Model podataka

EasyFlow

Članovi tima:

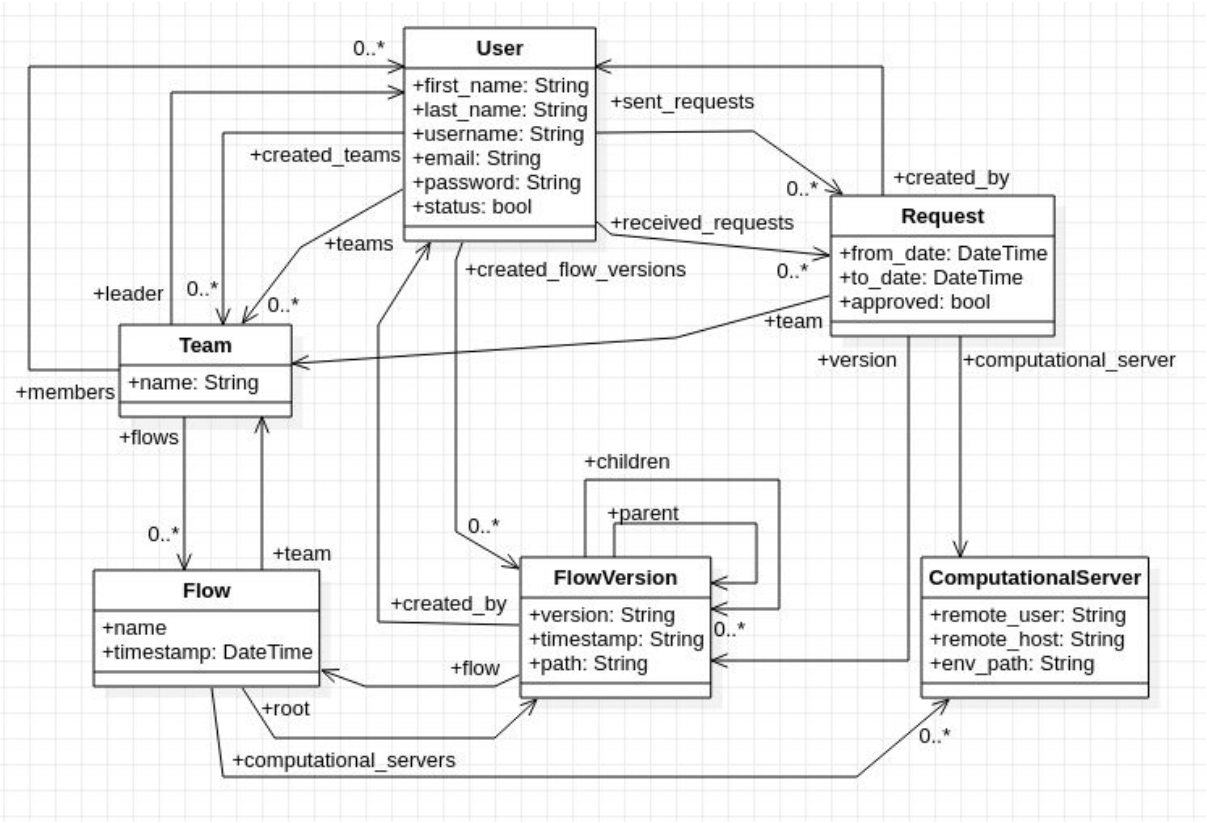
Đorđe Pavlović 16797
David Nikolić 16779

Model podataka

Aplikacija EasyFlow se sastoji iz više odvojenih serverskih delova. Komputacioni serveri i centralni server podataka povezani su preko centralnog servera koji upravlja i korisničkim zahtevima. Na ovom serveru se nalazi Version Control System, dok je na svakom od komputacionih servera instaliran Flow Manager framework.

Version Control System

Version Control System upravlja korisničkim zahtevima i šalje zahteve komputacionim serverima za izvršenje tokova algoritama. On vodi računa o verzijama algoritama koje korisnici kreiraju preko grafičkog interfejsa, tj. web aplikacije. Navedene klase su mapirane na MySQL bazu podataka.



User - Sadrži podatke o korisniku, kao i njegove veze sa timovima, tokovima kreiranih unutar timova i njihovih verzija.

Team - Objedinjuje više korisnika koji rade zajedno na nekom projektu, od kojih je jedan lider tima i ima svoje privilegije. Unutar tima kreiraju se tokovi koje svi korisnici mogu videti, a lider i korisnici sa dozvolom lidera mogu i pokrenuti njihovo izvršavanje na nekom komputacionom serveru.

Flow - Predstavlja tok algoritma koji se kreira unutar nekog tima i sadrži barem jednu verziju.

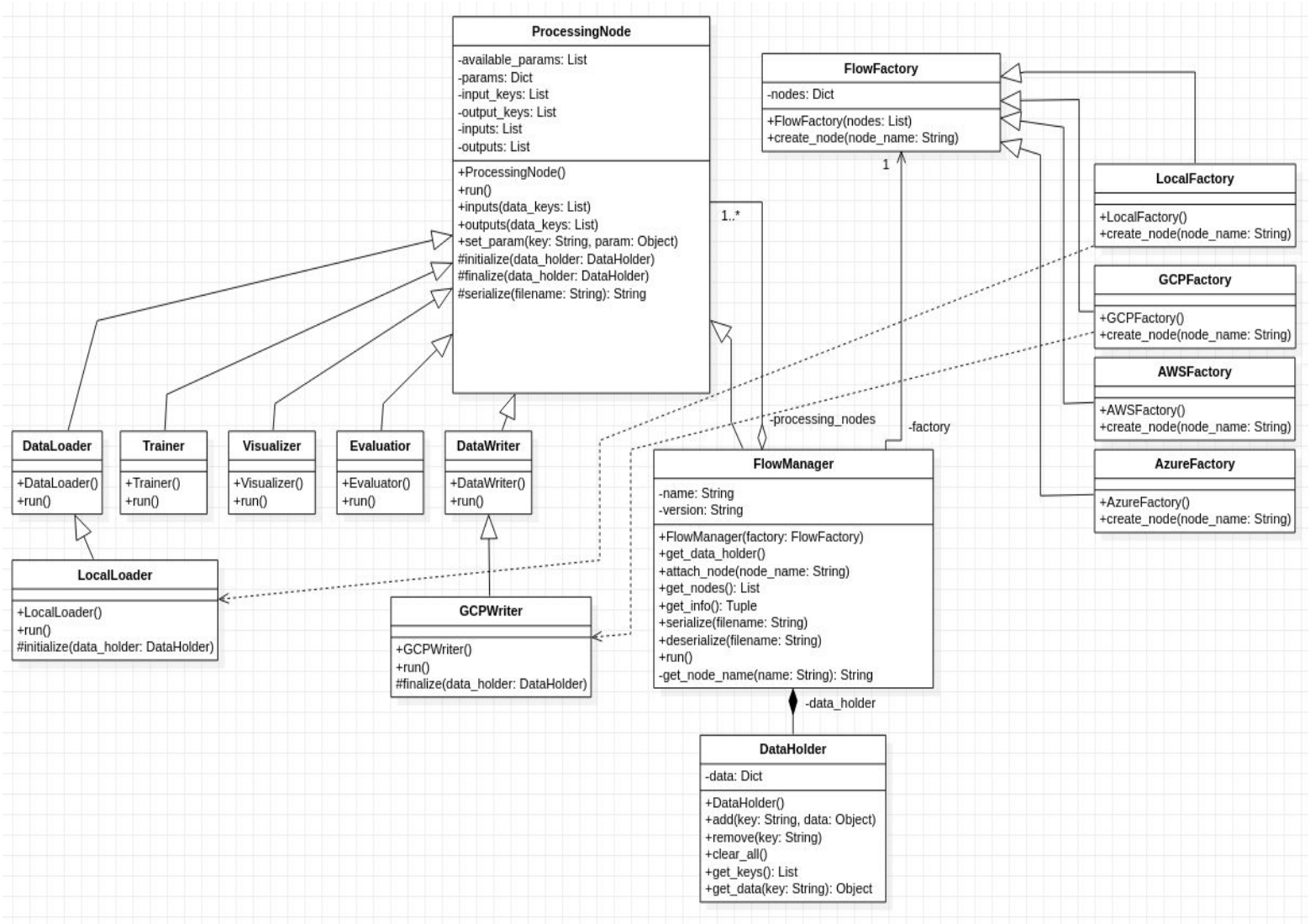
FlowVersion - Predstavlja konkretnu verziju nekog toka. Sadrži putanju do json fajla u kome je serijalizovan tok algoritma sa svim svojim parametrima, koji se nalazi na fajl sistemu centralnog servera podataka. Strukture je stabla pošto sadrži verziju iz koje je izveden (parent), kao i verzije koje su iz njega izvedene (children).

Request - Zahtev upućen lideru tima za dozvolom izvršenja date verzije toka algoritma na konkretnom komputacionom serveru u određenom vremenskom periodu.

ComputationalServer - Sadrži informacije o udaljenom serveru. Svakom toku se dodeljuje jedan broj komputacionih servera.

Flow Manager

FlowManager se instalira na komputacionim serverima i izvršava tokove algoritama. Algoritmi koji se kreiraju preko njega čuvaju se u json formatu na fajl sistemu centralnog servera podataka. Zasnovan je na PipelineFilter, Composite i AbstractFactory projektnim obrascima.



DataHolder - Sadrzi podatke koji se nalaze u pipeline-u, upravlja njihovim dodavanjem i uklanjanjem.

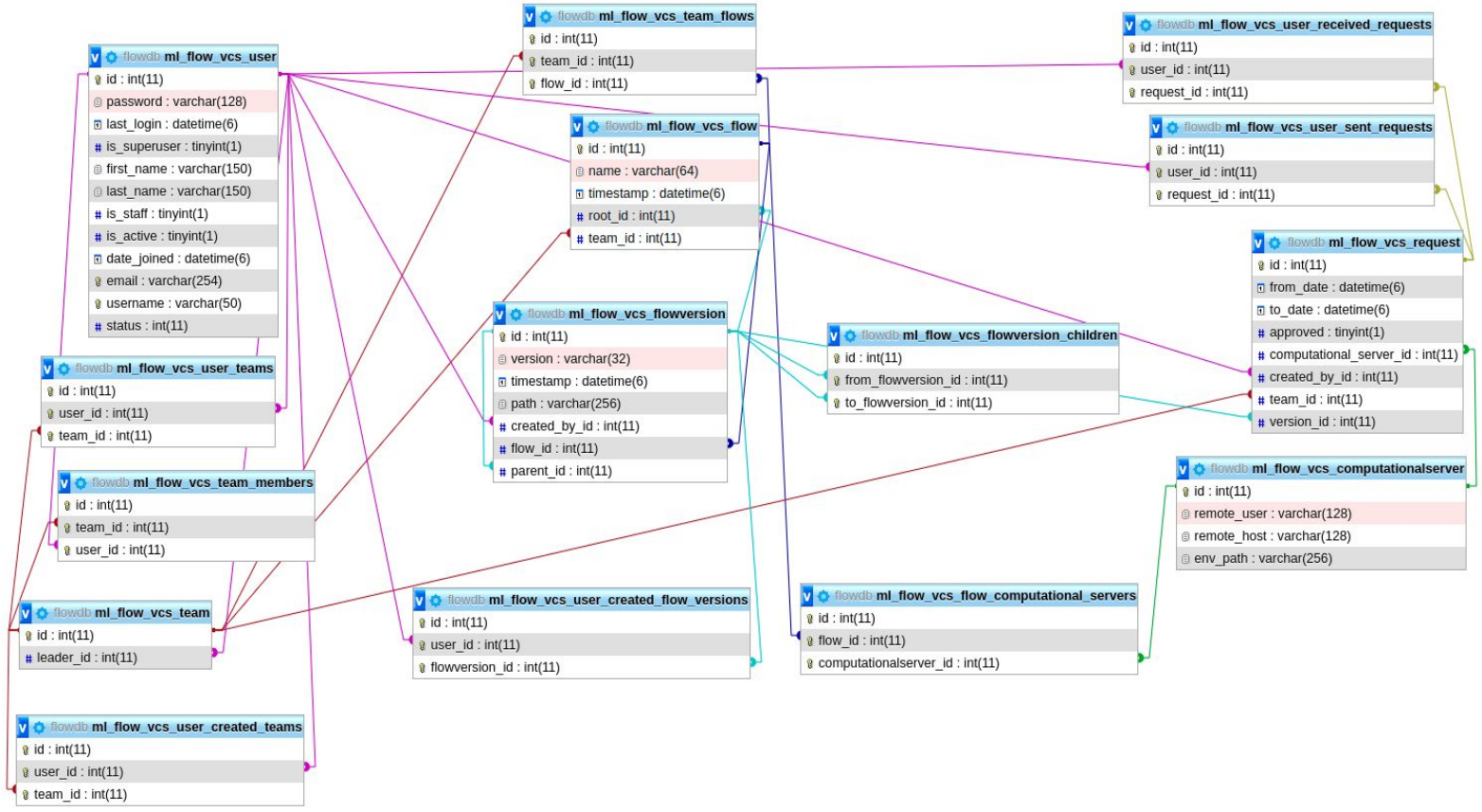
FlowManager - Baziran je na Composite projektnom obrascu. Upravlja izvršavanjem celog toka podataka tako što iterira i izvršava filtere koji su mu dodati.

ProcessingNode - Baziran je na Composite projektnom obrascu. Predstavlja apstraktnu klasu koju nasleđuje FlowManager i svi konkretni filteri.

FlowFactory - Baziran je na Abstract Factory projektnom obrascu. U zavisnosti od lokacije i tipa komputacionog servera, pojedine filtere je potrebno drugačije implementirati.

Model perzistencije

Sastoji se od MySQL baze u koju se skladišti VCS model podataka i json fajlova u kojima su serijalizovani tokovi algoritama. Tabele MySQL baze se mogu videti na sledećem dijagramu.



Tokovi algoritama serijalizovani su u json fajlu u sledećem formatu (primer toka koji vrši evaluaciju modela treniranom na mnist dataset-u).

```
{
    "flow_name": "mnist_flow",
    "flow_version": "0.0.1",
    "nodes": [
        {
            "available_params": [
                "dataset"
            ],
            "input_keys": [],
            "output_keys": [
                "x_train",
                "x_test",
                "y_train",
                "y_test"
            ],
            "params": {
                "dataset": "mnist"
            },
            "type": "keras_dataset_loader"
        },
        {
            "available_params": [],
            "input_keys": [
                "x_train",
                "x_test"
            ],
            "output_keys": [
                "x_train",
                "x_test"
            ],
            "params": {},
            "type": "data_normalizer"
        },
        {
            "available_params": [
                "model_path",
                "loss",
                "metrics",
                "optimizer",
                "compile"
            ],
            "input_keys": [],
            "output_keys": [
                "model"
            ],
            "params": {
                "compile": false,
                "model_path": "mnist.h5",
                "optimizer": null
            },
            "type": "model_loader"
        },
        {
            "available_params": [
                "model_path",
                "loss",
                "metrics",
                "optimizer"
            ],
            "input_keys": [
                "model",
                "x_train",
                "y_train"
            ],
            "output_keys": [
                "eval"
            ],
            "params": {
                "optimizer": null
            },
            "type": "model_evaluator"
        }
    ]
}
```

Mapiranje

Koristi se MySQL baza podataka i django-v objektno-relacioni maper za generisanje tabela. Tabele su mapirane iz python klasa koje nasledjuju klasu `django.db.models.Model`.

Repository Data Layer Pattern - Sistem sadrži centralnu bazu podataka kojoj pristupaju nezavisne komponente.