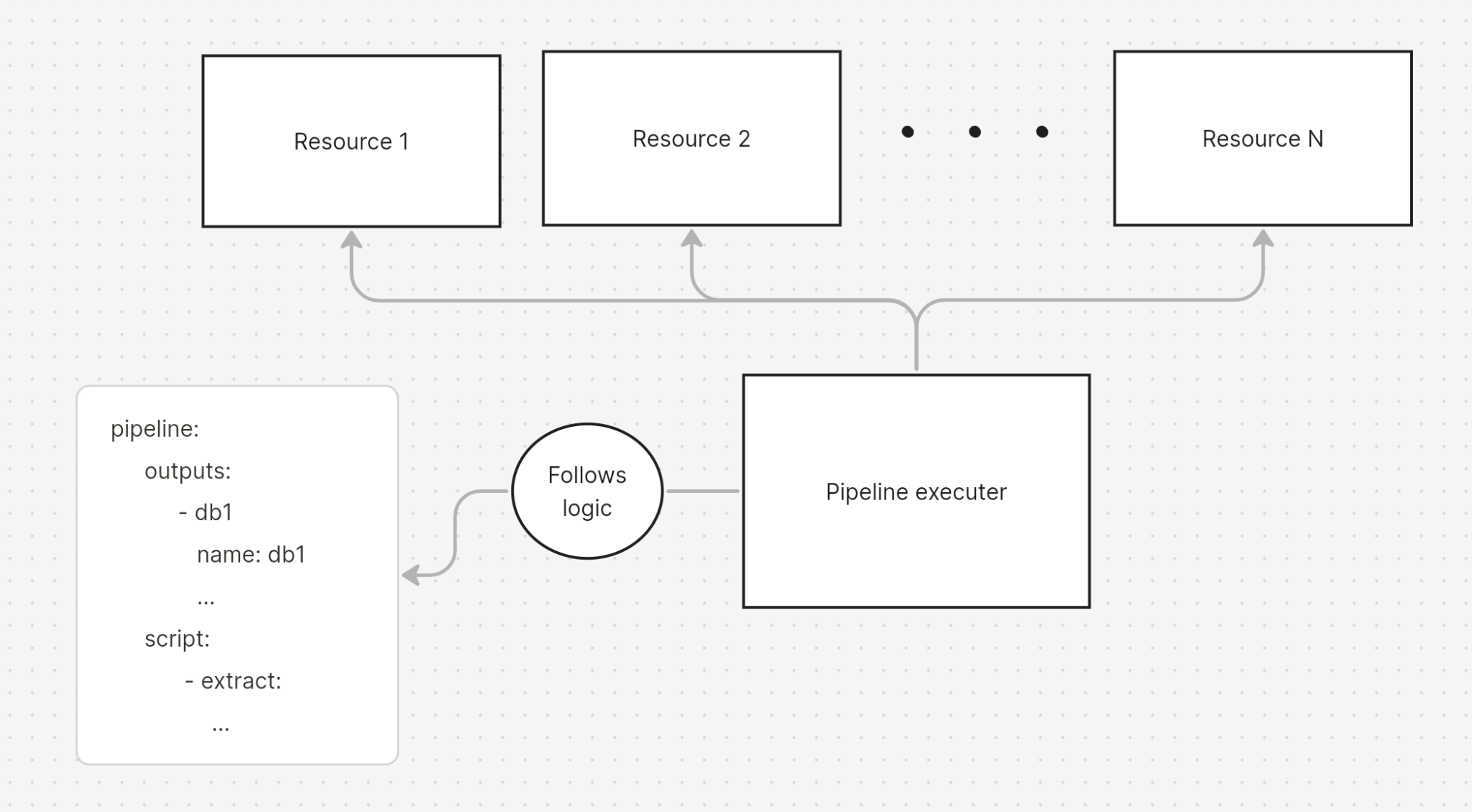
1. Viewing existing pipelines.
2. Creating new pipelines. This includes uploading configurations, declaring the resources used during the pipeline execution.
3. Editing the existing pipelines. This includes syntax checks
4. Saving the contents of the pipelines edited. This should happen both automatically and on a specific button click. The process should be executed in relation to either local device, or the “MLOps Platform” storages.
5. Synchronizing the pipeline with Git

# Constraints

1. **Security.** User actions logs should not contain any personal data (it should contain the in-editor actions only).
2. **Maintainability.** The pipelines editor’s logic should not be depended on Visual Studio Code’s specifics in order for it to be implemented as a standalone tool. The editor should expect and show changes from other sources (for possible multiple users support in future).
3. **Usability.** The web-editor should be implemented as the Visual Studio Code’s extension and use VS Code’s UI logic conventions and style.
4. **Reliability.**
   1. **Availability.** The editor’s offline functionality should be available for usage at any given time.
   2. **Recoverability.** The editor should be able to restore its functioning state after any failures at no longer time that is needed to open normally.
5. **Observability.** The users’ actions for the last 24 hours should be logged and saved into a file.

# Diagrams:

### General:



### Web-editor: