Soma Job History

Release 0.1

Apr 17, 2020

Contents:

CHAPTER 1

Descrição

Esse microsserviço é responsável pela busca de jobs. Essa busca é provida por meio de uma interface REST.

CHAPTER 2

Guia de uso

2.1 Subindo o serviço com a execução do projeto no ambiente docker

Para rodar todo o ambiente junto com este serviço basta rodar o docker-compose projeto csgrid que inclui todos os serviços de back-end incluindo os do soma-job-history. São usadas as imagens que estão cadastradas no docker-hub.

2.1.1 Para sobrescreever a imagem que o composer usa deste serviço temos que fazer o seguinte

- Build do projeto java:

```
mvn package
```

- Docker build para gerar a nova imagem docker usando o projeto compilado e o Dockerfile do projeto:

```
docker build -t csbase/soma-job-history . --build-arg JAR=<path-do-arquivo-jar-
↪compilado>
```

Sem passar o parâmetro –build-arg JAR=<path-do-jar-compilado> o build será feito com um arquivo soma-job-history-*-SNAPSHOT.jar que estiver na pasta target.

- **Para rodar a imagem**
  - Podemos passar o *config* do hibernate através de uma variável de ambiente **HIBERNATE_CONFIG**. Caso não seja passado, será usado um *config default*.
  - O job-history depende do postgres rodando. Se ele estiver rodando em um docker, podemos passar a network do docker. A porta escutada é a **8086**.
- Exemplo de execução rodando o postgres no docker-compose do csgrid (com a parte do job-history comentada):

```
docker run --network docker_default -p 8086:8086 -ti csbase/soma-job-history
```

- Exemplo de execução passando um config para o hibernate:

```
docker run --network docker_default -p 8086:8086 --env HIBERNATE_CONFIG="hibernate.
↪connection.url=jdbc:postgresql://postgres:5432/postgres,hibernate.connection.
↪username=postgres,hibernate.connection.password=password,hibernate.hbm2ddl.
↪auto=validate" -ti csbase/soma-job-history
```

- Para executar todo o ambiente do backend com a imagem gerada pelo build, rodar o docker-compose do projeto csgrid.

2.1.2 Lançando versão release

Para lançar uma nova versão release, execute o comando abaixo, que atualizará o arquivo pom.xml e gerará uma tag com a nova versão.

```
mvn -DpreparationGoals=clean release:prepare
```

Ao terminar o comando, verifique se a tag foi criada e se o pipeline foi finalizado com sucesso. As etapas de **k8s_run_stable** e **k8s_run_release** precisam ser disparadas manualmente.

2.1.3 Dependências de outros sistemas ou microsserviços.

Todo o csgrid deve está executando para que o soma-job-history recebe informações de job em execução.

CHAPTER 3

Atributos de Qualidade

- Escalabilidade horizontal: O microserviço deve ter a capacidade ser replicada em um ambiente de cluster.
- Testabilidade: As camadas permitem que seus elementos sejam testados em isolado.
- Interoperabilidade: Foi adotado uma API REST que permite que microsserviços ou sistemas diferentes possam utilizar o soma job history com facilidade.

CHAPTER 4

Informações de deployment

4.1 Compile

Baixa e vincula todas as dependências. Compila o código fonte gerando um artefato executável.

4.2 Versioning

Descobre a versão atual do código. Necessário para indiciar a versão do artefato que será colocado no Nexus e Kubernetes.

4.3 Test

Testes de unidade e integração são executados nessa etapa.

4.4 Deploy

Faz o deploy da imagem docker no Registry interno (Nexus).

4.5 Run

Executa a imagem docker no kubernetes.

CHAPTER 5

Interface

5.1 Interface REST

A busca por job é provida por meio de uma interface REST.

A interface REST está especificada em OpenAPI (swagger) em conjunto com sua documentação:

- Especificação
- Documentação

Nas seções seguintes são apresentados e explicados operadores do parâmetro *q* para busca de jobs.

5.2 Operadores do parâmetro *q*

Operadores Básico	Descrição
==	Igual
!=	Não igual
=gt= >	Maior que
=ge= >=	Maior ou igual
=lt= <	Menor que
=le= <=	Menor ou igual
=in=	Pertence
=out=	Não Pertence

Operadores de Composição	Descrição
;	AND lógico
,	OR lógico

Pode ser adicionado o caractere * antes ou depois de textos para operador *like*.

5.3 Tipo data/hora

Formatos permitidos na requisição de campos do tipo data/hora:

- yyyy/MM/ddx
- yyyy/MM/dd HH:mmx
- yyyy/MM/dd HH:mm:ssx
- yyyy/MM/dd HH:mm:ss.SSSx

Onde **x** é o time zone (obrigatório na requisição) com os seguintes formatos permitidos:

- -hh
- -hhmm
- +hh
- +hhmm

Observação: o uso de operadores básicos para comparar valores do tipo data/hora considera o formato **yyyy/MM/dd HH:mm:ss.SSS** ao filtrar.

Time zone padrão da resposta:

- GMT0

5.4 Parâmetros com valor padrão para a busca

- *showParam*

  Descrição: Adiciona ao retorno os valores dos parâmetros do algoritmo

  Valor padrão: verdadeiro

- *limit*

  Descrição: Define o número de jobs retornados na busca

  Valor padrão: 1000

  Valor máximo permitido: 1000

5.5 Exemplos de query RSQL

5.5.1 Buscar job por descrição

- **RSQL:** `description=="Teste Yade com batch"`
- **URL:** `http://<host>/v1/jobs/history?q=description=="Teste Yade com batch"&offset=0&limit=20&locale=pt_BR`

5.5.2 Buscar job por owner

- **RSQL:** `jobOwner=="adm*"`
- **URL:** `http://<host>/v1/jobs/history?q=jobOwner=="adm*"&offset=0&limit=20&locale=pt_BR`

5.5.3 Buscar job por prioridade

- **RSQL:** `priority==1`
- **URL:** `http://<host>/v1/jobs/history?q=priority==1&offset=0&limit=20&locale=pt_BR`

5.5.4 Buscar job por nome do algoritmo

- **RSQL:** `algorithmName=="Teste_Progresso"`
- **URL:** `http://<host>/v1/jobs/history?q=algorithmName==Teste_Progresso&offset=0&limit=20&l`

5.5.5 Buscar job ordenadas pela data de submissão de modo ascendente

- **RSQL:** `attr=submissionTime&asc=true`
- **URL:** `http://<host>/v1/jobs/history?offset=0&limit=20&attr=submissionTime&asc=true&local`

5.5.6 Buscar job ordenadas pela data de submissão de modo não ascendente

- **RSQL:** `attr=submissionTime&asc=false`
- **URL:** `http://<host>/v1/jobs/history?offset=0&limit=20&attr=submissionTime&asc=false&loca`

5.5.7 Buscar jobs executados em um determinada máquina e por um determinado dono

- **RSQL:** `executionMachine==40100;jobOwner=="admin"`
- **URL:** `http://<host>/v1/jobs/history?q=executionMachine==40100;jobOwner=="admin"&offset=0&limit=20&locale=pt_BR`

5.5.8 Buscar job com prioridade ou que foram rodadas por um determinado dono

- **RSQL:** `priority==1,jobOwner=="admin"`
- **URL:** `http://<host>/v1/jobs/history?q=priority==1,jobOwner=="admin"&offset=0&limit=20&lc`

5.5.9 Buscar jobs com descrição usando uma expressão regular

- **RSQL:** `description==*MS2`
- **URL:** `http://<host>/v1/jobs/history?q=(isDeleted==false);(description==*MS2)&offset=0&limit=20&locale=pt_BR`

5.5.10 Buscar jobs de um determinado algoritmo e que tenham sido cancelados

- **RSQL:** `algorithmName=="Yade";exitStatus=="KILLED"`
- **Observação:** O valor de "exitStatus" precisa ser todo maiúsculo.

- **URL:** `http://<host>/v1/jobs/history?q=algorithmName=="Yade";exitStatus=="KILLED"&offset=0&limit=20&locale=pt_BR`

5.5.11 Buscar jobs que possuam determinado parâmetro

- **RSQL:** `paramLabel=="Sleeps"`
- **URL:** `http://<host>/v1/jobs/history?q=paramLabel=="Sleeps"&offset=0&limit=20&locale=pt_B`

5.5.12 Buscar jobs que possuam determinado valor para um determinado parâmetro

- **RSQL:** `(paramLabel=='Sleeps');(paramValue=='3')`
- **URL:** `http://<host>/v1/jobs/history?q=(paramLabel=='Sleeps');(paramValue=='3')&offset=0&limit=20&locale=pt_BR`

5.5.13 Buscar jobs que possuam determinados valores para determinado parâmetro

- **RSQL:** `(paramLabel=='Sleeps');(paramValue=in=('3','1'))`
- **URL:** `http://<host>/v1/jobs/history?q=(paramLabel=='Sleeps');(paramValue=in=('3','1'))&offset=0&limit=20&locale=pt_BR`

5.5.14 Buscar jobs que foram submetidos em uma determinada data/hora:

- **RSQL:** `submissionTime=='2019/06/12 21:24:25.342-0300'`
- **URL:** `http://<host>/v1/jobs/history?q=submissionTime=='2019/06/12 21:24:25.342-0300'&offset=0&limit=20&locale=pt_BR`
  - Observação: para fazer requisição via URL o time zone é obrigatório sendo **-** o caractere para time zone negativo

5.5.15 Buscar jobs que finalizaram no intervalo de duas datas/horas:

- **RSQL:** `endTime=gt='2019/06/12 21:00+03';endTime=lt='2019/06/12 22:00+03'`
- **URL:** `http://<host>/v1/jobs/history?q=endTime=gt='2019/06/12 21:00%2B03';endTime=lt='2019/06/12 22:00%2B03'&offset=0&limit=20&locale=pt_BR`
  - Observação: para fazer requisição via URL o time zone é obrigatório sendo **%2B** o encode para o caractere **+** referente ao time zone positivo

5.6 Estrutura do Job

No caso do histórico de jobs, o JSON que define a estrutura básica de "jobs" sobre a qual a expressão RSQL é feita é o seguinte:

```
{
        "jobId":"admi@test.CBNNGZHYE2",
        "groupId":"admi@test.CBNNGZHYE2",
        "projectId":"admin/testeYade",
        "jobOwner":"admin",
        "automaticallyMachineSelection":false,
        "submissionMachines":null,
        "numberOfProcesses":1,
        "numberOfProcessesByMachine":1,
        "submissionTime":"2019-06-12T21:24:25.342",
        "description":"test yade pela web",
        "priority":1,
        "multipleExecution":false,
        "jobType":"ALGORITHM",
        "numberOfAttempts":null,
        "executionMachine":"40100",
        "endTime":"2019-06-12T21:31:20.926",
        "exitCode":0,
        "guiltyNodeId":null,
        "exitStatus":"success",
        "cpuTime":0.0,
        "wallclockTime":412,
        "ramMemory":null,
        "statusHistory":[
            {
                "status":"SCHEDULED",
                "timestamp":"2019-06-12T21:24:25.342"
            },
            {
                "status":"UPLOADING",
                "timestamp":"2019-06-12T21:24:27.334"
            },
            {
                "status":"EXECUTING",
                "timestamp":"2019-06-12T21:24:27.358"
            },
            {
                "status":"DOWNLOADING",
                "timestamp":"2019-06-12T21:31:20.815"
            },
            {
                "status":"FINISHED",
                "timestamp":"2019-06-12T21:31:20.926"
            }
        ],
        "algorithms":[
            {
                "algorithmId":"Yade",
                "algorithmVersion":"5.0.0",
                "algorithmName":"Yade",
                "parameters":[
                    {
                        "id":86,
                        "parameterId":"working_dir",
                        "label":"Working Folder",
                        "type":"INPUT_FILE",
                        "value":[
```


(continues on next page)

(continued from previous page)


```
                        ".:DIRECTORY_TYPE"
                    ]
                },
                {
                    "id":87,
                    "parameterId":"input_file",
                    "label":"Script File (.py)",
                    "type":"INPUT_FILE",
                    "value":[
                        "teste/fiveParticles_v2.py:UNKNOWN"
                    ]
                },
                {
                    "id":88,
                    "parameterId":"batch_run",
                    "label":"Enable multiple runs",
                    "type":"BOOLEAN",
                    "value":[
                        "false"
                    ]
                },
                {
                    "id":89,
                    "parameterId":"batch_file",
                    "label":"Parameter Table File",
                    "type":"INPUT_FILE",
                    "value":[

                    ]
                },
                {
                    "id":90,
                    "parameterId":"parallel_computation",
                    "label":"Enable parallel computation",
                    "type":"BOOLEAN",
                    "value":[
                        "true"
                    ]
                },
                {
                    "id":91,
                    "parameterId":"number_of_cores",
                    "label":"The number of cores for a job",
                    "type":"INTEGER",
                    "value":[
                        "4"
                    ]
                }
            ]
        }
    ],
    "flowId":null,
    "flowVersion":null,
    "flowName":null,
    "lastModifiedTime":"2019-06-12T21:31:20.926"
}
```

Possíveis valores de status:

- **exitStatus**
  - UNKNOWN
  - SUCCESS
  - EXECUTION_ERROR
  - JOB_IDENTIFIER_NOT_FOUND
  - UNEXPECTED_MACHINE_ERROR
  - PROJECT_NOT_FOUND
  - FAILED_SETUP_EXECUTION_ENVIRONMENT
  - NO_PERMISSION
  - NO_MACHINE_AVAILABLE
  - KILLED
  - LOST
  - UNDEFINED
- **jobStatus**
  - SCHEDULED
  - INIT
  - UPLOADING
  - QUEUED
  - EXECUTING
  - DOWNLOADING
  - FINISHED
  - UNKNOWN

CHAPTER 6

Dependência de Outros Microsserviços

Não possui dependência direta de outro microsserviço. Contudo, consulta a base de dados alimentada pelo microsserviço de soma-job-history-consumer.

6.1 Protocolos de comunicação

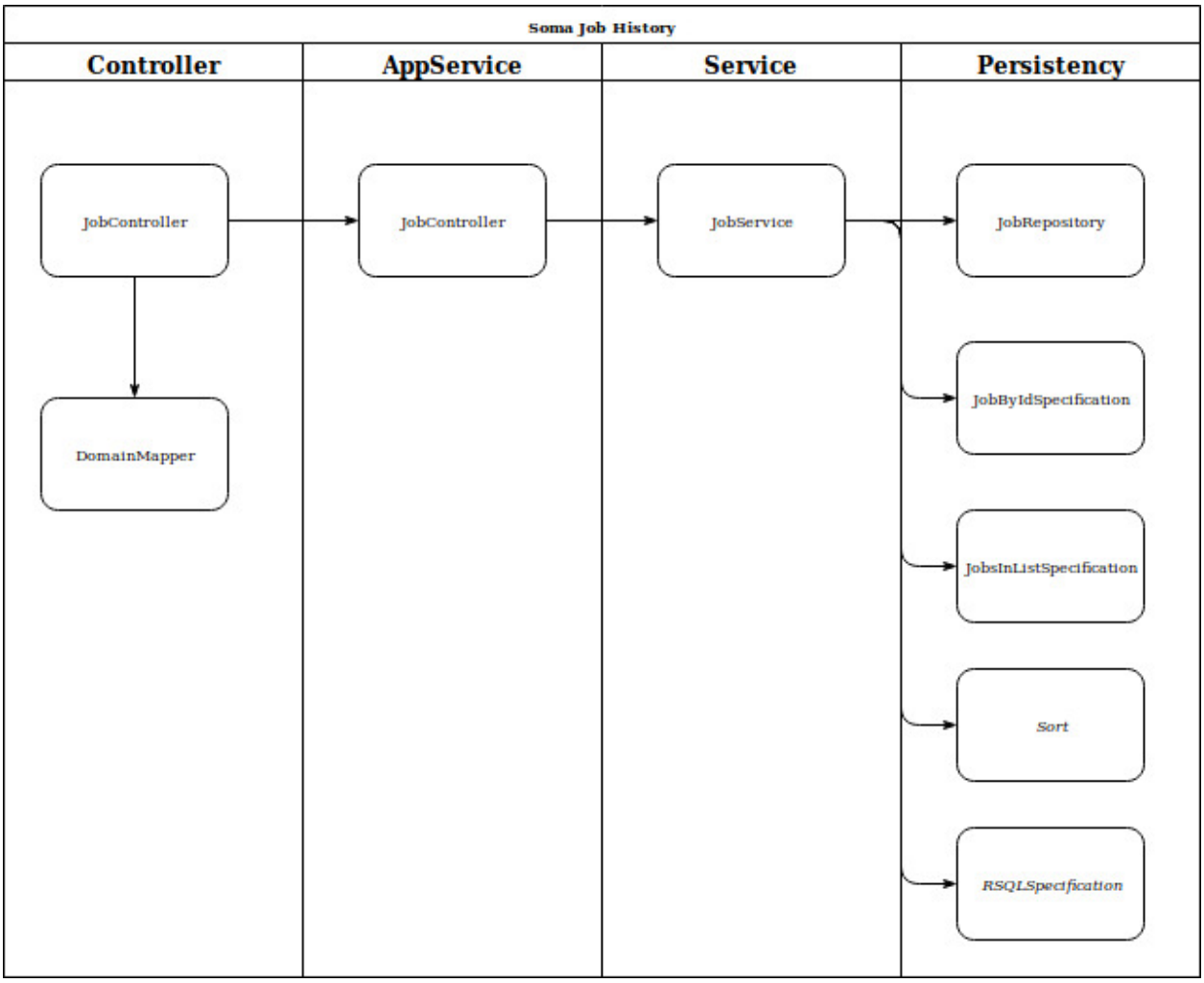
HTTP

- Fornece uma API para consultas de jobs. Detalhes sobre essa interface é descrito na Seção interface.

CHAPTER 7

Design

Diagrama exibindo as quatro camadas e classes mais relevantes em cada camada.


Soma Job History
Controller
AppService
Service
Persistency
JobController
JobController
JobService
JobRepository
DomainMapper
JobByIdSpecification
JobsInListSpecification
Sort
RSQLSpecification

CHAPTER 8

Clientes

Soma job history é uma versão mais geral que pode ser utilizado por diferentes clientes. Criado para integrar o sistema soma.

CHAPTER 9

Decisões arquiteturais

O microserviços soma job history foi construido em quatro camadas:

1. **Controller**: Implementa os endpoints da API REST. Quando uma chamada ao serviço é efetuada, no controlador exportado como recurso, é chamado o método correspondente aquela chamada da API REST. Exemplo de classe nessa camada: JobController

2. **AppService**: Serviço de aplicação redireciona a chamada (proveniente da camada de Controller) para o serviço de dados correspondente (camada Service). Exemplo de classe nessa camada: JobAppService

3. **Service**: É o serviço de dados. Responsável por contruir as queries e outras operadores especificamente relacionadas a dados (e.g, atualização de jobs como deletados). Exemplo de classe nessa camada: JobService

4. **Persistency**: Executa operações sobre a base de dados. Por exemplo, atualizações e buscas. Exemplo de classe nessa camada: JobRepository

Detalhes sobre as classes em cada camada e suas relações é descrito na seção Design.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search