PUC

Monografias em Ciência da Computação

INF2102 - Projeto Final de Programação

Sugestão de poços de petróleo correlatos através de clusterização

Waldir José Pereira Junior

Matrícula 1820998

(wjunior@inf.puc-rio.br)

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Sumário

Figuras

Resumo

Em uma aplicação web desenvolvida para configuração de escopo de atividades de perfuração em poços de petróleo (que chamaremos apenas por Sistema CW), existe uma ferramenta básica para busca de poços semelhantes, de forma a permitir a cópia (clonagem) do escopo de atividades entre poços. Porém esta ferramenta não envolve nenhuma técnica de inteligência artificial para a busca por poços semelhantes, além de ser extremamente lenta.

A ideia deste trabalho é construir uma nova ferramenta, composta de:

1. Algoritmo de clusterização para identificação automática de poços correlatos (em Python); e
2. Implementação de uma nova ferramenta para substituir a atual (em C# no backend e Angular no frontend), consultando esta base de poços correlatos automaticamente.

Introdução

A construção de um poço de petróleo é uma atividade muito custosa [1] e que demanda uma série de etapas a serem executadas [2]. Dentre estas etapas está a de perfuração, que, por sua vez, é uma área vasta de conhecimento e pesquisa.

Durante a perfuração de um poço de petróleo muitas atividades são desenvolvidas e quanto mais profundo o reservatório, mais custosa e complexa será a operação. Sendo assim, o planejamento das atividades de perfuração se tornou uma tarefa muito importante dentro do cenário de construção de poços.

Este trabalho consiste na criação de uma ferramenta para auxiliar o planejamento de perfuração de poços utilizando técnicas de aprendizado de máquina, ou seja, será uma ferramenta de inteligência artificial aplicada. O resultado será uma nova funcionalidade desenvolvida no Sistema CW (utilizado pelos projetistas para planejamento da perfuração dos poços de petróleo), que auxilie o usuário mostrando outros poços já planejados (e até perfurados) que sejam muito parecidos com o poço que ele está planejando no momento.

Para isto, utilizarei técnicas de engenharia de software para integrar soluções desenvolvidas em três linguagens de programação diferentes (Python [3], C# [4] e Angular/TypeScript [5] e [6]), três sistemas de armazenamento diferentes (sistema de arquivos, banco de dados relacional Oracle [7] e banco de dados não relacional MongoDB [8]), além das ferramentas de aprendizado de máquina disponíveis no scikit-learn [9] (K-Medoids [10] e K-Means [11]).

Na próxima seção discutirei os aspectos de arquitetura da solução. Em seguida, mostrarei algumas evidências da utilização e do funcionamento. Logo depois mostrarei como os testes foram realizados e, por fim, a conclusão do trabalho.

Todo o fonte do módulo de clusterização encontra-se em repositório público (https://github.com/waldirpereira/python_clustering).

Arquitetura da solução

Como dito nas seções anteriores, a aplicação foi desenvolvida em dois módulos:

1. Módulo de clusterização

Este módulo é o responsável por descobrir, a partir de uma lista de poços que tiveram suas perfurações já planejadas e/ou perfuradas, quantos grupos (clusters) de poços devem ser gerados e em qual cluster está cada poço.

A partir deste momento, o módulo gerará duas saídas:

a) Gravação do modelo de clusterização em arquivo binário, no sistema de arquivos; e
b) Gravação dos clusters e os nomes dos poços de cada cluster no MongoDB.

Abaixo o diagrama de sequência [12] do módulo de clusterização:

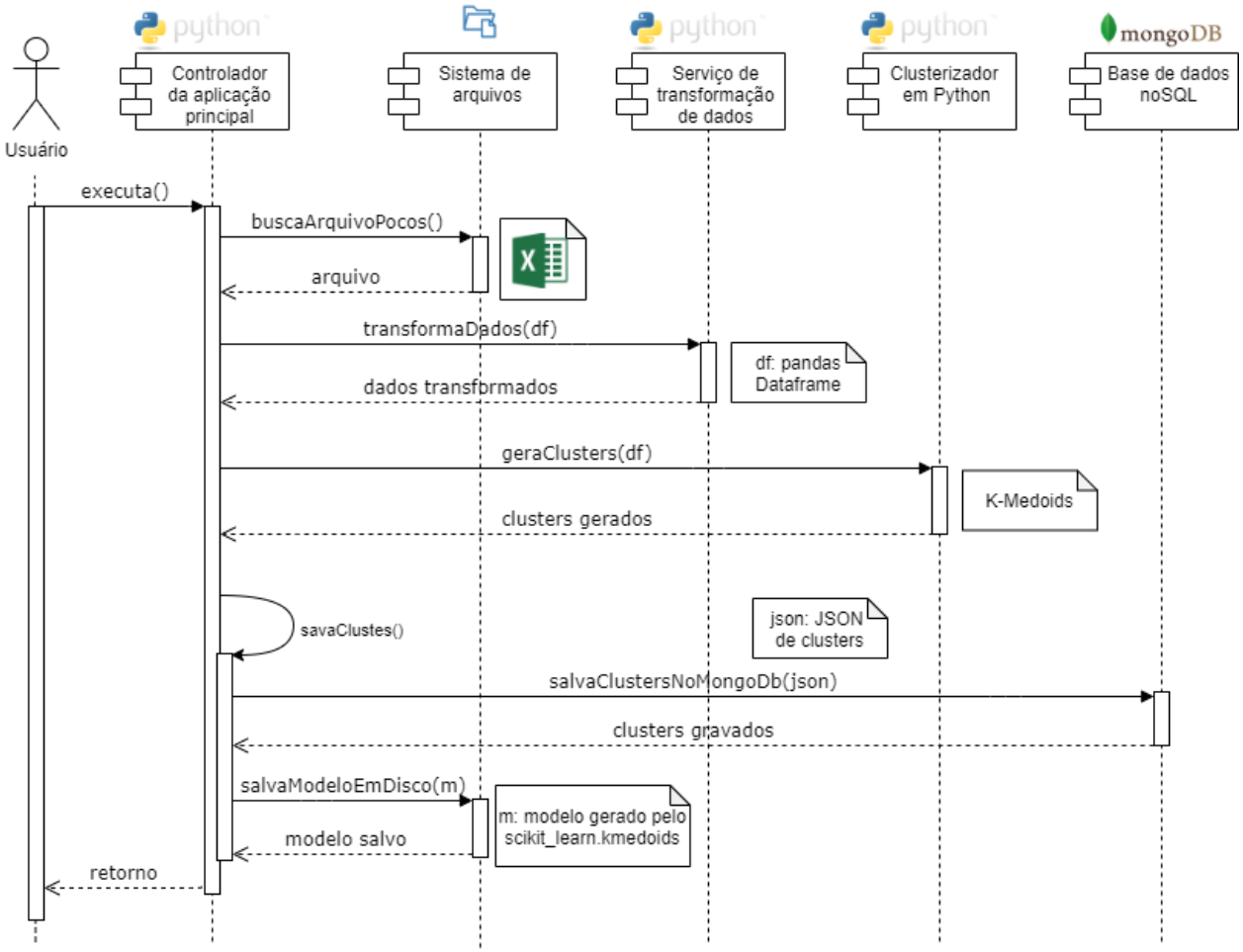


Figura 1 - Diagrama de sequência da aplicação de clusterização

Este módulo foi inteiramente escrito em Python e seu código está explicado no Anexo I deste documento, através da saída do Jupyter Notebook [13].

Para a criação deste módulo, implementei um serviço de transformação de dados, que será reutilizado no módulo seguinte (o de aplicação). Desta forma reduzi a duplicidade de código e permiti que os métodos do serviço de transformação fossem testados isoladamente.

Abaixo como funciona cada uma das partes do módulo de clusterização:

1.1. Leitura do cronograma em Excel

Nesta parte a planilha de saída do Sistema CW é importada e transformada em um Dataframe Pandas [14].

1.2. Tratamentos

O arquivo importado é composto por atividades/tarefas de todas as etapas de construção de poços. Primeiramente somente as atividades de perfuração serão mantidas (removendo atividades de completação, workover e avaliação).

Após isso, o dataframe é agrupado por poços.

1.3. Preparação para clusterização

Nesta etapa os dados textuais são transformados em categóricos e, após, estes são transformados em numéricos, criando uma nova coluna para cada possível valor, resultando em um dataframe contendo apenas valores 0 ou 1.

1.4. Calculando o K ideal (utilizando K-Means)

Agora o K-Means foi aplicado para que o gráfico para análise de "cotovelo" fosse gerado e, assim, nos mostrasse o K ideal, ou seja, o número ideal de clusters para nossa massa de dados.

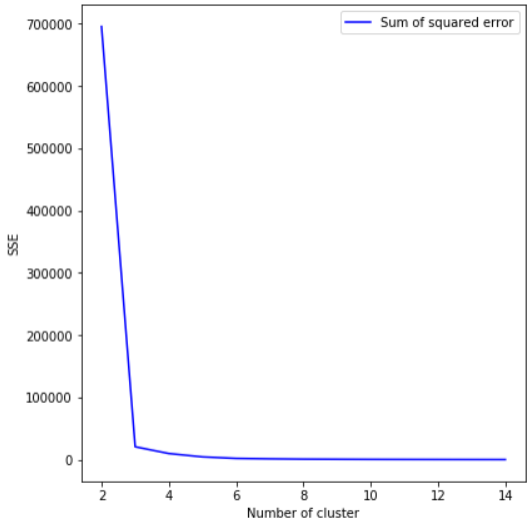


Figura 2 - Gráfico de cotovelo

Pelo gráfico podemos observar que o número ideal [15] de clusters é 3.

1.5. Calculando clusters através do K-Medoids

O K-Medoids [16] foi escolhido por manter os centroides representativos, ou seja, poços existentes serão escolhidos como centroides de cada cluster.

Primeiramente 3 poços foram escolhidos aleatoriamente. Depois, utilizei o algoritmo KMedoids da biblioteca sckit-learn-extra. Os clusters foram calculados, então.

Inicialização

Escolhendo aleatoriamente k medoids

In [14]:
```
df_size = len(df_numerico)

np.random.seed(123)
initial_medoids = np.random.randint(df_size, size=3).tolist()

initial_medoids
```

Out[14]:
```
[2, 28, 17]
```

Calculando clusters através do K-Medoids

In [15]:
```
x = df_numerico.iloc[:].values
```

In [16]:
```
from sklearn_extra.cluster import KMedoids
from pyclustering.utils import timedcall, distance_metric, type_metric
metric = distance_metric(type_metric.EUCLIDEAN_SQUARE, data=x)
kmedoids = KMedoids(n_clusters=3, random_state=0,metric=metric).fit(x)
labels = kmedoids.labels_

clusters_2 = []
for i in range(kmedoids.n_clusters):
    clusters_2.append([])
i = 0
for label in labels:
    clusters_2[label].append(i)
    i = i + 1
clusters_2
```

Out[16]:
```
[[1, 2, 3, 5, 7, 8, 9, 11, 14, 15, 16, 17, 18, 24, 25, 26, 27, 28, 29, 30, 31],
 [0, 10, 13, 19, 20, 21, 22, 23],
 [4, 6, 12, 32, 33]]
```

Figura 3 - Calculando clusters através do K-Medoids

1.6. Salvando dados dos clusters no MongoDB

Através da clusterização realizada no passo anterior, busquei os nomes dos poços de cada cluster e os salvei em uma MongoDB, em uma base de dados utilizada pelo Sistema CW.

Salvando dados dos clusters no MongoDB

In [17]:
```
from services.mongoDbService import MongoDbService
from projeto import configuracoes

cluster_id = 0
clusters_json = []
for cluster_indexes in clusters_2:
    listaNomesPGs = (df_ajustado.take(cluster_indexes)['Ponto geográfico']).tolist()
    clusters_json.append({'Cluster': cluster_id, 'NomesPontosGeograficos': listaNomesPGs})
    cluster_id = cluster_id + 1

mongoDbService = MongoDbService(configuracoes)
documentos_inseridos = mongoDbService.grava(clusters_json)
print("documentos_inseridos: ", documentos_inseridos)
```

```
documentos_inseridos:  <pymongo.results.InsertManyResult object at 0x000001D4EC4C0D88>
```

Figura 4 - Salvando clusters no MongoDB

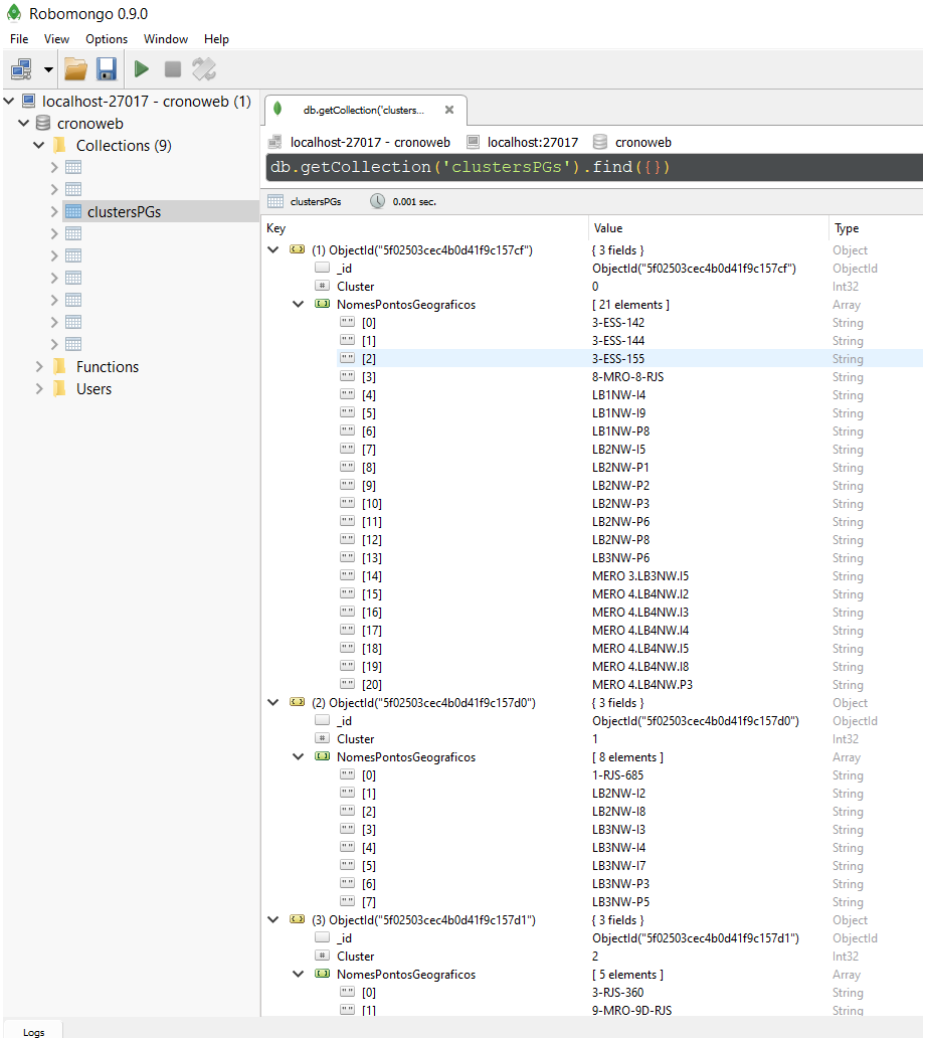


Figura 5 - Clusters e poços gravados no MongoDB

1.7. Salvando o modelo em arquivo

Já o modelo de clusterização foi salvo em arquivo utilizando a biblioteca “pickle”, para posteriormente ser lido pelo Sistema CW.

Salvando o modelo em arquivo

In [19]:
```
#https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/
import pickle

# save the model to disk
filename = 'finalized_model.sav'
filehandler = open(filename, 'wb')
pickle.dump(kmedoids, filehandler)
filehandler.close()
```

Figura 6 - Salvando modelo de clusterização em arquivo

1.8. Testando os pontos no modelo em memória e obtendo resultado correto

Por fim, para testar o carregamento do modelo pelo sistema de arquivos, busquei 3 poços no dataframe utilizado para a clusterização e chequei quais os clusters o modelo em memória retornaria.

Testando os pontos no modelo em memória e obtendo resultado correto

In [24]:
```
result_cluster_0 = kmedoids.predict(df_de_registro_teste_transformado_cluster_0)
result_cluster_1 = kmedoids.predict(df_de_registro_teste_transformado_cluster_1)
result_cluster_2 = kmedoids.predict(df_de_registro_teste_transformado_cluster_2)
print("cluster do result_cluster_0: ", result_cluster_0)
print("cluster do result_cluster_1: ", result_cluster_1)
print("cluster do result_cluster_2: ", result_cluster_2)
```

```
cluster do result_cluster_0:  [0]
cluster do result_cluster_1:  [2]
cluster do result_cluster_2:  [1]
```

Figura 7 - Resultado de clusters com poços aleatórios no modelo em memória

1.9. Testando os pontos no modelo buscado em arquivo e obtendo resultado correto

Verifiquei quais clusters o modelo buscado do sistema de arquivos apresentaria para cada um dos poços buscados no passo anterior. E os clusters foram os mesmos, mostrando que a serialização do modelo em arquivo funcionou corretamente.

Testando os pontos no modelo buscado em arquivo e obtendo resultado correto

In [26]:
```
filename = 'finalized_model.sav'
filehandler = open(filename, 'rb')
loaded_model = pickle.load(filehandler)
filehandler.close()
result_saved_model_cluster_0 = loaded_model.predict(df_de_registro_teste_transformado_cluster_0)
result_saved_model_cluster_1 = loaded_model.predict(df_de_registro_teste_transformado_cluster_1)
result_saved_model_cluster_2 = loaded_model.predict(df_de_registro_teste_transformado_cluster_2)
print("cluster do result_saved_model_cluster_0 (que deve ser o mesmo de result_cluster_0): ", result_saved_model_cluster_0)
print("cluster do result_saved_model_cluster_1 (que deve ser o mesmo de result_cluster_1): ", result_saved_model_cluster_1)
print("cluster do result_saved_model_cluster_2 (que deve ser o mesmo de result_cluster_2): ", result_saved_model_cluster_2)
```

```
cluster do result_saved_model_cluster_0 (que deve ser o mesmo de result_cluster_0):  [0]
cluster do result_saved_model_cluster_1 (que deve ser o mesmo de result_cluster_1):  [2]
cluster do result_saved_model_cluster_2 (que deve ser o mesmo de result_cluster_2):  [1]
```

Figura 8 - Resultado de clusters com poços aleatórios no modelo recuperado de arquivo

Desta forma, o módulo de clusterização foi finalizado, com um modelo de clusterização salvo em arquivo e o resultado de clusters e seus poços salvo no MongoDB.

2. Módulo de aplicação

Este módulo é o responsável por integrar as diversas tecnologias envolvidas para fornecer ao usuário final a facilidade de encontrar poços correlatos através da clusterização realizada no módulo 1.

Abaixo o diagrama de sequência do módulo de aplicação:

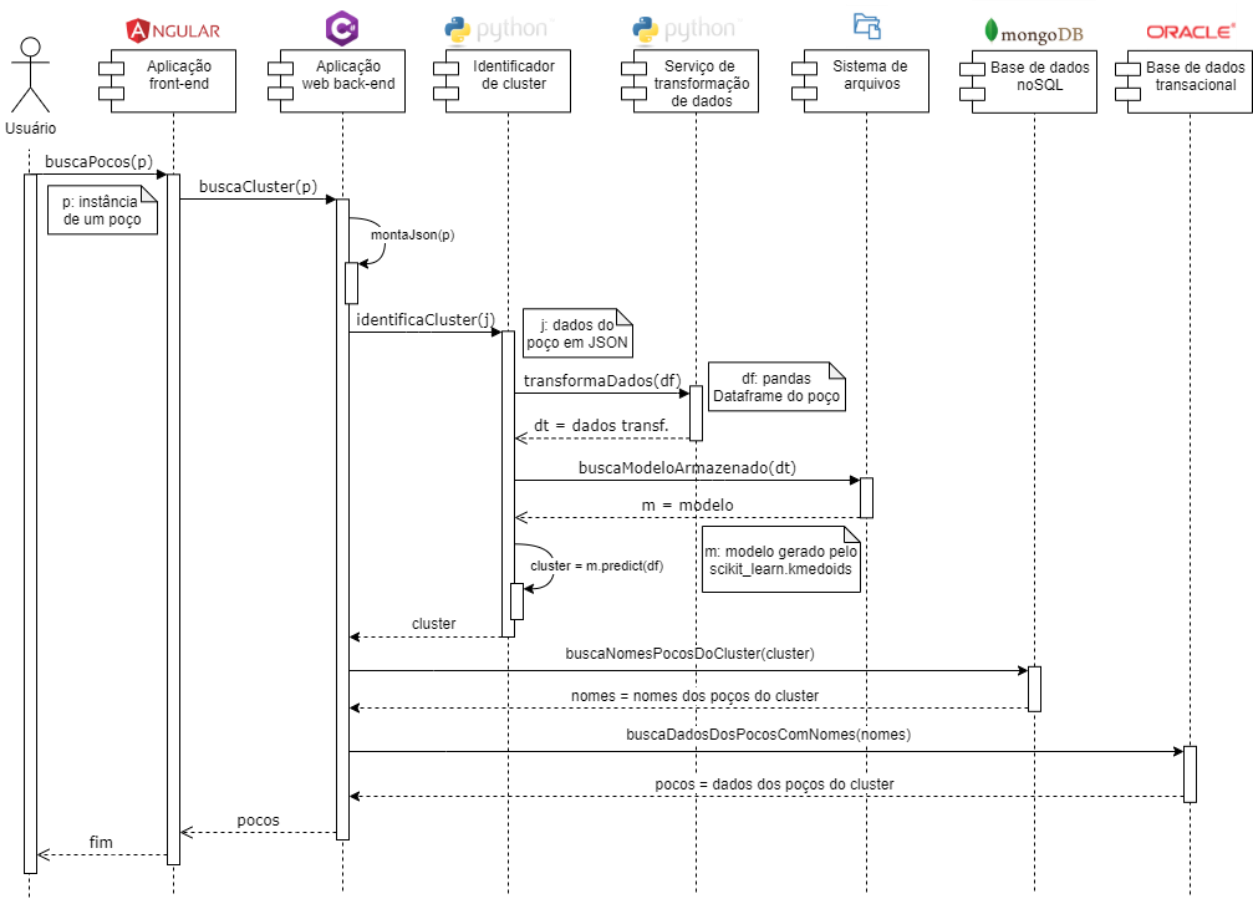


Figura 9 - Diagrama de sequência da aplicação de consulta a poços correlatos

Podemos dividir o módulo de aplicação em 3 submódulos:

2.1. Frontend em Angular

Esta parte do código implementado foi integrada ao que já existia no Sistema CW. Ou seja, na tela de definição de atividades de perfuração foi implementado um mecanismo para buscar poços recomendados. Este mecanismo foi alterado para que se buscasse os poços do mesmo cluster através do script Python que utiliza o modelo salvo em arquivo pelo módulo de clusterização.

```
127    buscaEscoposPorTipoServico(dataTablesParameters: any, codigoTipoServico: number, codigoPontoGeografico: number): Observable<any> {
128        let params = `?codigoTipoServico=${codigoTipoServico}&codigoPontoGeografico=${codigoPontoGeografico}`;
129        return this.http.post<any>(this.ROTAS["buscaEscoposPorTipoServico"] + params, dataTablesParameters, {});
130    }
```

Figura 10 - Método no serviço Angular que se comunica com o backed para buscar poços correlatos

2.2. Backend em C#

No backend a chamada inicial é feita ao controlador (EscopoController). Neste, o método "BuscaEscoposPorTipoServico" foi alterado para buscar os poços do mesmo cluster do poço em edição (representado pelo parâmetro "codigoPontoGeografico"), como pode ser visto na imagem abaixo:

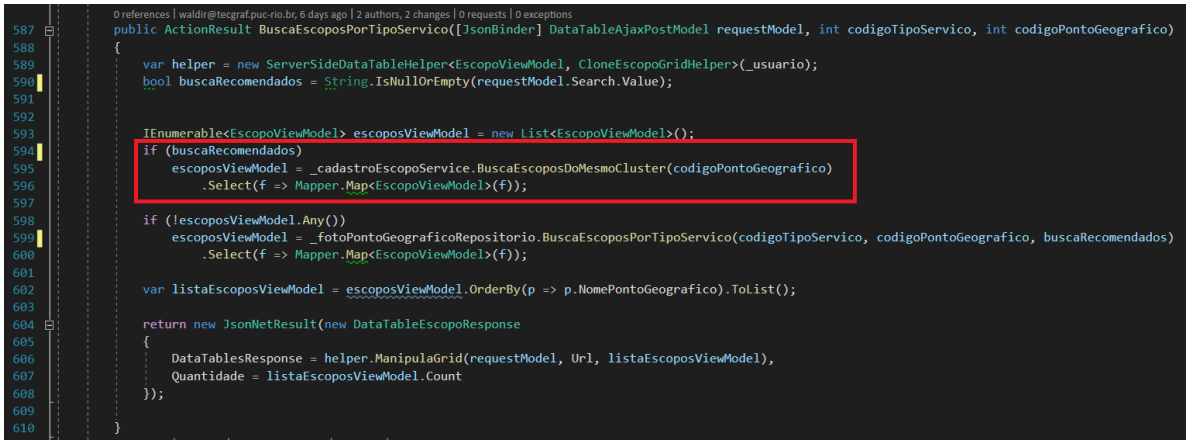


Figura 11 - Alteração no método BuscaEscoposPorTipoServico para buscar escopos do mesmo cluster

Foi criado, então, o método "BuscaEscoposDoMesmoCluster" no serviço do cadastro de escopo (CadastroEscopoService). Este método é o responsável por buscar o cluster do poço (no Sistema CW chamado de Ponto Geográfico), depois buscar os nomes dos poços registrados para aquele cluster, no MongoDB e, por fim, buscar os objetos referentes aos poços através de uma busca por seus nomes, no Oracle.

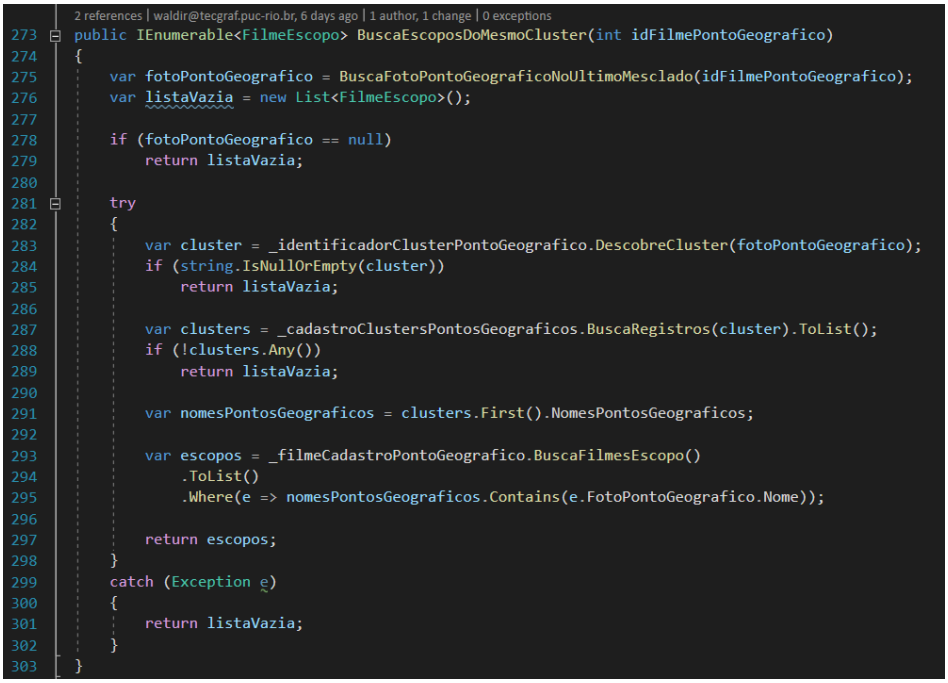


Figura 12 - Método BuscaEscoposDoMesmoCluster do CadastroEscopoService

Para realizar a identificação do cluster, este método utiliza outro serviço (IdentificadorClusterPontoGeografico), que faz a comunicação com o script Python através da biblioteca PyRunner [17]. Este serviço pode ser visto no Anexo II – IdentificadorClusterPontoGeografico.cs.

Os passos seguidos por este serviço são:

1. Cria um JSON através dos dados do poço passado por parâmetro
2. Executa o script em Python para descobrir o cluster do poço passado
3. Retorna o cluster encontrado

2.3. Script Python para identificação de cluster

Como última parte do módulo de aplicação, o script "cluster-identifier.py" é o responsável pelas seguintes etapas:

1. Receber o JSON com o poço a ter seu cluster calculado;
2. Converter este JSON em um Dataframe;
3. Aplicar as transformações necessárias utilizando o mesmo serviço de transformação utilizado no módulo de clusterização (TransformacaoDadosService);
4. Ler o modelo de clusterização salvo em arquivo;
5. Aplicar o Dataframe do poço a ser clusterizado no modelo lido do sistema de arquivos; e
6. Retornar o cluster identificado.

O código deste script pode ser lido no Anexo III – Script Python para identificação do cluster de um poço.

Testes

Os testes foram divididos em algumas partes:

1. Serviço de transformação de dados

Para o serviço de transformação de dados foram criados testes unitários (utilizando a biblioteca PyTest) e executados através da ferramenta Spyder.

Em cada método foi realizado um "Mock" dos dados necessários à sua execução, de forma a manter a independência entre os métodos.

Abaixo uma imagem mostrando a execução de todos os testes relacionados aos métodos do serviço de transformação de dados.

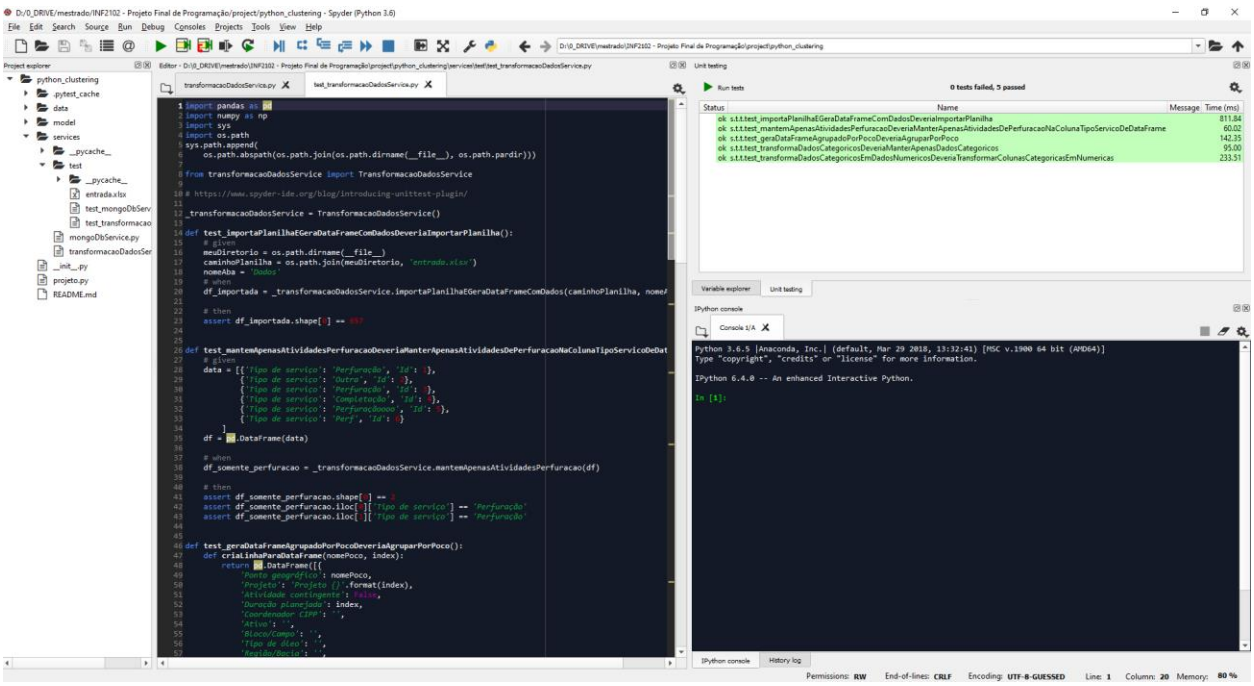


Figura 13 - Resultado dos testes unitários do serviço de transformação de dados

O código do arquivo de testes pode ser visto no Anexo IV – Testes do serviço de transformação de dados.

2. Integração com sistema de arquivos

Para a integração com o sistema de arquivos, o método de testes foi através do próprio módulo de clusterização, já que, para garantia do resultado, seria melhor utilizar o modelo gerado em memória para comparar os dados com o modelo lido do sistema de arquivos.

Como mostrado nos itens "1.8. Testando os pontos no modelo em memória e obtendo resultado correto" e "1.9. Testando os pontos no modelo buscado em arquivo e obtendo resultado correto" da seção Arquitetura da solução, os testes foram realizados com sucesso.

3. Integração entre backend C# e script Python

Para a integração entre backend C# e o script em Python foi escrito um teste de integração em C#. O código pode ser visto abaixo:

```
32  [TestMethod]
    | 0 references | waldir@tecgraf.puc-rio.br, 7 days ago | 1 author, 3 changes | 0 exceptions
33  public void DescobreClusterDeveriaDescobrirClusterDePontoGeografico()
34  {
35      // given
36      var fotoPontoGeografico = new FotoPontoGeografico();
37      fotoPontoGeografico.Nome = "POCO1";
38
39      // when
40      var result = _identificadorClusterPontoGeografico.DescobreCluster(fotoPontoGeografico);
41
42      // then
43      Assert.AreEqual(expected: "1", actual: result);
44  }
```

Figura 14 - Teste de integração entre backend C# e o script em Python

Estudo de caso

A fim de verificar que toda implementação foi bem-sucedida, realizei um teste pela própria aplicação, editando um novo poço e verificando se o sistema seria capaz de buscar poços correlatos através do módulo de aplicação, com base nos clusteres gerados pelo módulo de clusterização.

Segue abaixo sequência de imagens demonstrando o passo-a-passo do teste realizado:

1. Edição de um poço sem nenhuma atividade

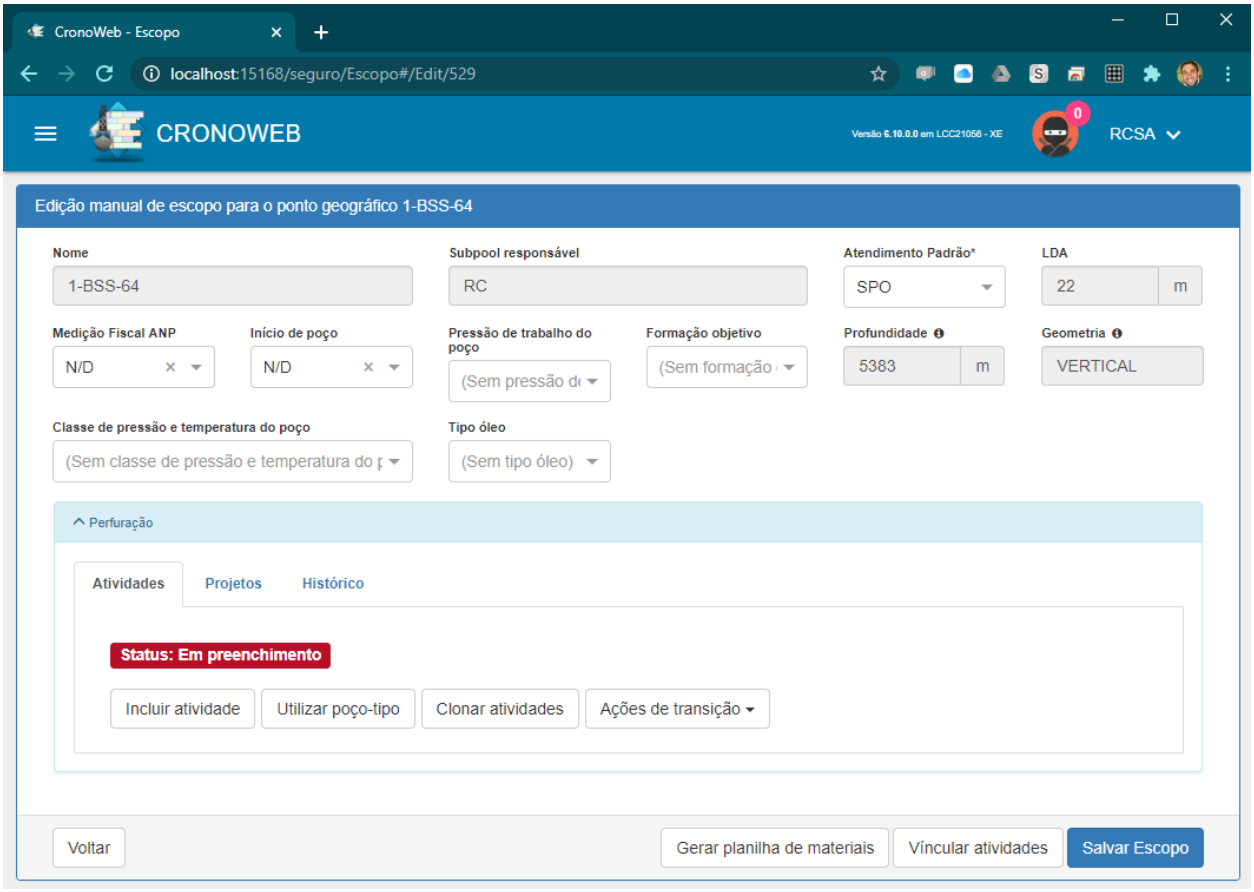


Figura 15 - Edição de um poço sem nenhuma atividade

2. Clique no botão “Clonar atividades”

Esta ação abre a janela que busca os poços correlatos através da clusterização.

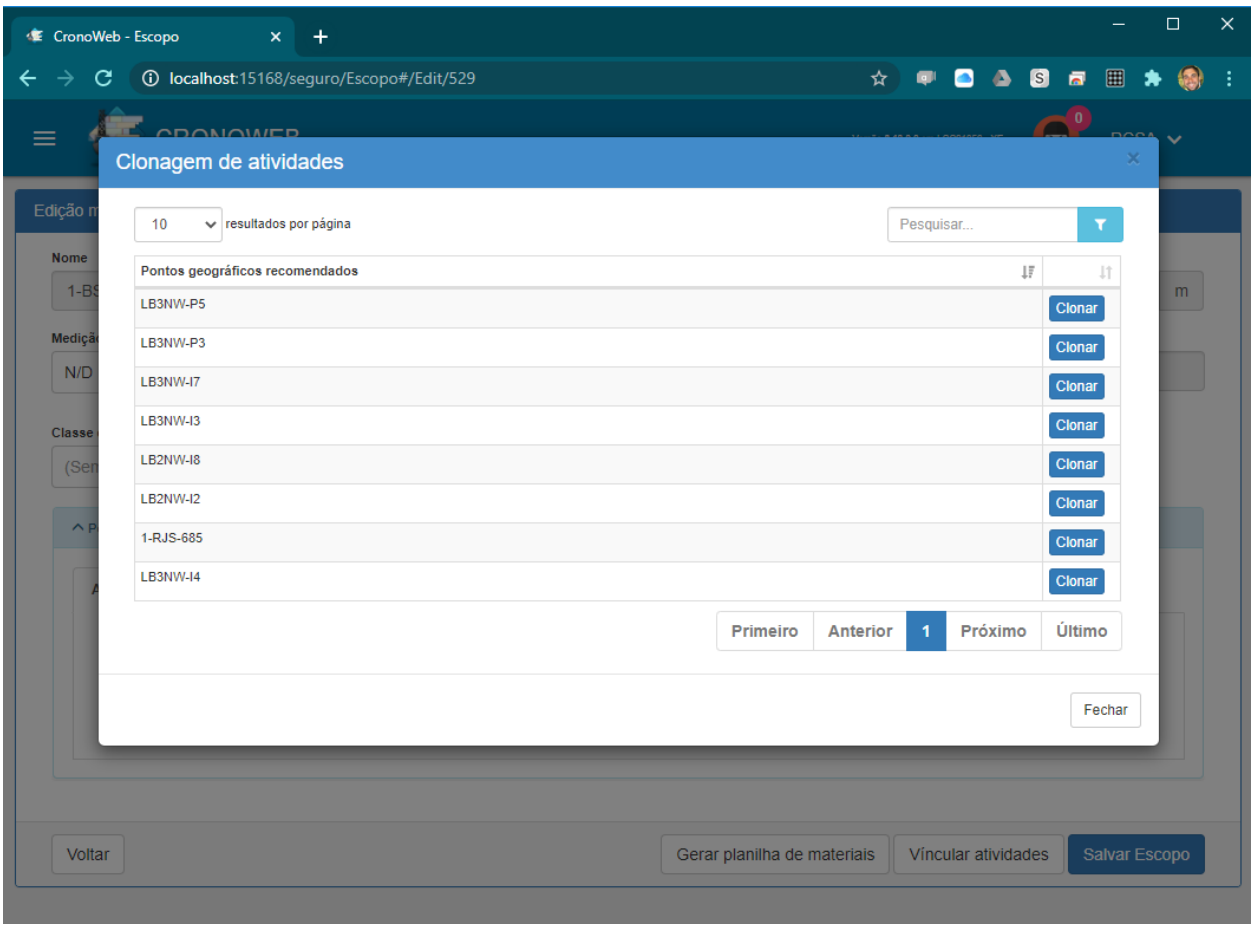


Figura 16 - Clique no botão “Clonar atividades”

Os poços que apareceram foram os mesmos do cluster “1”, já vistos na consulta ao MongoDB mostrada na Figura 5 - Clusters e poços gravados no MongoDB, conforme esperado.

3. Clique sobre a ação “Clonar” do poço 1-RJS-685

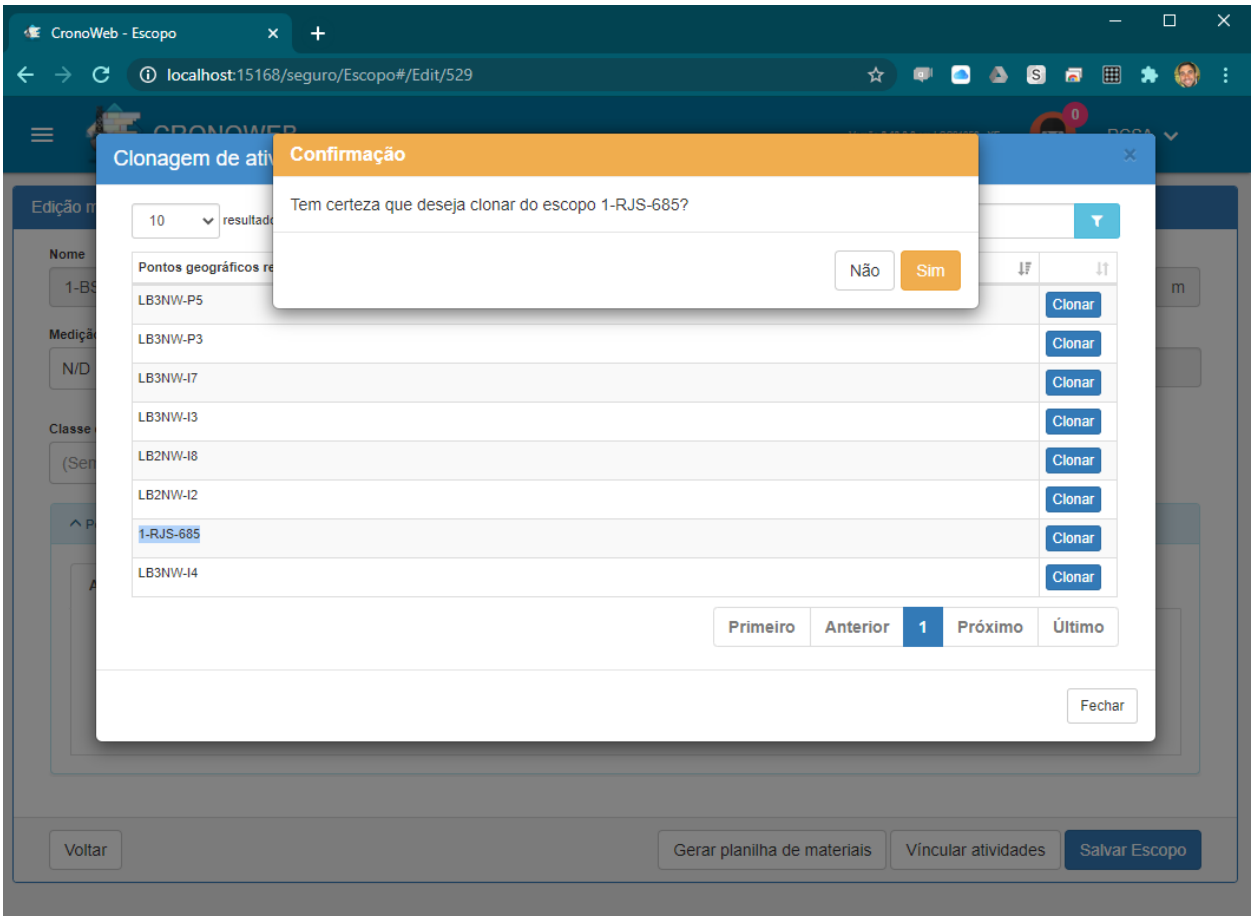


Figura 17 - Clique sobre a ação “Clonar” do poço 1-RJS-685

4. Atividades “clonadas” do poço correlato 1-RJS-685

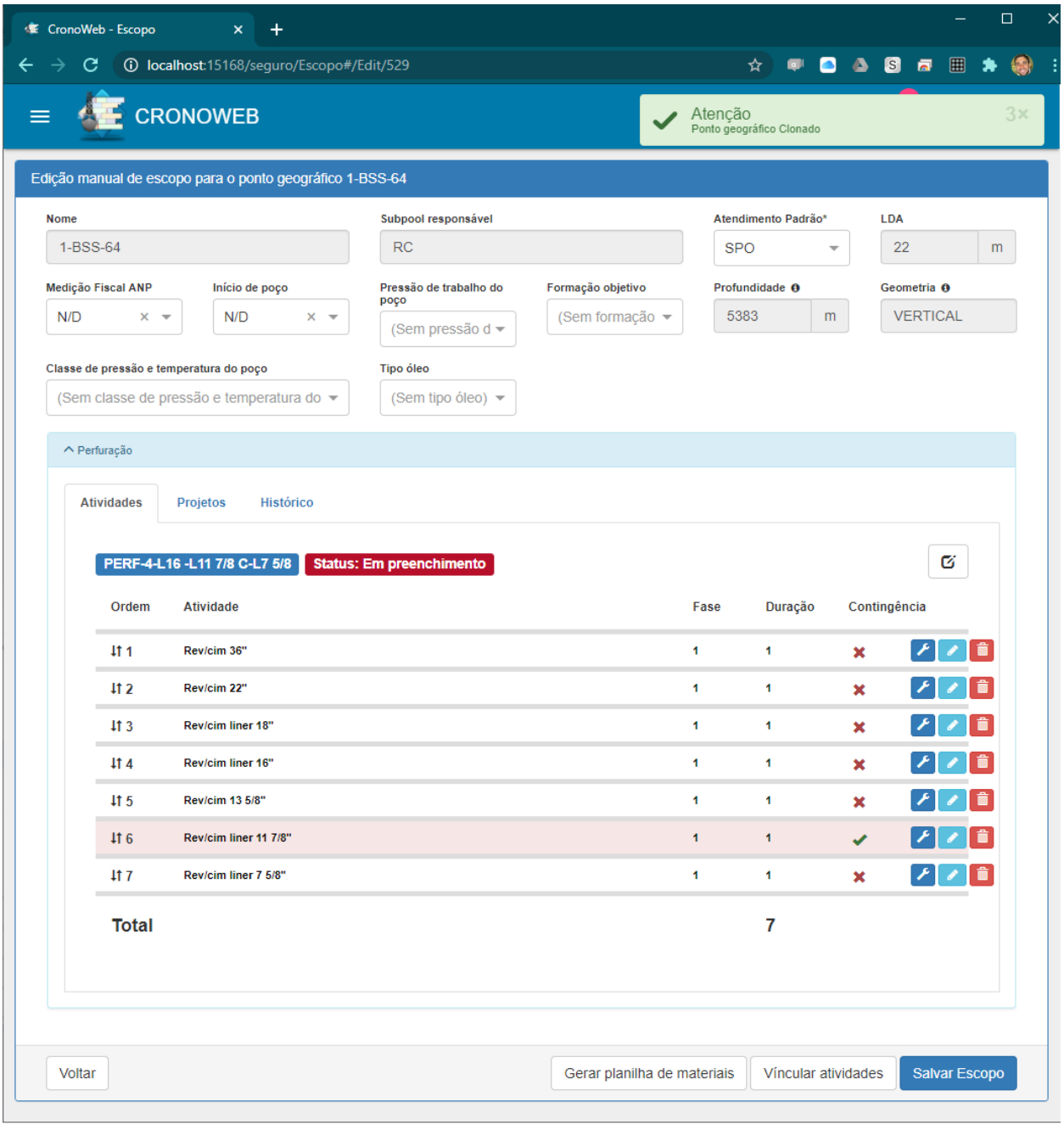


Figura 18 - Atividades “clonadas” do poço correlato 1-RJS-685

Conclusão

Este trabalho apresentou uma solução envolvendo campos distintos de pesquisa, tais como Ciência de Dados, Engenharia de Software e Banco de Dados. O objetivo foi mostrar que é possível integrar soluções de diferentes arquiteturas visando um objetivo final, que é auxiliar o projetista de poço a descobrir poços correlatos ao que está sendo planejado no Sistema CW.

Algumas oportunidades de melhorias podem ser listadas, tais como:

- Automatização da execução do módulo de clusterização diariamente, regerando todos os clusters a partir dos poços existentes no momento.
- Mudança do modelo de clusterização, para buscar dados geoglógicos além de dados de projeto de poço.
- Busca de dados de poços através de consultas a API REST ou diretamente no banco de dados Oracle, em vez de importar planilha Excel.
- Substituir alguns caminhos absolutos existentes no código por caminhos relativos ou configurados externamente.

Apesar da massa de dados ter sido pequena (apenas 34 poços), pude perceber o potencial da ferramenta. Estando em ambiente de produção e com um total de milhares de poços para clusterização, a ajuda que poderá fornecer aos usuários será de grande valia.

Referências

[1] "The Cost of Oil & Gas Wells," [Online]. Available: http://www.oilscams.org/how-much-does-oil-gas-well-cost. [Accessed 5 July 2020].

[2] D. C. M. Pearson, "THE SEVEN STEPS OF OIL AND NATURAL GAS EXTRACTION," [Online]. Available: https://www.cred.org/seven-steps-of-oil-and-natural-gas-extraction/. [Accessed 5 July 2020].

[3] "Welcome to Python.org," [Online]. Available: https://www.python.org/. [Accessed 1 June 2020].

[4] "C# documentation," [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/. [Accessed 1 June 2020].

[5] Google, "Angular," [Online]. Available: https://angular.io/. [Accessed 1 June 2020].

[6] "TypeScript: Typed JavaScript at Any Scale," [Online]. Available: https://www.typescriptlang.org/. [Accessed 1 June 2020].

[7] "Oracle | Integrated Cloud Applications and Platform Services," [Online]. Available: https://www.oracle.com/. [Accessed 1 June 2020].

[8] "MongoDB: The most popular database for modern apps," [Online]. Available: https://www.mongodb.com/. [Accessed 1 June 2020].

[9] "scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/stable/index.html. [Accessed 5 July 2020].

[10] "sklearn_extra.cluster.KMedoids," [Online]. Available: https://scikit-learn-extra.readthedocs.io/en/latest/generated/sklearn_extra.cluster.KMedoids.html. [Accessed 15 June 2020].

[11] "sklearn.cluster.KMeans," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html. [Accessed 15 June 2020].

[12] "What is Sequence Diagram?," [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/. [Accessed 1 July 2020].

[13] "Project Jupyter," [Online]. Available: https://jupyter.org/. [Accessed 5 July 2020].

[14] "pandas.DataFrame," [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html. [Accessed 15 June 2020].

[15] M. A. Syakur, B. K. Khotimah, E. M. S. Rochman and B. D. Satoto, "Integration K-Means Clustering Method and Elbow," *IOP Conference Series: Materials Science and Engineering,* 2018.

[16] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," 1987.

[17] T. Weller, "Using Python Scripts from a C# Client (Including Plots and Images)," 26 August 2019. [Online]. Available: https://www.codeproject.com/Articles/5165602/Using-Python-Scripts-from-a-Csharp-Client-Includin. [Accessed 5 July 2020].

[18] J. Niesen, "Introducing the unittest plugin," 23 February 2018. [Online]. Available: https://www.spyder-ide.org/blog/introducing-unittest-plugin/. [Accessed 27 June 2020].

[19] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications, Volume 36, Issue 2, Part 2,* pp. 3336-3341, March 2009.

Anexo I – Jupyter Notebook do módulo de clusterização

Trabalho Final de Programação - 2020.1

Programa de Mestrado em Informática - PUC-Rio

Professor Orientador: Marcelo Gattass

Professor Co-orientador: Helio Cortes Vieira Lopes

Aluno: Waldir José Pereira Junior

Matrícula: 1820998

In [1]:

```
import pandas as pd
import numpy as np
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

from services.transformacaoDadosService import TransformacaoDadosService
transformacaoDadosService = TransformacaoDadosService()
Inicializando o TransformacaoDadosService.
```

Leitura do cronograma em Excel

In [2]:

```
df = transformacaoDadosService.importaPlanilhaEGeraDataFrameComDados('data/Escopo_Integracao_DELFOS_Cronograma_
Pocos-SPO.xlsx', 'Dados do escopo (DELFOS)')
```

In [3]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 657 entries, 0 to 656
Data columns (total 58 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Prefixo do recurso                    657 non-null    object
 1   Nome do recurso                       175 non-null    object
 2   Código único                          20 non-null     object
 3   Projeto                               657 non-null    object
 4   Locações (Subprojeto)                 657 non-null    object
 5   Ponto geográfico                      657 non-null    object
 6   Tipo de serviço                       657 non-null    object
 7   Subtipo de serviço                    141 non-null    object
```

```
8   Tipo de atividade                          657 non-null    object
9   Nome da atividade                          657 non-null    object
10  Fase da atividade                          657 non-null    float64
11  Atividade contingente                      657 non-null    object
12  Duração planejada                          657 non-null    float64
13  Duração realizada                          0 non-null      float64
14  Início                                     657 non-null    datetime64[ns]
15  Término                                    657 non-null    datetime64[ns]
16  Coordenador CIPP                           477 non-null    object
17  Ativo                                      657 non-null    object
18  Bloco/Campo                                657 non-null    object
19  Tipo de óleo                               657 non-null    object
20  Poço POLEO                                 0 non-null      float64
21  Projeto de investimento                    657 non-null    object
22  Região/Bacia                               657 non-null    object
23  Natureza do projeto                        0 non-null      float64
24  Tipo de locação                            657 non-null    object
25  Projeto pré-sal                            0 non-null      float64
26  Cessão onerosa                             657 non-null    object
27  SICAR                                      657 non-null    object
28  Parcerias                                  657 non-null    object
29  Fase do projeto                            657 non-null    object
30  Rodada ANP                                 657 non-null    object
31  Pressão de trabalho do poço                406 non-null    object
32  Classe de pressão e temperatura do poço    657 non-null    object
33  Formação                                   400 non-null    object
34  LDA                                        657 non-null    float64
35  Início de poço                             0 non-null      float64
36  Poço-tipo                                  376 non-null    object
37  Subtipo de poço                            340 non-null    object
38  ID do projeto                              657 non-null    float64
39  ID da atividade                            657 non-null    object
40  Demandante                                 657 non-null    object
41  Coordenador do projeto no cliente          0 non-null      float64
42  Geometria do poço                          131 non-null    object
43  Diâmetro do packer                         0 non-null      float64
44  Tipo de completação inferior               258 non-null    object
45  Demanda CATS                               215 non-null    object
46  Necessidade de SCC                         248 non-null    object
47  ID Tarefa original                         657 non-null    float64
48  Porto de embarque                          0 non-null      float64
49  IUPI do projeto de investimento            657 non-null    object
50  Tipo de ponto geográfico                   505 non-null    object
51  Tipo de tarefa                             657 non-null    object
```

```
 52  Tarefa probabilística/determinística     657 non-null    object
 53  Prefixo do recurso na base integrada     175 non-null    object
 54  Status de contratação do recurso         657 non-null    object
 55  Tipo de recurso                          657 non-null    object
 56  Subtipo de recurso                       657 non-null    object
 57  Tipo posicionamento do recurso           657 non-null    object
dtypes: datetime64[ns](2), float64(13), object(43)
memory usage: 297.8+ KB
```

Tratamentos

Este arquivo é gerado por atividades/tarefas.

Primeiramente somente as atividades de perfuração serão mantidas (removendo atividades de completação, workover e avaliação)

In [4]:

```
df = transformacaoDadosService.mantemApenasAtividadesPerfuracao(df)
```

In [5]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 368 entries, 0 to 614
Data columns (total 58 columns):
 #   Column                                   Non-Null Count  Dtype
---  ------                                   --------------  -----
 0   Prefixo do recurso                       368 non-null    object
 1   Nome do recurso                          80 non-null     object
 2   Código único                             0 non-null      object
 3   Projeto                                  368 non-null    object
 4   Locações (Subprojeto)                    368 non-null    object
 5   Ponto geográfico                         368 non-null    object
 6   Tipo de serviço                          368 non-null    object
 7   Subtipo de serviço                       75 non-null     object
 8   Tipo de atividade                        368 non-null    object
 9   Nome da atividade                        368 non-null    object
 10  Fase da atividade                        368 non-null    float64
 11  Atividade contingente                    368 non-null    object
 12  Duração planejada                        368 non-null    float64
 13  Duração realizada                        0 non-null      float64
 14  Início                                   368 non-null    datetime64[ns]
 15  Término                                  368 non-null    datetime64[ns]
 16  Coordenador CIPP                         261 non-null    object
 17  Ativo                                    368 non-null    object
 18  Bloco/Campo                              368 non-null    object
```

```
 19  Tipo de óleo                                368 non-null    object
 20  Poço POLEO                                  0 non-null      float64
 21  Projeto de investimento                     368 non-null    object
 22  Região/Bacia                               368 non-null    object
 23  Natureza do projeto                         0 non-null      float64
 24  Tipo de locação                             368 non-null    object
 25  Projeto pré-sal                             0 non-null      float64
 26  Cessão onerosa                              368 non-null    object
 27  SICAR                                       368 non-null    object
 28  Parcerias                                   368 non-null    object
 29  Fase do projeto                             368 non-null    object
 30  Rodada ANP                                  368 non-null    object
 31  Pressão de trabalho do poço                 218 non-null    object
 32  Classe de pressão e temperatura do poço     368 non-null    object
 33  Formação                                    215 non-null    object
 34  LDA                                         368 non-null    float64
 35  Início de poço                              0 non-null      float64
 36  Poço-tipo                                   368 non-null    object
 37  Subtipo de poço                             332 non-null    object
 38  ID do projeto                               368 non-null    float64
 39  ID da atividade                             368 non-null    object
 40  Demandante                                  368 non-null    object
 41  Coordenador do projeto no cliente           0 non-null      float64
 42  Geometria do poço                           26 non-null     object
 43  Diâmetro do packer                          0 non-null      float64
 44  Tipo de completação inferior                88 non-null     object
 45  Demanda CATS                                89 non-null     object
 46  Necessidade de SCC                          114 non-null    object
 47  ID Tarefa original                          368 non-null    float64
 48  Porto de embarque                           0 non-null      float64
 49  IUPI do projeto de investimento             368 non-null    object
 50  Tipo de ponto geográfico                    269 non-null    object
 51  Tipo de tarefa                              368 non-null    object
 52  Tarefa probabilística/determinística        368 non-null    object
 53  Prefixo do recurso na base integrada        80 non-null     object
 54  Status de contratação do recurso            368 non-null    object
 55  Tipo de recurso                             368 non-null    object
 56  Subtipo de recurso                          368 non-null    object
 57  Tipo posicionamento do recurso              368 non-null    object
dtypes: datetime64[ns](2), float64(13), object(43)
memory usage: 169.6+ KB
```

In [6]:

```
df.head()
```

Out[6]:

	Prefixo do recurso	Nome do recurso	Código único	Projeto	Locações (Subprojeto)	Ponto geográfico	Tipo de serviço	Subtipo de serviço	Tipo de atividade	Nome da atividade	...	Porto de embarque	IUPI do projeto de investimento	Tipo de ponto geográfico	
0	NS-2400-LB-03	NaN	NaN	MERO 2	LB2NW.P2	LB2NW-P2	Perfuração	NaN	Movimentação	DMM	...	NaN	DP-14-0075	Poço atual / Objetivo Final	Pe
1	NS-2400-LB-03	NaN	NaN	MERO 2	LB2NW.P2	LB2NW-P2	Perfuração	NaN	Perfuração	PERFURACAO 28" COM ALARGADOR 42"	...	NaN	DP-14-0075	Poço atual / Objetivo Final	Pe
2	NS-2400-LB-03	NaN	NaN	MERO 2	LB2NW.P2	LB2NW-P2	Perfuração	NaN	Rev/cim	REV/CIM 36"	...	NaN	DP-14-0075	Poço atual / Objetivo Final	Pe
3	NS-2400-LB-03	NaN	NaN	MERO 2	LB2NW.P2	LB2NW-P2	Perfuração	NaN	Perfuração	PERFURACAO 28"	...	NaN	DP-14-0075	Poço atual / Objetivo Final	Pe
4	NS-2400-LB-03	NaN	NaN	MERO 2	LB2NW.P2	LB2NW-P2	Perfuração	NaN	Rev/cim	REV/CIM 22"	...	NaN	DP-14-0075	Poço atual / Objetivo Final	Pe

5 rows × 58 columns

Um novo dataframe agrupado por poço será gerado.

In [7]:

```
df_ajustado = transformacaoDadosService.geraDataFrameAgrupadoPorPoco(df)
```

In [8]:

```
df_ajustado.head()
```

Out[8]:

	Ponto geográfico	Projeto	Possui contingente	Duração	Coordenador CIPP	Ativo	Bloco/Campo	Tipo de óleo	Região/Bacia	Natureza do projeto	...	Cessão onerosa	SICAR	Rodada ANP	Formaçã possu Carbonat
0	1-RJS-685	MERO 1	True	58.0	CTNT	UN-RIO/ATP-MERO	MERO	N/D	SANTOS	NaN	...	False	SP0200A	PARTILHA - ROUND 1	Tru
1	3-ESS-142	MERO 1	False	88.0	CTNT	UN-RIO/ATP-MERO	MERO	N/D	SANTOS	NaN	...	False	SP0200A	PARTILHA - ROUND 1	Tru
2	3-ESS-144	MERO 1	False	81.0	CTNT	UN-RIO/ATP-MERO	MERO	N/D	SANTOS	NaN	...	False	SP0200A	PARTILHA - ROUND 1	Tru
3	3-ESS-155	MERO 1	False	81.0	CTNT	UN-RIO/ATP-MERO	MERO	N/D	SANTOS	NaN	...	False	SP0200A	PARTILHA - ROUND 1	Tru
4	3-RJS-360	MERO 1	False	201.0	CTNT	UN-RIO/ATP-MERO	MERO	N/D	SANTOS	NaN	...	False	SP0200A	PARTILHA - ROUND 1	Fals

5 rows × 22 columns

Preparação para clusterização

Transformação de dados categóricos

In [9]:

```
df_categorico = transformacaoDadosService.transformaDadosCategoricos(df_ajustado)
df_categorico.dtypes
```

Out[9]:

```
Projeto                          category
Possui contingente                   bool
Duração                           float64
Coordenador CIPP                 category
Ativo                            category
Bloco/Campo                      category
Tipo de óleo                     category
Região/Bacia                     category
Natureza do projeto              category
Tipo de locação                  category
Projeto pré-sal                      bool
Cessão onerosa                       bool
SICAR                            category
Rodada ANP                       category
Formação possui Carbonato            bool
Formação possui Arenito              bool
LDA Max                           float64
Demandante                       category
Tipo de completação inferior     category
Demanda CATS Firme                   bool
Necessita SCC                        bool
Poço Injetor                         bool
Poço Produtor                        bool
Poço Especial                        bool
dtype: object
```

Transformação de dados categóricos em dados numéricos

In [10]:

```
df_numerico = transformacaoDadosService.transformaDadosCategoricosEmDadosNumericos(df_categorico)

df_numerico.head()
```

Out[10]:

	Possui contingente	Duração	Projeto pré-sal	Cessão onerosa	Formação possui Carbonato	Formação possui Arenito	LDA Max	Demanda CATS Firme	Necessita SCC	Poço Injetor	...	Tipo de locação_Poço Produtor de óleo	SICAR_SP0200A	SICAR_SP0201A	
0	1	58.0	0	0	1	0	2400.0	0	0	0	...	0	1	0	
1	0	88.0	0	0	1	0	2400.0	0	0	0	...	0	1	0	
2	0	81.0	0	0	1	0	2400.0	0	0	0	...	0	1	0	
3	0	81.0	0	0	1	0	2400.0	0	0	0	...	0	1	0	
4	0	201.0	0	0	0	0	0.0	0	0	0	...	1	1	0	

5 rows × 35 columns

To give equal importance to all features, we need to scale the continuous features. We will be using scikit-learn's MinMaxScaler as the feature matrix is a mix of binary and continuous features . Other alternatives includes StandardScaler.

In [11]:

```
#https://blog.cambridgespark.com/how-to-determine-the-optimal-number-of-clusters-for-k-means-clustering-14f27070048f
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt

def transforma(x):
    mms = MinMaxScaler()
    mms.fit(x)
    return mms.transform(x)

#df_transformado = transforma(df_numerico)
df_transformado = df_numerico.values
df_transformado
```

Out[11]:

```
array([[  1.,  58.,   0., ...,   0.,   0.,   1.],
       [  0.,  88.,   0., ...,   0.,   0.,   0.],
       [  0.,  81.,   0., ...,   0.,   0.,   1.],
       ...,
       [  0.,  83.,   0., ...,   0.,   0.,   0.],
       [  0., 114.,   0., ...,   0.,   0.,   0.],
       [  0., 118.,   0., ...,   0.,   0.,   0.]])
```

Calculando o K ideal (utilizando K-Means)

In [12]:

```
#https://stackoverflow.com/a/59962760/4784342
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
range_n_clusters = range(2,15)
elbow = []
ss = []
```

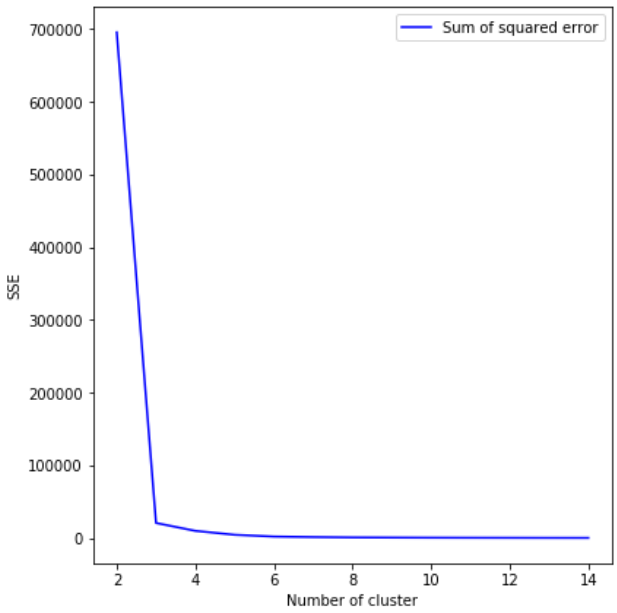
```
for n_clusters in range_n_clusters:
    #iterating through cluster sizes
    clusterer = KMeans(n_clusters = n_clusters, random_state=42)
    cluster_labels = clusterer.fit_predict(df_transformado)
    silhouette_avg = silhouette_score(df_transformado, cluster_labels)
    ss.append(silhouette_avg)
    elbow.append(clusterer.inertia_)
```

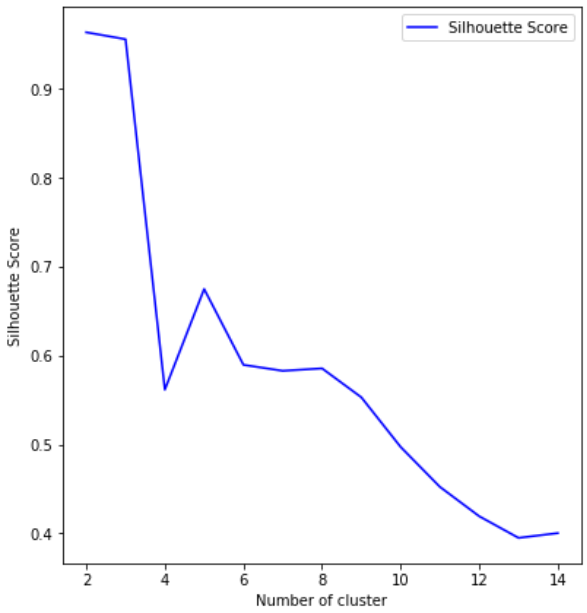
As k increases, the sum of squared distance tends to zero. Imagine we set k to its maximum value n (where n is number of samples) each sample will form its own cluster meaning sum of squared distances equals zero.

Below is a plot of sum of squared distances for k in the range specified above. If the plot looks like an arm, then the elbow on the arm is optimal k.

In [13]:

```
#https://stackoverflow.com/a/59962760/4784342
fig = plt.figure(figsize=(14,7))
fig.add_subplot(121)
plt.plot(range_n_clusters, elbow,'b-',label='Sum of squared error')
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
plt.legend()
fig.add_subplot(122)
plt.plot(range_n_clusters, ss,'b-',label='Silhouette Score')
plt.xlabel("Number of cluster")
plt.ylabel("Silhouette Score")
plt.legend()
plt.show()
```





Calculando os medoids

Inicialização

Escolhendo aleatoriamente k medoids

In [14]:

```
df_size = len(df_numerico)


np.random.seed(123)
initial_medoids = np.random.randint(df_size, size=3).tolist()


initial_medoids
```

Out[14]:

```
[2, 28, 17]
```

Calculando clusters através do K-Medoids

In [15]:

```
x = df_numerico.iloc[:].values
```

In [16]:

```
from sklearn_extra.cluster import KMedoids
from pyclustering.utils import timedcall, distance_metric, type_metric
metric = distance_metric(type_metric.EUCLIDEAN_SQUARE, data=x)
kmedoids = KMedoids(n_clusters=3, random_state=0,metric=metric).fit(x)
labels = kmedoids.labels_


clusters_2 = []
for i in range(kmedoids.n_clusters):
    clusters_2.append([])
i = 0
for label in labels:
    clusters_2[label].append(i)
    i = i + 1
clusters_2
```

Out[16]:

```
[[1, 2, 3, 5, 7, 8, 9, 11, 14, 15, 16, 17, 18, 24, 25, 26, 27, 28, 29, 30, 31],
 [0, 10, 13, 19, 20, 21, 22, 23],
 [4, 6, 12, 32, 33]]
```

Salvando dados dos clusters no MongoDB

In [17]:

```
from services.mongoDbService import MongoDbService
from projeto import configuracoes

cluster_id = 0
clusters_json = []
for cluster_indexes in clusters_2:
    listaNomesPGs = (df_ajustado.take(cluster_indexes)['Ponto geográfico']).tolist()
    clusters_json.append({'Cluster': cluster_id, 'NomesPontosGeograficos': listaNomesPGs})
    cluster_id = cluster_id + 1

mongoDbService = MongoDbService(configuracoes)
documentos_inseridos = mongoDbService.grava(clusters_json)
print("documentos_inseridos: ", documentos_inseridos)
```

```
documentos_inseridos:  <pymongo.results.InsertManyResult object at 0x000001D4EC4C0D88>
```

Imprimindo os clusters gerados

In [18]:

```
i = 0
for cluster_indexes in clusters_2:
    print('Cluster ', i)
    print(df_ajustado.take(cluster_indexes)['Ponto geográfico'].tolist())
    print('------')
    i = i + 1
```

```
Cluster  0
['3-ESS-142', '3-ESS-144', '3-ESS-155', '8-MRO-8-RJS', 'LB1NW-I4', 'LB1NW-I9', 'LB1NW-P8', 'LB2NW-I5', 'LB2NW-P
1', 'LB2NW-P2', 'LB2NW-P3', 'LB2NW-P6', 'LB2NW-P8', 'LB3NW-P6', 'MERO 3.LB3NW.I5', 'MERO 4.LB4NW.I2', 'MERO 4.L
B4NW.I3', 'MERO 4.LB4NW.I4', 'MERO 4.LB4NW.I5', 'MERO 4.LB4NW.I8', 'MERO 4.LB4NW.P3']
------
Cluster  1
['1-RJS-685', 'LB2NW-I2', 'LB2NW-I8', 'LB3NW-I3', 'LB3NW-I4', 'LB3NW-I7', 'LB3NW-P3', 'LB3NW-P5']
------
Cluster  2
['3-RJS-360', '9-MRO-9D-RJS', 'LB2NW-I6', 'MERO 4.LB4NW.P5', 'MERO 4.LB4NW.P8']
------
```

Salvando o modelo em arquivo

In [19]:

```
#https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/
import pickle

# save the model to disk
filename = 'finalized_model.sav'
filehandler = open(filename, 'wb')
pickle.dump(kmedoids, filehandler)
filehandler.close()
```

Buscando um ponto de cada cluster, para testar o modelo

In [20]:

```
df_numerico[1:2]
```

Out[20]:

	Possui contingente	Duração	Projeto pré-sal	Cessão onerosa	Formação possui Carbonato	Formação possui Arenito	LDA Max	Demanda CATS Firme	Necessita SCC	Poço Injetor	...	Tipo de locação_Poço Produtor de óleo	SICAR_SP0200A	SICAR_SP0201A	S
1	0	88.0	0	0	1	0	2400.0	0	0	0	...	0	1	0	

1 rows × 35 columns

In [21]:

```
#pegando um registro qualquer para verificar se a predição está correta
df_de_registro_teste_cluster_0 = pd.DataFrame(df_numerico[1:2])
df_de_registro_teste_cluster_1 = pd.DataFrame(df_numerico[4:5])
df_de_registro_teste_cluster_2 = pd.DataFrame(df_numerico[0:1])
df_de_registro_teste_transformado_cluster_0 = df_de_registro_teste_cluster_0.values
df_de_registro_teste_transformado_cluster_1 = df_de_registro_teste_cluster_1.values
df_de_registro_teste_transformado_cluster_2 = df_de_registro_teste_cluster_2.values
print(df_de_registro_teste_transformado_cluster_0)
print(df_de_registro_teste_transformado_cluster_1)
print(df_de_registro_teste_transformado_cluster_2)
[[0.0e+00 8.8e+01 0.0e+00 0.0e+00 1.0e+00 0.0e+00 2.4e+03 0.0e+00 0.0e+00
  0.0e+00 0.0e+00 0.0e+00 0.0e+00 1.0e+00 0.0e+00 0.0e+00 0.0e+00 1.0e+00
  1.0e+00 1.0e+00 1.0e+00 1.0e+00 0.0e+00 0.0e+00 1.0e+00 0.0e+00 1.0e+00
  0.0e+00 0.0e+00 0.0e+00 1.0e+00 1.0e+00 0.0e+00 0.0e+00 0.0e+00]]
[[  0. 201.   0.   0.   0.   0.   0.   0.   0.   0.   0.   0.   1.
    0.   0.   0.   1.   1.   1.   1.   1.   0.   0.   0.   1.   1.   0.
    0.   0.   1.   1.   0.   0.   0.]]
[[1.0e+00 5.8e+01 0.0e+00 0.0e+00 1.0e+00 0.0e+00 2.4e+03 0.0e+00 0.0e+00
  0.0e+00 0.0e+00 0.0e+00 0.0e+00 1.0e+00 0.0e+00 0.0e+00 0.0e+00 1.0e+00
  1.0e+00 1.0e+00 1.0e+00 1.0e+00 0.0e+00 0.0e+00 1.0e+00 0.0e+00 1.0e+00
  0.0e+00 0.0e+00 0.0e+00 1.0e+00 1.0e+00 0.0e+00 0.0e+00 1.0e+00]]
```

Testando os pontos no modelo em memória e obtendo resultado correto

In [24]:

```
result_cluster_0 = kmedoids.predict(df_de_registro_teste_transformado_cluster_0)
result_cluster_1 = kmedoids.predict(df_de_registro_teste_transformado_cluster_1)
result_cluster_2 = kmedoids.predict(df_de_registro_teste_transformado_cluster_2)
print("cluster do result_cluster_0: ", result_cluster_0)
print("cluster do result_cluster_1: ", result_cluster_1)
print("cluster do result_cluster_2: ", result_cluster_2)
```

```
cluster do result_cluster_0:  [0]
cluster do result_cluster_1:  [2]
cluster do result_cluster_2:  [1]
```

Testando os pontos no modelo buscado em arquivo e obtendo resultado correto

In [26]:

```
filename = 'finalized_model.sav'
filehandler = open(filename, 'rb')
loaded_model = pickle.load(filehandler)
filehandler.close()
result_saved_model_cluster_0 = loaded_model.predict(df_de_registro_teste_transformado_cluster_0)
result_saved_model_cluster_1 = loaded_model.predict(df_de_registro_teste_transformado_cluster_1)
result_saved_model_cluster_2 = loaded_model.predict(df_de_registro_teste_transformado_cluster_2)
print("cluster do result_saved_model_cluster_0 (que deve ser o mesmo de result_cluster_0): ", result_saved_mode
l_cluster_0)
print("cluster do result_saved_model_cluster_1 (que deve ser o mesmo de result_cluster_1): ", result_saved_mode
l_cluster_1)
print("cluster do result_saved_model_cluster_2 (que deve ser o mesmo de result_cluster_2): ", result_saved_mode
l_cluster_2)
```

```
cluster do result_saved_model_cluster_0 (que deve ser o mesmo de result_cluster_0):  [0]
cluster do result_saved_model_cluster_1 (que deve ser o mesmo de result_cluster_1):  [2]
cluster do result_saved_model_cluster_2 (que deve ser o mesmo de result_cluster_2):  [1]
```

Anexo II – IdentificadorClusterPontoGeografico.cs

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Diagnostics;
using System.Dynamic;
using System.IO;
using System.Linq;
using System.Net.Http;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using System.Web.Script.Serialization;
using CronogramaDeSondas.Dominio.Modelos;
using CronogramaDeSondas.Dominio.Modelos.Cronograma.Filme;
using CronogramaDeSondas.Dominio.Modelos.Cronograma.Foto;
using CronogramaDeSondas.Dominio.Modelos.Sonda;
using CronogramaDeSondas.Infra.ExportacaoPlanilha.Cronograma;
using MongoDB.Bson;
using Newtonsoft.Json;
using PyRunner;

namespace CronogramaDeSondas.Infra.Python
{
    public class IdentificadorClusterPontoGeografico : IIdentificadorClusterPontoGeografico
    {
        private PythonRunner PythonRunner { get; }

        private class PropriedadeParaArgumento<TP>
        {
            public Func<TP, object> PropriedadeValor { get; set; }
            public string NomeColuna { get; set; }
        }

        public IdentificadorClusterPontoGeografico()
        {
            //var pythonRunner = new PythonRunner(ConfigurationManager.AppSettings["pythonPath"], 20000);
            PythonRunner = new PythonRunner("D:\\ProgramData\\Anaconda3\\python.exe", 20000);
        }

        private string PegaCaminhoScript()
        {
            //var codeBase = Assembly.GetAssembly(typeof(IdentificadorClusterPontoGeografico)).CodeBase;
            var location = Assembly.GetExecutingAssembly().Location;
            var estaRodandoTeste = location.Contains("TestResults");
            var caminho = estaRodandoTeste ? location :
Assembly.GetAssembly(typeof(IdentificadorClusterPontoGeografico)).CodeBase;
            return Path.Combine(
                Path.GetDirectoryName(caminho) ?? throw new InvalidOperationException(),
                estaRodandoTeste ? string.Empty : "Python\\Scripts\\");
        }
```

```
public string DescobreCluster(FotoPontoGeografico fotoPontoGeografico)
{
    var caminhoScript = PegaCaminhoScript().Replace("file:\\", "");

    var tipoServico = new TipoServico
    {
        Codigo = TipoServico.CodigoTipoServicoPerfuracao,
        Nome = "Perfuração"
    };

    var fotoTarefa = new FotoTarefa
    {
        FotoPontoGeografico = fotoPontoGeografico,
        TipoTarefa = new TipoTarefa { Nome = "TT" },
        FotoServico = new FotoServico
        {
            Nome = "SRV",
            TipoServico = tipoServico,
            FotoLocacao = new FotoLocacao
            {
                Nome = "LOC",
                CodigoLocacaoFilme = 1,
                WibrNovosPadsNovosBids = 1,
                Bloco = new Bloco
                {
                    Nome = "BLOCO",
                    Ativo = new Ativo
                    {
                        Sigla = "ATIVO"
                    },
                    Rodada = new Rodada
                    {
                        Id = Rodada.CodigoCessaoOnerosa
                    }
                },
                FotoProjeto = new FotoProjeto
                {
                    Nome = "PRJ",
                    UnidadeOperativa = new UnidadeOperativa { Nome = "UO" },
                    ChaveUsuarioCaCipp = "CIPP",
                    ProjetoNovosPadsNovosBids = true,
                    GrupoEditorResponsavelPeloProjeto = new GrupoEditor
                    {
                        Codigo = GrupoEditor.CodigoGrupoAgp,
                        Nome = "AGP"
                    }
                }
            }
        }
    };

    var fotoRecurso = new FotoSonda
    {
```

```
            Nome = "SONDA",
            Prefixo = "SND",
            StatusContratacao = StatusContratacao.Ativo,
            PosicionamentoSonda = PosicionamentoSonda.Ancorado,
            TipoRecurso = TipoRecurso.Sonda
        };

        var fotoAlocacaoTarefa = new FotoAlocacaoTarefa
        {
            FotoTarefa = fotoTarefa,
            FotoRecurso = fotoRecurso,
            FotoCronograma = new FotoCronograma
            {
                DataDaFoto = DateTime.Today
            }
        };

        var escopoAtividade = new FilmeEscopoAtividade
        {
            Atividade = new Atividade
            {
                Codigo = "A999999",
                Nome = "Atividade",
                TipoAtividade = new TipoAtividade
                {
                    Codigo = TipoAtividade.CodigoTipoAtividadePerfuracao,
                    Nome = "TA",
                    TipoServico = tipoServico
                }
            },
            FilmeEscopo = new FilmeEscopo(),
            Fase = 1,
            Duracao = 1,
            EhContingencial = false,
            DataInicioCalculada = DateTime.Now
        };

        var tupla = new
TuplaEscopoAtividadeFotoAlocacaoTarefaParaPlanilhaEscopoIntegracaoDelfos(fotoAlocacaoTarefa,
            new CodigoUnico { Codigo = "CODIGO_UNICO" }, escopoAtividade);

        var dic = new Dictionary<string, object>();
        dic.Add("Prefixo do recurso", tupla.PrefixoRecurso);
        dic.Add("Nome do recurso", tupla.NomeRecurso);
        dic.Add("Código único", tupla.CodigoUnico);
        dic.Add("Projeto", tupla.NomeProjeto);
        //dic.Add("UO do projeto", tupla.SiglaUoProjeto);
        //dic.Add("ID da Locação", tupla.CodigoLocacaoFilme);
        dic.Add("Locações (Subprojeto)", tupla.NomeSubProjeto);
        //dic.Add("ID do Poço", tupla.CodigoPoco);
        dic.Add("Ponto geográfico", tupla.NomePontoGeografico);
        dic.Add("Tipo de serviço", tupla.TipoServico);
        dic.Add("Subtipo de serviço", tupla.SubtipoServico);
```

```
dic.Add("Tipo de atividade", tupla.TipoAtividade);
dic.Add("Nome da atividade", tupla.NomeAtividade);
dic.Add("Fase da atividade", tupla.FaseAtividade);
dic.Add("Atividade contingente", tupla.AtividadeContingente);
dic.Add("Duração planejada", tupla.Duracao);
dic.Add("Duração realizada", tupla.DuracaoRealizada);
dic.Add("Início", tupla.Inicio);
dic.Add("Término", tupla.Termino);
dic.Add("Coordenador CIPP", tupla.CoordenadorCipp);
dic.Add("Ativo", tupla.Ativo);
dic.Add("Bloco/Campo", tupla.BlocoCampo);
dic.Add("Tipo de óleo", tupla.TipoOleo);
dic.Add("Poço POLEO", tupla.PocoPoleo);
dic.Add("Projeto de investimento", tupla.ProjetoInvestimento);
dic.Add("Região/Bacia", tupla.Bacia);
dic.Add("Natureza do projeto", tupla.NomeNaturezaProjeto);
dic.Add("Tipo de locação", tupla.TipoLocacao);
dic.Add("Projeto pré-sal", tupla.ProjetoPreSal);
dic.Add("Cessão onerosa", tupla.CessaoOnerosa);
dic.Add("SICAR", tupla.Sicar);
dic.Add("Parcerias", tupla.Parcerias);
dic.Add("Fase do projeto", tupla.FaseProjeto);
dic.Add("Rodada ANP", tupla.RodadaAnp);
dic.Add("Pressão de trabalho do poço", tupla.PressaoTrabalhoPoco);
dic.Add("Classe de pressão e temperatura do poço", tupla.ClassePressaoTemperatura);
dic.Add("Formação", tupla.Formacao);
dic.Add("LDA", tupla.Lda);
dic.Add("Início de poço", tupla.InicioPoco);
dic.Add("Poço-tipo", tupla.PocoTipo);
dic.Add("Subtipo de poço", tupla.SubtipoPoco);
dic.Add("ID do projeto", tupla.IdProjeto);
dic.Add("ID da atividade", tupla.IdAtividade);
dic.Add("Demandante", tupla.Demandante);
dic.Add("Coordenador do projeto no cliente", tupla.CoordenadorProjetoCliente);
dic.Add("Geometria do poço", tupla.GeometriaPoco);
dic.Add("Diâmetro do packer", tupla.DiametroPacker);
dic.Add("Tipo de completação inferior", tupla.TipoCompletacaoInferior);
dic.Add("Demanda CATS", tupla.DemandaCats);
dic.Add("Necessidade de SCC", tupla.NecessidadeScc);
dic.Add("ID Tarefa original", tupla.CodigoDaTarefaFilmeOriginal);
dic.Add("Porto de embarque", tupla.PortoEmbarque);
dic.Add("IUPI do projeto de investimento", tupla.IupiProjetoInvestimento);
dic.Add("Tipo de ponto geográfico", tupla.TipoPontoGeografico);
dic.Add("Tipo de tarefa", tupla.TipoTarefa);
dic.Add("Tarefa probabilística/determinística", tupla.IndicadorProbabilisticoDeterministico);
dic.Add("Prefixo do recurso na base integrada", tupla.PrefixoBaseIntegrada);
dic.Add("Status de contratação do recurso", tupla.StatusContratacaoRecurso);
dic.Add("Tipo de recurso", tupla.TipoRecurso);
dic.Add("Subtipo de recurso", tupla.SubtipoRecurso);
dic.Add("Tipo posicionamento do recurso", tupla.TipoPosicionamentoRecurso);

string json = JsonConvert.SerializeObject(dic);
json = json.Replace("\"", "\\\"");
```

```
            var output = PythonRunner.Execute($"{caminhoScript}\\identificador-cluster.py", json,
caminhoScript);
            return output;
        }

        public async Task<string> Soma(int a, int b)
        {
            var output = await PythonRunner.ExecuteAsync($"{PegaCaminhoScript()}\\identificador-cluster.py", a,
b);
            return output;
        }
    }
}
```

Anexo III – Script Python para identificação do cluster de um poço

```
import os
import sys, warnings
import pandas as pd
import numpy as np
import json

# Suppress all kinds of warnings (this would lead to an exception on the client side).
warnings.simplefilter("ignore")

from transformacaoDadosService import TransformacaoDadosService
transformacaoDadosService = TransformacaoDadosService()

jsonData = sys.argv[1]
path = sys.argv[2]

data = json.loads(jsonData)
df = pd.json_normalize(data)

df_ajustado = transformacaoDadosService.geraDataFrameAgrupadoPorPoco(df)
df_categorico = transformacaoDadosService.transformaDadosCategoricos(df_ajustado)
df_numerico = transformacaoDadosService.transformaDadosCategoricosEmDadosNumericos(df_categorico)

import pickle

filename = '{}/finalized_model.sav'.format(path)
filehandler = open(filename, 'rb')
loaded model = pickle.load(filehandler)
filehandler.close()

valores = df_numerico.values

tamanho_inicial = valores.shape[1]
tamanho_objetivo = len(loaded_model.labels_)
i = 0
for i in range(tamanho_objetivo - tamanho_inicial + 1):
    valores = np.append(valores, 0)

df_to_fit = [valores]

result_saved_model_cluster = loaded_model.predict(df_to_fit)

print(result_saved_model_cluster[0])
```

Anexo IV – Testes do serviço de transformação de dados

```
import pandas as pd
import numpy as np
import sys
import os.path
sys.path.append(
    os.path.abspath(os.path.join(os.path.dirname(__file__), os.path.pardir)))

from transformacaoDadosService import TransformacaoDadosService

# https://www.spyder-ide.org/blog/introducing-unittest-plugin/

 transformacaoDadosService = TransformacaoDadosService()

def test_importaPlanilhaEGeraDataFrameComDadosDeveriaImportarPlanilha():
    # given
    meuDiretorio = os.path.dirname(__file__)
    caminhoPlanilha = os.path.join(meuDiretorio, 'entrada.xlsx')
    nomeAba = 'Dados'
    # when
    df_importada = _transformacaoDadosService.importaPlanilhaEGeraDataFrameComDados(caminhoPlanilha, nomeAba)

    # then
    assert df_importada.shape[0] == 657


def
test_mantemApenasAtividadesPerfuracaoDeveriaManterApenasAtividadesDePerfuracaoNaColunaTipoServicoDeDataFrame():
    # given
    data = [{'Tipo de serviço': 'Perfuração', 'Id': 1},
            {'Tipo de serviço': 'Outra', 'Id': 2},
            {'Tipo de serviço': 'Perfuração', 'Id': 3},
            {'Tipo de serviço': 'Completação', 'Id': 4},
            {'Tipo de serviço': 'Perfuraçãoooo', 'Id': 5},
            {'Tipo de serviço': 'Perf', 'Id': 6}
        ]
    df = pd.DataFrame(data)

    # when
    df_somente_perfuracao = _transformacaoDadosService.mantemApenasAtividadesPerfuracao(df)

    # then
    assert df_somente_perfuracao.shape[0] == 2
    assert df_somente_perfuracao.iloc[0]['Tipo de serviço'] == 'Perfuração'
    assert df_somente_perfuracao.iloc[1]['Tipo de serviço'] == 'Perfuração'


def test_geraDataFrameAgrupadoPorPocoDeveriaAgruparPorPoco():
    def criaLinhaParaDataFrame(nomePoco, index):
        return pd.DataFrame([{
            'Ponto geográfico': nomePoco,
            'Projeto': 'Projeto {}'.format(index),
```

```
            'Atividade contingente': False,
            'Duração planejada': index,
            'Coordenador CIPP': '',
            'Ativo': '',
            'Bloco/Campo': '',
            'Tipo de óleo': '',
            'Região/Bacia': '',
            'Natureza do projeto': '',
            'Tipo de locação': '',
            'Projeto pré-sal': '',
            'Cessão onerosa': '',
            'SICAR': '',
            'Rodada ANP': '',
            'Formação': 'CARBONATO',
            'LDA': '',
            'Demandante': '',
            'Tipo de completação inferior': '',
            'Demanda CATS': 'Não aplicável',
            'Necessidade de SCC': 'Não'
        }])

    # given
    df = pd.DataFrame()

    df = df.append(criaLinhaParaDataFrame('PG1', 1))
    df = df.append(criaLinhaParaDataFrame('PG2', 2))
    df = df.append(criaLinhaParaDataFrame('PG3', 3))
    df = df.append(criaLinhaParaDataFrame('PG4', 4))
    df = df.append(criaLinhaParaDataFrame('PG1', 5))
    df = df.append(criaLinhaParaDataFrame('PG2', 6))
    df = df.append(criaLinhaParaDataFrame('PG2', 7))
    df = df.append(criaLinhaParaDataFrame('PG3', 8))

    # when
    df_agrupado= _transformacaoDadosService.geraDataFrameAgrupadoPorPoco(df)

    # then
    assert df_agrupado.shape[0] == 4
    assert df_agrupado.iloc[0]['Ponto geográfico'] == 'PG1'
    assert df_agrupado.iloc[1]['Ponto geográfico'] == 'PG2'
    assert df_agrupado.iloc[2]['Ponto geográfico'] == 'PG3'
    assert df_agrupado.iloc[3]['Ponto geográfico'] == 'PG4'


def test_transformaDadosCategoricosDeveriaManterApenasDadosCategoricos():
    def criaLinhaParaDataFrame(nomePoco, index):
        return pd.DataFrame([{
            'Ponto geográfico': nomePoco,
            'Projeto': 'Projeto {}'.format(index),
            'Atividade contingente': False,
            'Duração planejada': index,
            'Coordenador CIPP': 'CIPP {}'.format(index),
            'Ativo': 'Ativo {}'.format(index),
```

```
        'Bloco/Campo': 'Bloco {}'.format(index),
        'Tipo de óleo': 'TipoOleo {}'.format(index),
        'Região/Bacia': 'Bacia {}'.format(index),
        'Natureza do projeto': 'Natureza {}'.format(index),
        'Tipo de locação': 'TipoLocacao {}'.format(index),
        'Projeto pré-sal': 'Sim',
        'Cessão onerosa': 'Não',
        'SICAR': 'SICAR {}'.format(index),
        'Rodada ANP': 'Rodada {}'.format(index),
        'Formação': 'Formacao {}'.format(index),
        'LDA': 1000,
        'Demandante': 'Demandante {}'.format(index),
        'Tipo de completação inferior': 'TipoCompletacaoInferior {}'.format(index),
        'Demanda CATS': 'CATS {}'.format(index),
        'Necessidade de SCC': 'Não'
    }])

# given
df = pd.DataFrame()

df = df.append(criaLinhaParaDataFrame('PG1', 1))
df = df.append(criaLinhaParaDataFrame('PG2', 2))
df = df.append(criaLinhaParaDataFrame('PG3', 3))
df = df.append(criaLinhaParaDataFrame('PG4', 4))
df = df.append(criaLinhaParaDataFrame('PG1', 5))
df = df.append(criaLinhaParaDataFrame('PG2', 6))
df = df.append(criaLinhaParaDataFrame('PG2', 7))
df = df.append(criaLinhaParaDataFrame('PG3', 8))

# then
# checando que as colunas atualmente sao nao-categoricas
assert df.select_dtypes(include='category').shape[1] == 0
assert df['Projeto'].dtype.name != 'category'
assert df['Coordenador CIPP'].dtype.name != 'category'
assert df['Ativo'].dtype.name != 'category'
assert df['Bloco/Campo'].dtype.name != 'category'
assert df['Tipo de óleo'].dtype.name != 'category'
assert df['Região/Bacia'].dtype.name != 'category'
assert df['Natureza do projeto'].dtype.name != 'category'
assert df['Tipo de locação'].dtype.name != 'category'
assert df['SICAR'].dtype.name != 'category'
assert df['Rodada ANP'].dtype.name != 'category'
assert df['Demandante'].dtype.name != 'category'
assert df['Tipo de completação inferior'].dtype.name != 'category'

# when
df_categorico = _transformacaoDadosService.transformaDadosCategoricos(df)

# then
# checando que as colunas atualmente sao categoricas
assert df_categorico.select_dtypes(include='category').shape[1] > 0
assert df['Projeto'].dtype.name == 'category'
assert df['Coordenador CIPP'].dtype.name == 'category'
```

```
    assert df['Ativo'].dtype.name == 'category'
    assert df['Bloco/Campo'].dtype.name == 'category'
    assert df['Tipo de óleo'].dtype.name == 'category'
    assert df['Região/Bacia'].dtype.name == 'category'
    assert df['Natureza do projeto'].dtype.name == 'category'
    assert df['Tipo de locação'].dtype.name == 'category'
    assert df['SICAR'].dtype.name == 'category'
    assert df['Rodada ANP'].dtype.name == 'category'
    assert df['Demandante'].dtype.name == 'category'
    assert df['Tipo de completação inferior'].dtype.name == 'category'

def test_transformaDadosCategoricosEmDadosNumericosDeveriaTransformarColunasCategoricasEmNumericas():
    # given
    df = pd.DataFrame([
            {'COL_CAT': 'teste1', 'COL_NUM': 1, 'COL_BOOL': True},
            {'COL_CAT': 'teste2', 'COL_NUM': 2, 'COL_BOOL': True},
            {'COL_CAT': 'teste3', 'COL_NUM': 3, 'COL_BOOL': True},
            {'COL_CAT': 'teste4', 'COL_NUM': 4, 'COL_BOOL': True}
        ])

    df['COL_CAT'] = df['COL_CAT'].astype('category')

    # then
    assert 'COL_CAT' in df
    assert df['COL_CAT'].dtype.name == 'category'

    # when
    df_numerico = _transformacaoDadosService.transformaDadosCategoricosEmDadosNumericos(df)

    # then
    # checando que a coluna COL CAT nao existe mais
    assert 'COL_CAT' not in df_numerico

    #checando que as colunas categoricas foram criadas e sao numericas
    assert np.issubdtype(df_numerico['COL_CAT_teste1'].dtype, np.number)
    assert np.issubdtype(df_numerico['COL_CAT_teste2'].dtype, np.number)
    assert np.issubdtype(df_numerico['COL_CAT_teste3'].dtype, np.number)
    assert np.issubdtype(df_numerico['COL_CAT_teste4'].dtype, np.number)
```