escola
britânica de
artes criativas
& tecnologia

Profissão: Analista de dados

escola britânica de artes criativas & tecnologia

4º Projeto: Pipeline de dados do Telegram

GUIA DA AULA 3

Colete dados

- **Mensagem**
- **Wrangling**

Acompanhe aqui os temas que serão tratados na videoaula

Mensagem

Uma mensagem recuperada via API é um dado semiestruturado no formato JSON com algumas chaves mandatórias e diversas chaves opcionais, estas últimas presentes (ou não) dependendo do tipo da mensagem. Por exemplo, mensagens de texto apresentam a chave `text` enquanto mensagens de áudio apresentam a chave `audio`. Neste projeto vamos focar em mensagens do tipo texto, ou seja, vamos ingerir as chaves mandatórias e a chave `text`.

Nota: A lista completa das chaves disponíveis pode ser encontrada no *link* https://core.telegram.org/bots/api#message.

Exemplo:

In [ ]:

```
%%writefile telegram.json
{
    "update_id": 123,
    "message": {
        "message_id": 1,
        "from": {
            "id": 321,
            "is_bot": false,
            "first_name": "Andre"
        },
        "chat": {
            "id": -789,
            "type": "group"
        },
        "date": 1640995200,
        "text": "Ola, mundo!"
    }
}
```

Descrição:

chave	tipo valor	opcional	descrição
updated_id	int	não	id da mensagem enviada ao bot
message_id	int	não	id da mensagem enviada ao grupo
from_id	int	sim	id do usuário que enviou a mensagem
from_is_bot	bool	sim	se o usuário que enviou a mensagem é um bot
from_first_name	str	sim	primeiro nome do usário que enviou a mensagem
chat_id	int	não	id do *chat* em que a mensagem foi enviada
chat_type	str	não	tipo do *chat*: private, group, supergroup ou channel
date	int	não	data de envio da mensagem no formato unix
text	str	sim	texto da mensagem

Wrangling

Vamos denormalizar o conteúdo da mensagem semiestruturado no formato JSON utilizando apenas Python nativo, ou seja, sem o auxílio de pacotes, como Pandas.

Para começar, vamos carregar o arquivo `telegram.json` utilizando o pacote nativo json.

In [ ]:
```
import json

with open('telegram.json', mode='r', encoding='utf8') as fp:
  data = json.load(fp)
  data = data["message"]
```

In [ ]:
```
print(json.dumps(data, indent=2))
```

Vamos, então, utilizar um laço de repetição para varrer todas as chaves do arquivo e selecionar apenas as de interesse. Caso a mensagem não possua a chave `text`, ela será criada com o valor igual a `None`. Além disso, vamos adicionar duas chaves de tempo para indicar o momento em que o dado foi processado: `context_date` e `context_timestamp`.

In [ ]:

```
from datetime import datetime

date = datetime.now().strftime('%Y-%m-%d')
timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

parsed_data = dict()
```

```
for key, value in data.items():

    if key == 'from':
        for k, v in data[key].items():
            if k in ['id', 'is_bot', 'first_name']:
              parsed_data[f"{key if key == 'chat' else 'user'}_{k}"] = [v]

    elif key == 'chat':
        for k, v in data[key].items():
            if k in ['id', 'type']:
              parsed_data[f"{key if key == 'chat' else 'user'}_{k}"] = [v]

    elif key in ['message_id', 'date', 'text']:
        parsed_data[key] = [value]

if not 'text' in parsed_data.keys():
  parsed_data['text'] = [None]

parsed_data['context_date'] = [date]
parsed_data['context_timestamp'] = [timestamp]
```

In [ ]:
```
for k, v in parsed_data.items():
  print(f"{k}: {v}")
```

Por fim, vamos utilizar o pacote Python PyArrow para criar uma tabela com os dados processado que, posteriormente, pode ser facilmente persistida em um arquivo no formato Apache Parquet.

In [ ]:
```
import pyarrow as pa

table = pa.Table.from_pydict(mapping=parsed_data)
```

In [ ]:
```
table
```