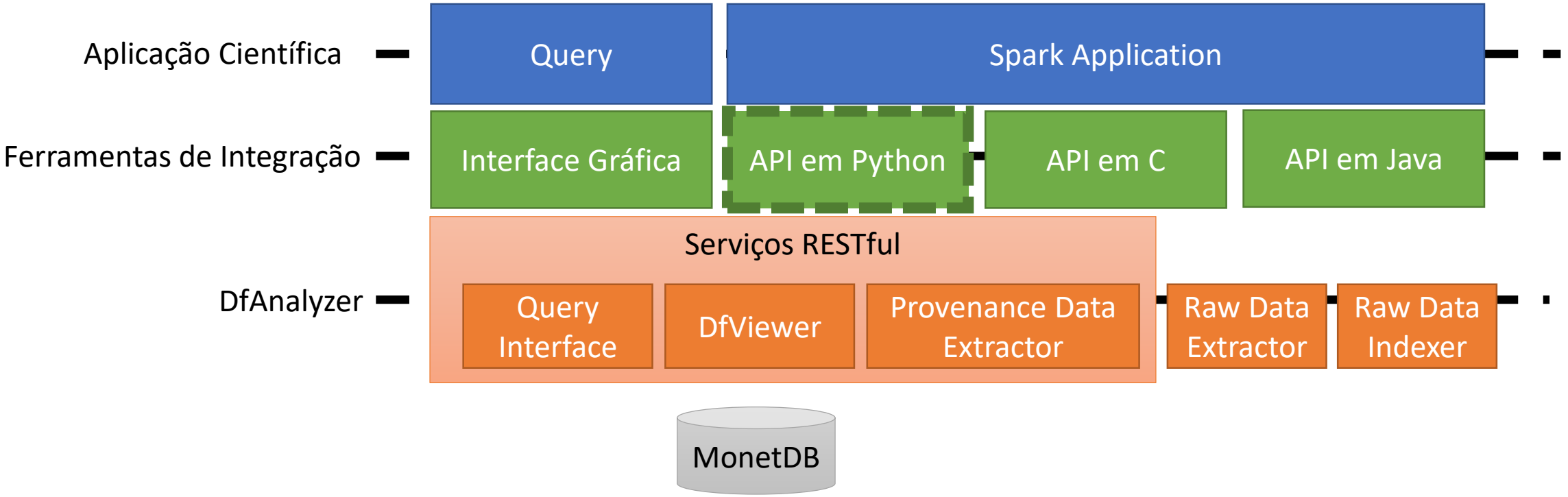
API em Python para Extração de Dados de Proveniência usando a DfAnalyzer

Vinícius Silva Campos

DfAnalyzer

- Ferramenta para a extração de dados de proveniência e dados científicos em simulações computacionais intensivas em dados
- Arquitetura



Proposta

- APIs para apoiar a captura de **dados de proveniência** e **dados científicos**
  - Linguagens de programação: Python e C++

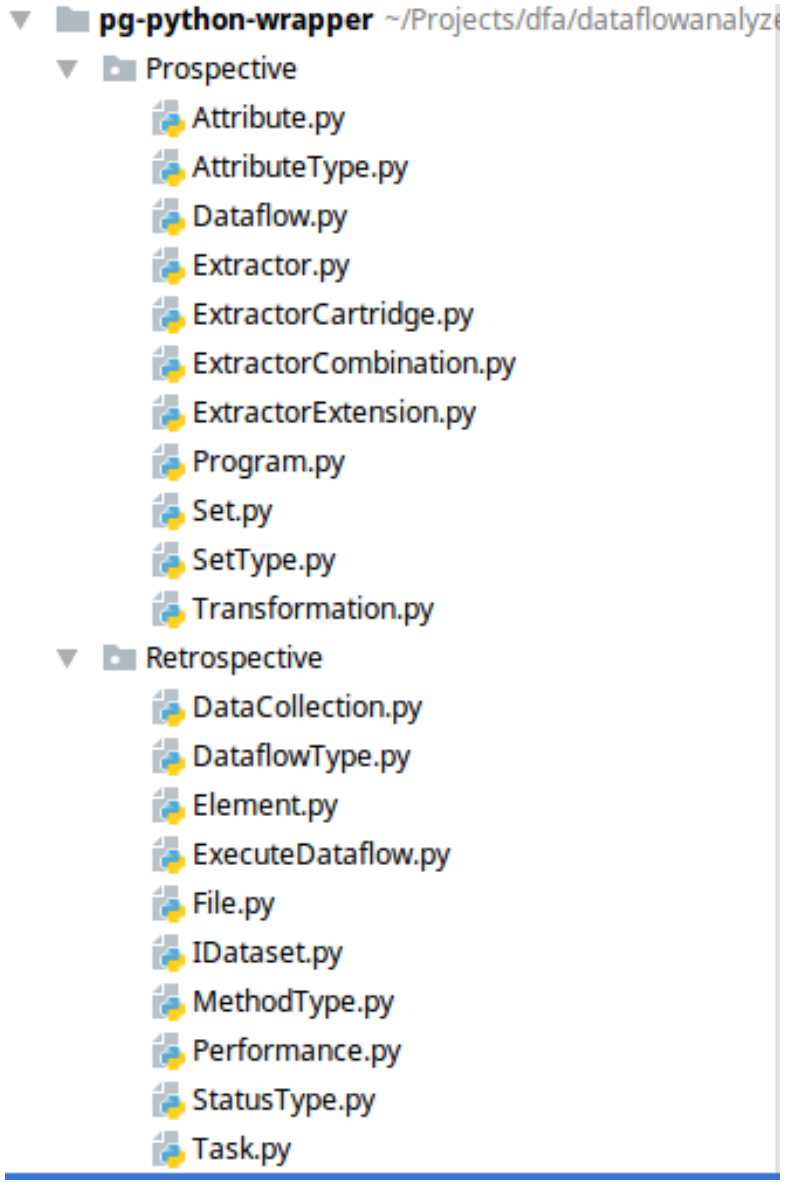
- Abordagem baseada na **instrumentação do código da aplicação científica**
  - Abstrair a complexidade para o cientista computacional
  - Facilidade de uso

API em Python

- Uso da abstração de fluxo de dados
- Classes para representar...
  - Dados de proveniência prospectiva

    e.g., dataflow, transformation, attribute
  - Dados de proveniência retrospectiva

    e.g., task, element, performance



API em Python – Conceitos de Fluxo de Dados

- Métodos para representar, de uma forma mais genérica, os registros de dados de proveniência prospectiva

```
# region ProspectiveProvenance
@staticmethod
def dataflow(tag):...

@staticmethod
def program(name, path):...

@staticmethod
def set(tag, type, attributes=None, extractors=None, extractor_combinations=None):
    return Set(tag, type, attributes, extractors, extractor_combinations)

@staticmethod
def extractor(tag, cartridge=None, extension=None):
    return Extractor(tag, cartridge, extension)

@staticmethod
def transformation(tag, programs=None, sets=None):
    return Transformation(tag, programs, sets)

@staticmethod
def attribute(tag, type, extractor=None):
    return Attribute(tag, type, extractor)

@staticmethod
def set_type():
    return SetType
```

API em Python – Carga de dados (Dataflow)

- Carga de dados da API por meio dos serviços RESTful da DfAnalyzer

```python
# region ProspectiveProvenance
@staticmethod
def dataflow(tag):...

@staticmethod
def program(name, path):...

@staticmethod
def set(tag, type, attributes=None, extractors=None, extractor_combinations=None):
    return Set(tag, type, attributes, extractors, extractor_combinations)

@staticmethod
def extractor(tag, cartridge=None, extension=None):
    return Extractor(tag, cartridge, extension)

@staticmethod
def transformation(tag, programs=None, sets=None):
    return Transformation(tag, programs, sets)

@staticmethod
def attribute(tag, type, extractor=None):
    return Attribute(tag, type, extractor)

@staticmethod
def set_type():
    return SetType
```

```python
class PDE(object):
    def __init__(self, configuration=None):...

    ProspectiveProvenance

    RetrospectiveProvenance

    # region RestAPI
    @staticmethod
    def ingest_dataflow_json(base_url, dataflow_json):
        url = base_url + '/pde/dataflow/json'
        r = requests.post(url, json=dataflow_json)
        print(r.status_code)
        print(r.content)

    @staticmethod
    def ingest_task_json(base_url, task_json):
        url = base_url + '/pde/task/json'
        r = requests.post(url, json=task_json)
        print(r.status_code)
        print(r.content)

        # endregion
```

API em Python – Carga de dados (Task)

- Carga de dados da API por meio dos serviços RESTful da DfAnalyzer

```python
# region RetrospectiveProvenance
@staticmethod
def task(tag, dataflow_tag, transformation_tag, resource, workspace, status, id, performances=[],
         output=None, error=None, sub_id=None, invocation=None, dependency=None, files=[], idatasets=[]):
    return Task(tag, dataflow_tag, transformation_tag, resource, workspace, status, id, performances, output, error, sub_id,
                invocation, dependency, files, idatasets)

@staticmethod
def file(name, path):
    return File(name, path)

@staticmethod
def performance(tag, method_type, start_time, task_id=None, description=None, invocation=None
                , task_sub_id=None, end_time=None):
    return Performance(tag, description, method_type, invocation, task_id, task_sub_id, start_time, end_time)

@staticmethod
def element(tag, values):
    return Element(tag, values)

@staticmethod
def idataset(tag, set, elements):
    return IDataset(tag, set, elements)

@staticmethod
def data_collection(idataset, tag=None):
    return DataCollection(idataset, tag)

@staticmethod
def status_type():
    return StatusType

@staticmethod
def method_type():
    return MethodType
```

```python
class PDE(object):
    def __init__(self, configuration=None):...

    ProspectiveProvenance

    RetrospectiveProvenance

    # region RestAPI
    @staticmethod
    def ingest_dataflow_json(base_url, dataflow_json):
        url = base_url + '/pde/dataflow/json'
        r = requests.post(url, json=dataflow_json)
        print(r.status_code)
        print(r.content)

    @staticmethod
    def ingest_task_json(base_url, task_json):
        url = base_url + '/pde/task/json'
        r = requests.post(url, json=task_json)
        print(r.status_code)
        print(r.content)

    # endregion
```

Tecnologias para a Integração com a ferramenta DfAnalyzer

- Serviços RESTful da DfAnalyzer
  - Java
  - Spring Boot
  - MonetDB (requisito para a base de dados)

- Extra
  - Uso do docker para criação de containers para os serviços RESTful e para a base de dados usando o MonetDB

Exemplo: Aplicação usando Apache Spark

- PySpark
- Jupyter Notebook
- PG-Python-Wrapper

- Código-fonte disponível em…
  - https://drive.google.com/open?id=1uLQhiB-N29BbolBgrZ5njerRaXXrLIbF

Obrigado!

API em Python para Extração de Dados de Proveniência usando a DfAnalyzer

Vinícius Silva Campos