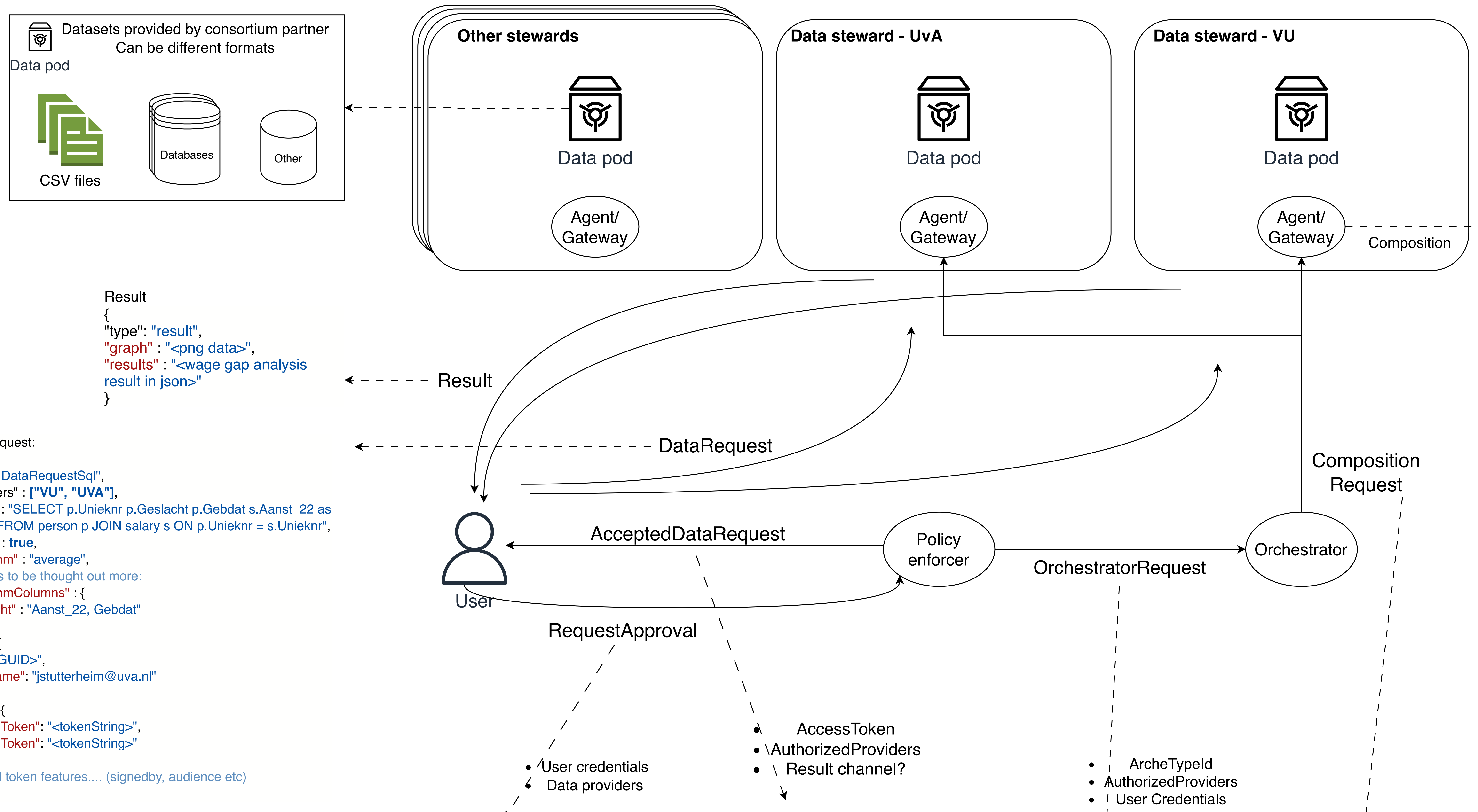


```
Result
{
"type": "result",
"graph" : "<png data>",
"results" : "<wage gap analysis
result in json>"
}
```

```
DataRequest:
{
"type": "DataRequestSql",
"providers" : ["VU", "UVA"],
"query" : "SELECT p.Unieknr p.Geslacht p.Gebdat s.Aanst_22 as
Salary FROM person p JOIN salary s ON p.Unieknr = s.Unieknr",
"graph" : true,
"algorithm" : "average",
// Needs to be thought out more:
"algorithmColumns" : {
"Geslacht" : "Aanst_22, Gebdat"
},
"user": {
"ID": "<GUID>",
"userName": "jstutterheim@uva.nl"
},
"auth" : {
"accessToken": "<tokenString>",
"refreshToken": "<tokenString>"
}
// Actual token features.... (signedby, audience etc)
}
```

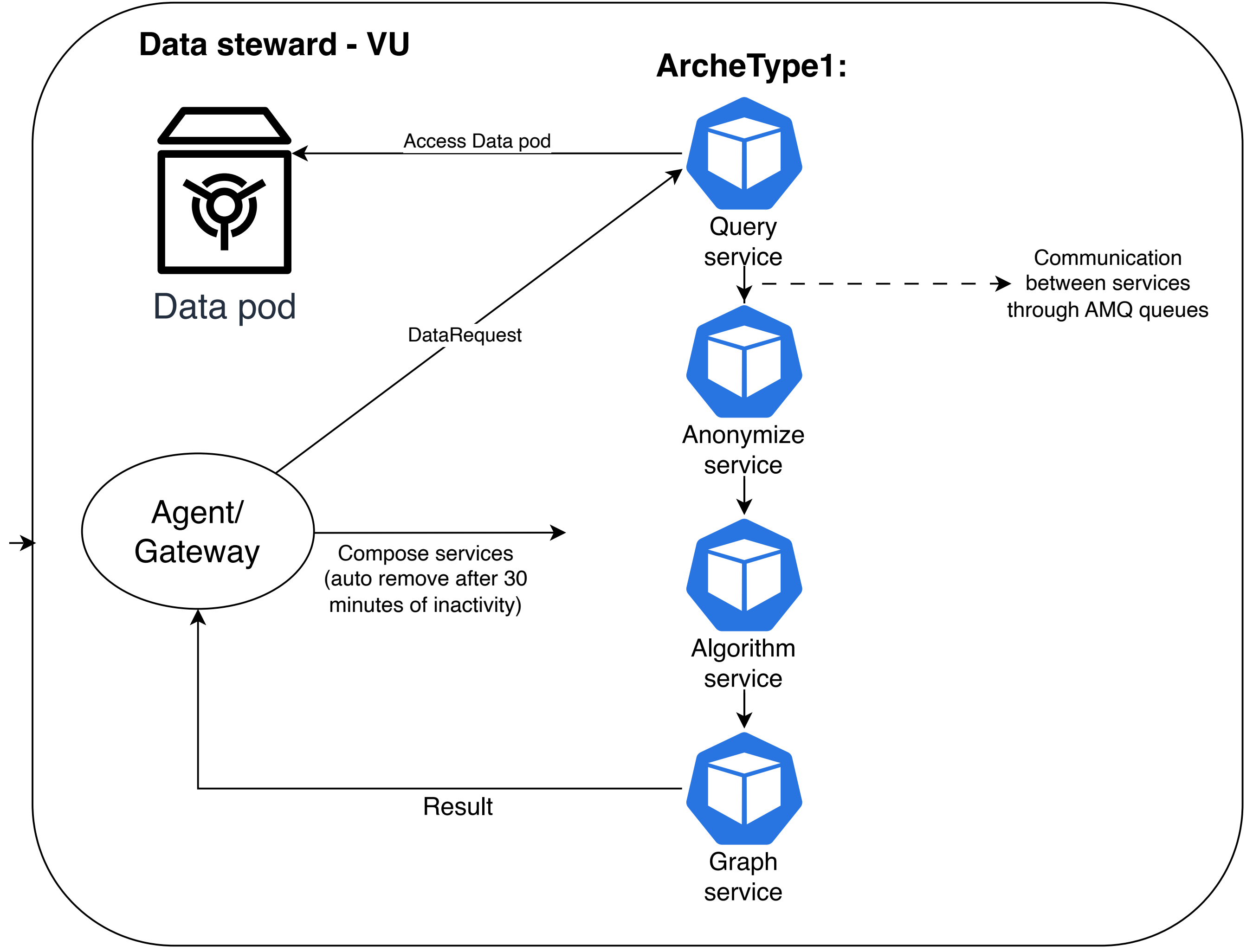
```
RequestApproval:

{
"type": "DataRequestSql",
"user": {
"ID": "<GUID>",
"userName": "jstutterheim@uva.nl"
// Actual loginToken features....
},
"providers": ["VU","UVA","RUG"]
}
```

```
AcceptedDataRequest:
{
"type": "DataRequestSql",
"user": {
"ID": "<GUID>",
"userName": "jstutterheim@uva.nl"
},
"auth" : {
"accessToken": "<tokenString>",
"refreshToken": "<tokenString>"
},
// Actual token features.... (signedby, audience etc)
"authorizedProviders" : {
"VU" : "<endpoint/queueName>",
"UVA" : "<endpoint/queueName>"
},
"resultChannel" : "<endpoint for results?>"
}
```

```
OrchestratorRequest
{
"archetypeId": "ArcheType1",
"user": {
"ID": "<GUID>",
"userName": "jstutterheim@uva.nl"
},
"providers": ["VU","UVA"],
"request_type": "datarequestsql"
}
```

```
CompositionRequest
{
"archetypeId": "ArcheType1",
"request_type": "datarequestsql",
"io_config": {
"service_io": {
"query_service": "start",
"anonymize_service": "query_service",
"algorithm" : "query_service"
},
"finish": "algorithm"
}
}
```



Query service

1. Based on the name of the data source (for example wage-dataset). The microservice can find out whether it is a CSV file, Mysql database or something else. (there would be some key-value store with that data)

2. If it is CSV, it needs to load the file into a sqllite DB. However, it would be good to save the csv hash and sqllite hash somewhere to compare so that this is only done if the CSV changes.

3. The provided query is executed, and the results passed to the next Microservice in JSON format.

4. Nice to have: Caching of sql results

Anonymize service

1. Recognize a type of anonymizing (Generalization, Masking or Perturbation), and get the columns of which this should be done.

2. The anonymization should identify the datatypes and take appropiate action (for example a birthdate can be in different formats)

3. After doing the work it should pass it on to the next microservice

Algorithm service

1. Accept JSON from the previous microservice.

2. Create generic functions that can calculate things like average, sum, graph, etc.

3. Based on the column input and request it can calculate these things

4. This should work for different type of datasets and columns.

Graph service

1. Convert data to a type of graph based on request and the github.com/go-echarts/go-echarts package

2. Convert the bar chart to SVG/PNG to send in the response or next MS.