

# Cloud and Service Oriented Computing

Subsystem: MAS Controller and Algorithms

Delivery 2



Professor:

Jorge Barbosa

Students:

Domingos Junior up202001536

Priscilla Melin up201900048

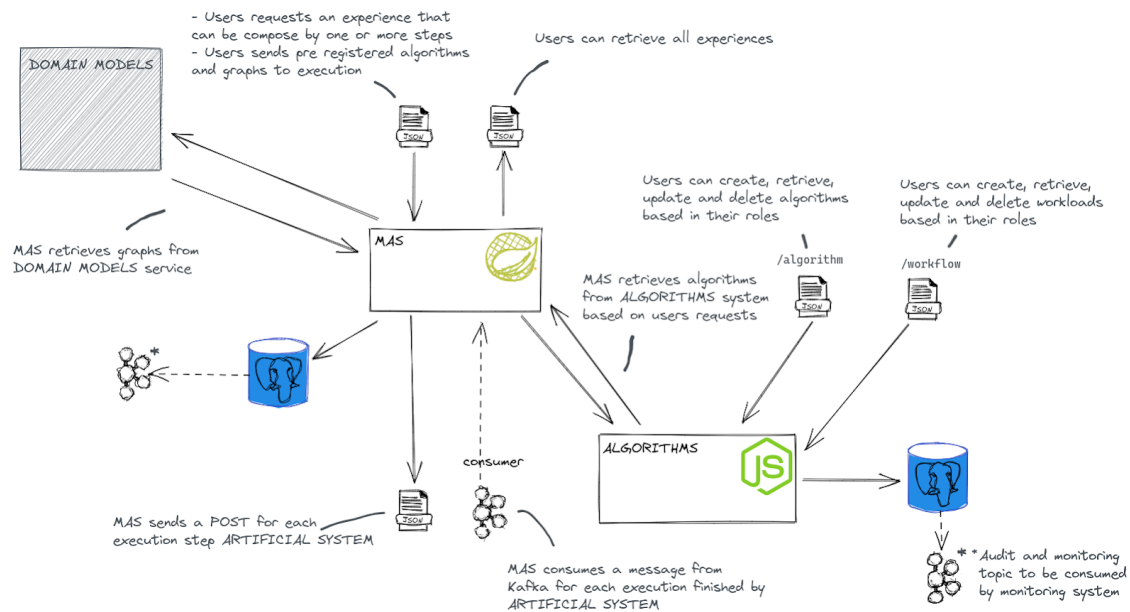
Kadu Barral up202000017

## 1 **TABLE OF CONTENTS**

---

Architectural System Design	<b>3</b>
Services Descriptions and Operations	<b>3</b>
Algorithm Service	3
Operations	4
Algorithm Operations	4
Workflow Operations	4
MAS Service	4
Operations	4
Resilience Patterns	<b>5</b>
Observability Patterns	<b>6</b>
Security Implementation	<b>6</b>
Service API Specification	<b>6</b>
Algorithm Service API:	6
MAS Service API:	6
Testing	<b>7</b>

## 2 ARCHITECTURAL SYSTEM DESIGN



## 3 SERVICES DESCRIPTIONS AND OPERATIONS

In terms of business capabilities, the MAS Controller and Algorithms subsystem have two main goals:

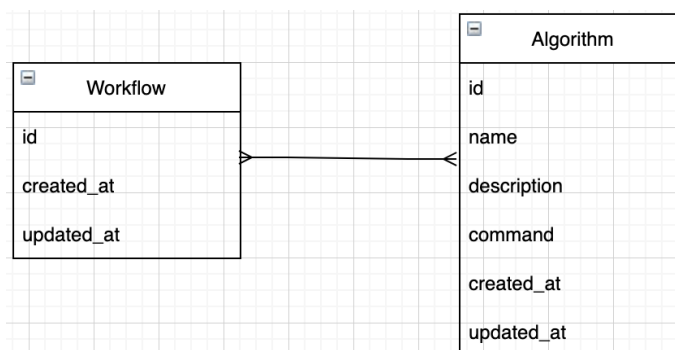
1. Management of Algorithms and Workflows.
2. Coordination of the Workflow execution.

Since these capabilities are independent they can be developed and deployed as autonomous services.

### ALGORITHM SERVICE

Responsible for managing algorithms and workflows.

An algorithm is represented by a name, description and a command while a workflow is represented by a set of algorithm ids.



## Operations

### 1. *Algorithm Operations*

**Create:** Given a Designation and a Command, create an Algorithm.

**Read:** Given the id of an existing Algorithm, retrieve its current data.

**Read All:** Retrieve all registered algorithms.

**Update:** Given the id of an existing Algorithm, and a new designation, description and/or command, update its data in the database.

**Delete:** Given the id of an existing Algorithm, remove it from the database.

### 2. *Workflow Operations*

**Create:** Given a set of ids of existing Algorithms, create a Workflow keeping the provided order.

**Read:** Given the id of an existing Workflow, retrieve its current data.

**Read All:** Retrieve all registered workflows.

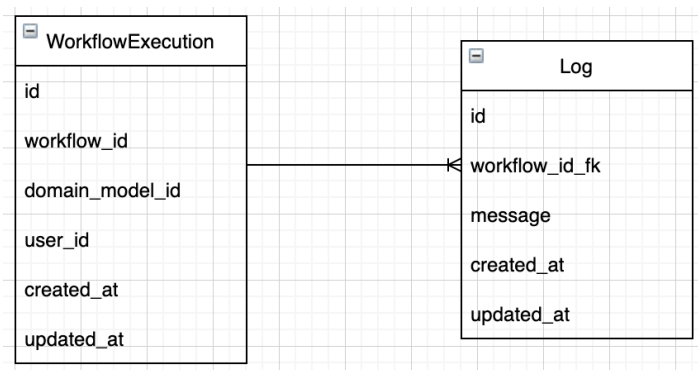
**Update:** Given the id of an existing Workflow, and a new set of existing ids, update its data in the database.

**Delete:** Given the id of an existing Workflow, remove it from the database.

## **MAS SERVICE**

Responsible for coordinating the workflow execution.

An execution is represented by a workflow\_id, an user\_id and a domain\_model\_id. Each execution will be logged in for coordination purposes.



## Operations

**Create Workflow Execution:**

Given the id of an existing Domain Model, a Workflow id, and a user id, create an instance of Workflow execution in the database, and coordinate its full execution.

This operation takes the Domain Model as input for the first node/Algorithm, and each subsequent input is the result of the processing of the last step, which means that if any of the steps fails, the whole process fails.

**Read:** Given the id of an existing Workflow Execution, and a user id, retrieve its current data, which will contain the status of the execution at the moment.

**Read All:** Given a user id, retrieve data from all Workflow Executed by them.

Service	Operations	Collaborators
Algorithm Service	createAlgorithm() findAlgorithmById() findAllAlgorithms() deleteAlgorithm() updateAlgorithm() createWorkflow() findWorkflowById() findAllWorkflows() deleteWorkflow() updateWorkflow()	Monitoring Service
MAS Service	createWorkflowExecution() findAllWorkflowExecutions() findWorkflowExecutionById()	Algorithm Service findAllAlgorithms() findAlgorithmById() Domain Model Service findGraphs() Monitoring Service ArtificialSystemService

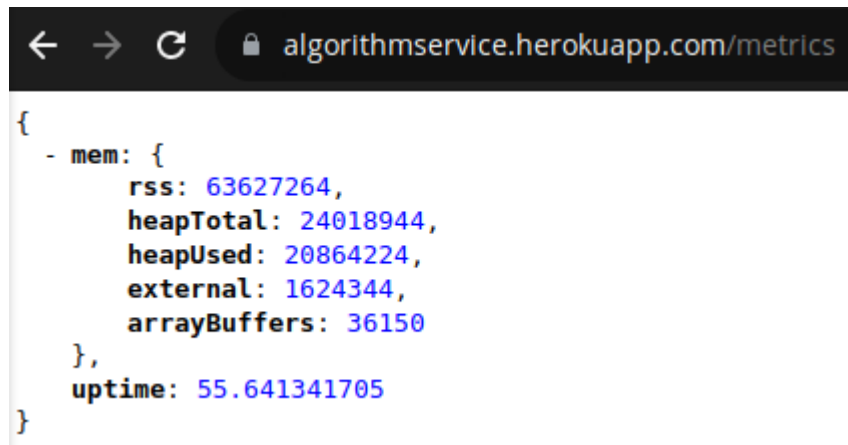
## 4 RESILIENCE PATTERNS

- All services are running in the cloud with high availability multi-zone distribution;
- We implemented asynchronous communication using Kafka;
- Our services are stateless connecting in a external database (one database per server), with this is possible to scale in diverse instances;

## 5 OBSERVABILITY PATTERNS

---

- Health check API
  - MAS and Algorithm exposes an endpoint informing their status - /health. This endpoint is monitored by the Monitoring Service.
- Application Metrics
  - Metrics about system performance can be returned in /metrics endpoint.



```
{
  - mem: {
    rss: 63627264,
    heapTotal: 24018944,
    heapUsed: 20864224,
    external: 1624344,
    arrayBuffers: 36150
  },
  uptime: 55.641341705
}
```

- Log Aggregation
  - All log events are sent to Monitoring System
- Audit logging
  - All action are logged in Monitoring System, not just errors or exceptions

## 6 SECURITY IMPLEMENTATION

---

- Authentication with API gateway access token (authorization user roles not implemented)
- HTTPS requests only

## 7 SERVICE API SPECIFICATION

---

### ALGORITHM SERVICE API:

<https://algorithmservice.herokuapp.com/api-docs/>

### MAS SERVICE API:

<https://mascontroller.herokuapp.com/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config#>

## 8 TESTING

Given the id of an existing Workflow Execution, and a user id, retrieve its current data, which will contain the status of the execution at the moment.

```
POST https://mascontroller.herokuapp.com/workflow/findById Send 200 OK 476 ms 434 B Just Now
Other Auth Query Header Docs Preview Header Cookie Timeline
1 {
2   "workflowExecutionId": 23,
3   "userId": "test@test.pt"
4 }
1 {
2   "error": false,
3   "message": "Workflow found!",
4   "workflowExecution": {
5     "id": 23,
6     "workflowId": "c91efcdf-f3c6-48bd-b06a-adbb8d37e9ac",
7     "domainModelId": "51",
8     "userId": "test@test.pt",
9     "description": "Execution of the first workflow.",
10    "steps": 4,
11    "currentStep": 1
12  },
13  "logs": [
14    {
15      "action": "Execution of the first workflow.",
16      "message": "Workflow execution id: 23. Step 1 of 4. Algorithm being processed:
Algorithm1. Domain topology/file: MAS TEST: topology 1."
17    }
18  ]
19 }
```

Create: Given a set of ids of existing Algorithms, create a Workflow keeping the provided order.

```
POST https://algorithmsservice.herokuapp.com/workflows Send 200 OK 207 ms 600 B
Other Auth Query Header Docs Preview Header Cookie Timeline
1 {
2   "ids": ["a25943c4-3f22-4715-82e7-581e1591481c", "8dd064b7-c208-45c8-912a-67fd7409a74b"]
3 }
1 {
2   "id": "60e478f7-b1a3-46d7-8d8e-08d031341d86",
3   "createdAt": "2022-01-21T17:45:22.300Z",
4   "updatedAt": "2022-01-21T17:45:22.300Z",
5   "algorithms": [
6     {
7       "id": "a25943c4-3f22-4715-82e7-581e1591481c",
8       "name": "Algoritmo1",
9       "description": "Descrição do algoritmo1.",
10      "command": "Comando - algoritmo1.",
11      "createdAt": "2022-01-17T13:30:48.687Z",
12      "updatedAt": "2022-01-17T13:30:48.687Z",
13      "order": 1
14    },
15    {
16      "id": "8dd064b7-c208-45c8-912a-67fd7409a74b",
17      "name": "Algoritmo2",
18      "description": "Descrição do algoritmo2.",
19      "command": "Comando - algoritmo2.",
20      "createdAt": "2022-01-17T13:30:48.807Z",
21      "updatedAt": "2022-01-17T13:30:48.807Z",
22      "order": 2
23    }
24  ]
25 }
```

All requests can be found in the Postman collection.