

# Teste para seleção de Engenheiro de Dados

Olá candidato,

Esse teste consiste na aplicação de conhecimentos de engenharia de dados para um cenário de ingestão, curadoria, processamento e consumo de dados.

O objetivo do teste é realizar as etapas envolvidas na construção de uma solução Big Data. Para tanto, uma arquitetura simplificada do *Data Lake* (ver Figura 1) será usada como referência para as atividades.

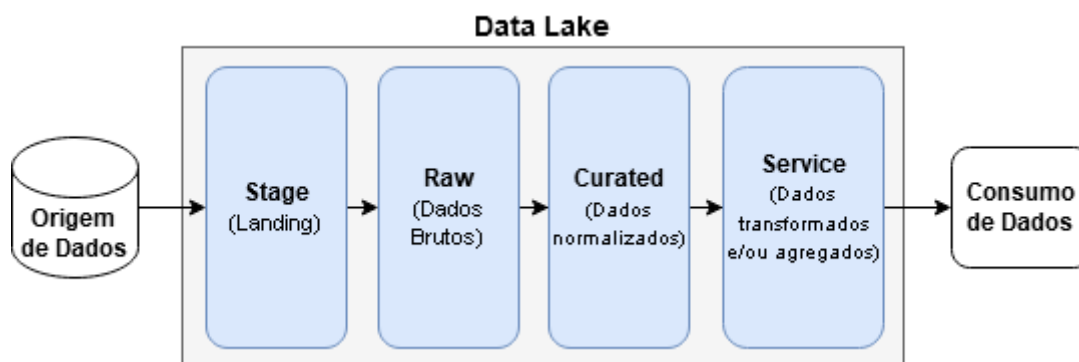


Figura 1: Arquitetura simplificada de Data Lake

## Atividade 1: Ingestão

**Origem dos Dados:** <https://repositoriodatasharingfapesp.uspdigital.usp.br/handle/item/98>

Pasta de arquivos, com dados anonimizados de pacientes que fizeram teste para COVID-19 a partir de 1/11/2019, compactada em formato *zip*. Contem 2 arquivos em formato CSV e 1 arquivo em formato XLXS: (1) Planilha com dados anonimizados sobre pacientes que fizeram teste para o COVID-19 (sorologia ou PCR) no Hospital Israelita Albert Einstein, incluindo: identificador anonimizado do paciente, gênero, país, estado, município e região de residência; (2) Respectivos resultados de exames laboratoriais, contendo dentre outros o identificador anonimizado do paciente; e (3) Dicionário de dados: planilha em que cada aba descreve respectivamente todos os campos das planilhas de Pacientes e de Exames. Pacientes e seus Exames são interligáveis pelo identificador anonimizado do paciente. Devido ao grande numero de registros, nem sempre as planilhas poderão ser importadas diretamente para EXCEL ou semelhantes.




 EINSTEIN\_Dicionario\_2.xlsx  
 EINSTEIN\_Exames\_2.csv  
 EINSTEIN\_Pacientes\_2.csv

Figura 2: conteúdo do arquivo *zip* da origem

**Entrada:** arquivo **zip** da origem de dados descrita

**Atividades:**

- Criar e organizar as camadas do *data lake* (*stage*, *raw*, *curated* e *service*) no sistema de arquivos do seu Sistema Operacional.
  - Levar em consideração a separação por origem e tabela/arquivo (*stage*, *raw* e *curated*), e também o domínio específico (*service*)
- Extração e movimentação de dados: Colocar os arquivos CSV na camada *stage*.
  - Levar em consideração a data de movimentação
- Usando Spark, faça a ingestão na camada *raw* a partir dos dados que estão na camada *stage*
  - Adicionar um campo DT\_CARGA de tipo DATE para a data de ingestão;
  - Os dados na camada *Raw* devem estar em formato *parquet* e compressão *snappy* ;
  - A codificação dos dados deve ser UTF-8;
  - Criar uma rotina para mover os arquivos CSVs, após a ingestão, para uma pasta de histórico (que inclui a data de movimentação) na camada *stage*.

**Resultado esperado:**

- Na camada *Stage*: uma pasta de histórico (com data de movimentação) que tem os arquivos originais ingeridos na camada *Raw*;
- Na camada *Raw*: arquivos por base, tabela/arquivo em formato *parquet*, compressão *snappy* e codificação UTF-8

## Atividade 2: Normalização e aplicação de regras de qualidade

**Entrada:** arquivos *parquet* de exames e pacientes na camada *Raw*

**Atividades:**

- Usando Spark, para cada base/tabela/arquivo normalizar e aplicar regras de qualidade. Seguir a especificação da Tabela 1;
- Colocar numa pasta **hot** na camada *Curated*, os registros normalizados e que passaram as regras de qualidade;
- Colocar numa pasta **rejected** na camada *Curated*, os registros que não passaram as regras de qualidade.

Tabela/arquivo	Normalização	Regras de Qualidade
Pacientes	- No campo CD_MUNICIPIO, converter valores MMMM para HIDDEN - No campo CD_CEPREDUZIDO, convertes valores CCCC para HIDDEN	Campos obrigatórios, não vazios e não NULL: ID_PACIENTE, AA_NASCIMENTO, CD_PAIS, CD_MUNICIPIO
Exames	- Campo DE_ANALITO deve ser de tipo DATE - No campo DE_RESULTADO, converter valores para minúsculo e garantir que cada palavra esteja separada por apenas um espaço ( <b>usar UDF</b> para este caso)	Campos obrigatórios, não vazios e não NULL: ID_PACIENTE, DT_COLETA, DE_ORIGEM, DE_EXAME, DE_RESULTADO,

Tabela 1: Regras de normalização e qualidade

**Resultado esperado:**

- Arquivos parquet com compressão *snappy* na camada *Curated* por base e tabela/arquivo;
- Se os registros passaram as regras de qualidade e não deram erro devem estar na pasta **hot** de outro modo na pasta **rejected**.

## Atividade 3: Transformação e processamento

**Entrada:** arquivos parquet de exames e pacientes na camada *Curated*

**Atividades:**

- Usando Spark, cruzar os dados de exames e pacientes, das respectivas pasta **hot** na camada *Curated*, para gerar um novo conjunto de dados e colocar os resultados numa pasta **exames\_por\_pacientes** na camada **Service**;
- O novo conjunto de dados deve ser gerado cruzando pelo campo ID\_PACIENTE. A seguir mais detalhes das colunas:
  - Dos pacientes:
    - Manter as colunas exceto AA\_NASCIMENTO e DT\_CARGA
    - Adicionar um campo VL\_IDADE calculado a partir do campo AA\_NASCIMENTO
  - Dos exames:
    - Manter todas as colunas exceto DT\_CARGA
  - Particionar pelas colunas CD\_PAIS e CD\_UF
- Gerar mais um novo conjunto de dados **exames\_por\_paciente\_sp** na camada *Service*, gerado a partir de **exames\_pacientes** para o estado de São Paulo (campo CD\_UF = 'SP').

**Resultado esperado:**

- Arquivos parquet numa pasta **exames\_por\_pacientes** na camada **Service** e particionado pelas colunas CD\_PAIS e CD\_UF;
- Arquivos parquet numa pasta **exames\_por\_pacientes\_sp** na camada **Service**.

## Atividade 4: Uso dos dados transformados

**Entrada:** arquivos parquet da pasta **exames\_por\_pacientes\_sp** na camada *Service*.

**Atividades:** usando Pandas e bibliotecas de visualização em Python:

- Obter um novo Dataframe com registros que tenham resultados de tipo qualitativo: filtrar pelo campo DE\_RESULTADO os valores que não sejam numéricos;
- A partir de Dataframe anterior, visualizar um histograma mostrando a quantidade (contagem) de registros por DE\_ANALITO ordenados decendentemente pela quantidade;

### Resultado esperado:

Histograma mostrando quantidade de registros por DE\_ANALITO a partir dos dados na pasta **exames\_por\_pacientes\_sp** na camada *Service*.

### Critérios de avaliação

- Boas práticas de programação: nomeação de variáveis, modularização, otimização, segurança, etc;
- Organização do projeto: estrutura do projeto, documentação e uso de controle de versão;
- Coerência: avaliar se os requisitos foram atendidos;
- Bônus - Controle de Qualidade: testes automatizados.

### Processo de submissão

O desafio deve ser entregue num repositório Git privado enviando URL e modo de acesso por e-mail. O repositório deve conter o código fonte, documentação e evidências para cada atividade.

Bom trabalho!