Objetivos:

- Compreender os principais conceitos de operações espaciais em SIG;
- Realizar operações espaciais utilizando a API do PyQGIS;
- Automatizar análises espaciais com scripts em Python no ambiente QGIS;
- Exportar para arquivo GeoPackage.

I. Introdução às Operações Espaciais

As operações espaciais são procedimentos fundamentais no processamento e análise de dados geográficos. Elas permitem explorar relações geométricas entre feições, como sobreposição, proximidade, interseção, união, diferença e cálculo de distância.

No contexto do QGIS, a API PyQGIS oferece uma interface poderosa para realizar essas operações de forma automatizada, permitindo o desenvolvimento de scripts complexos e a integração com fluxos de trabalho em SIG.

Principais operações espaciais:

- **Interseção:** obtém a parte comum entre duas ou mais geometrias;
- **União:** combina geometrias, resultando em uma nova feição composta;
- **Diferença:** extrai a parte de uma geometria que não se sobrepõe a outra;
- **Buffer (zona de influência):** cria uma área a uma distância fixa ao redor de uma geometria;
- **Distância:** calcula a distância mínima entre geometrias;
- **Contém, intersecta, dentro:** operações relacionais para verificar as relações espaciais entre feições.

II. Conceitos Fundamentais no PyQGIS

O módulo `qgis.core` disponibiliza classes e métodos para manipulação de geometrias e execução de operações espaciais.

Principais classes e métodos:

- `QgsGeometry`: representa a geometria de uma feição e possui métodos para realizar operações espaciais;
- `QgsFeature`: representa uma feição de camada, contendo atributos e geometria;
- `QgsVectorLayer`: representa uma camada vetorial;
- `processing.run()`: executa algoritmos da Caixa de Ferramentas de Processamento do QGIS.

Exemplo de métodos da classe `QgsGeometry`**:**

- intersects(outra_geom)
- intersection(outra_geom)
- union(outra_geom)
- difference(outra_geom)
- buffer(distância, segmentos)

III. Exemplo 1: Interseção de camadas vetoriais

Objetivo: Identificar áreas de interseção entre duas camadas: uma de polígonos representando as Bacias Hidrográficas do Brasil e outra com polígonos dos estados.

1. Carregar as camadas

```
# Carrega as camadas vetoriais a partir dos arquivos GeoPackage
bacia = QgsVectorLayer('D:/pasta/bacia.gpkg', 'bacia', 'ogr')
uf = QgsVectorLayer('D:/pasta/uf.gpkg', 'uf', 'ogr')

# Adiciona as camadas carregadas ao projeto QGIS atual
QgsProject.instance().addMapLayer(bacia)
QgsProject.instance().addMapLayer(uf)
```

2. Criar camada de saída para interseção

```
# Cria uma camada em memória, para receber o resultado da interseção
intersecao = QgsVectorLayer('Polygon?crs=EPSG:4326', 'interseção', 'memory')

# Obtém o provedor de dados da camada de interseção para manipulação
provedor_intersecao = intersecao.dataProvider()

# Define os campos (atributos) da camada de interseção
provedor_intersecao.addAttributes([
        QgsField('uf', QVariant.String),
        QgsField('bacia', QVariant.String)
    ])

# Atualiza a definição dos campos na camada de interseção
intersecao.updateFields()
```

3. Processar interseções

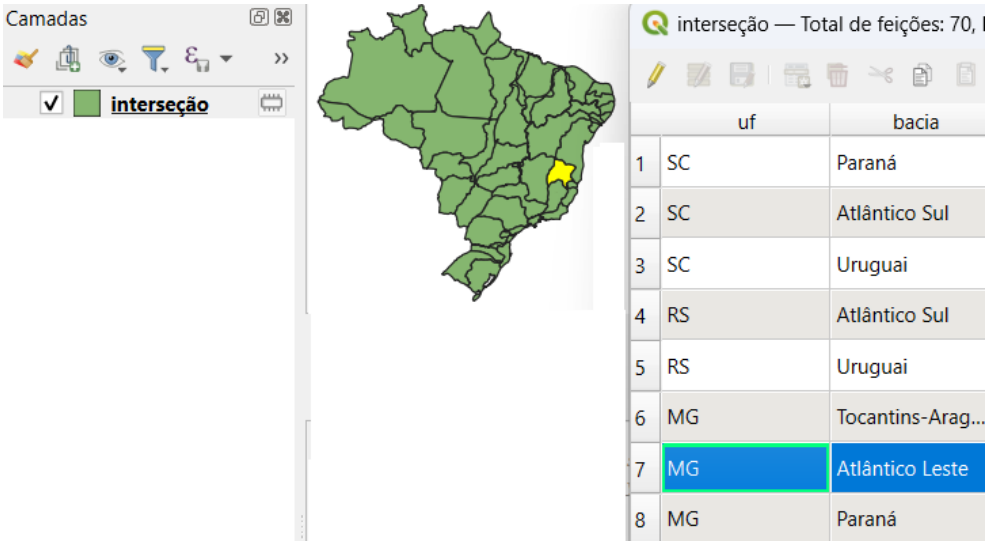
```
# Itera sobre cada feição da camada de estados
for feicao_uf in uf.getFeatures():
    # Obtém a geometria da feição de estados
    geom_uf = feicao_uf.geometry()
    # Obtém a sigla do estado
    nome_uf = feicao_uf['uf']

    # Itera sobre cada feição da camada de bacia hidrográfica
    for feicao_bacia in bacia.getFeatures():
        # Obtém a geometria da feição de bacia
        geom_bacia = feicao_bacia.geometry()
        # Obtém o nome da bacia
        nome_bacia = feicao_bacia['nome']

        # Verifica se as geometrias de UF e da bacia se intersectam
```

```
        if geom_uf.intersects(geom_bacia):
            # Calcula a geometria resultante da interseção
            inter_geom = geom_uf.intersection(geom_bacia)

            # Cria uma nova feição com os campos da camada de interseção
            nova_feicao = QgsFeature(intersecao.fields())
            # Define a geometria da nova feição como sendo a interseção calculada
            nova_feicao.setGeometry(inter_geom)
            # Atribui os valores de UF e bacia aos respectivos campos
            nova_feicao.setAttribute('uf', nome_uf)
            nova_feicao.setAttribute('bacia', nome_bacia)

            # Adiciona a nova feição ao provedor de dados da camada de interseção
            provedor_intersecao.addFeature(nova_feicao)
```

4. Adicionar camada ao projeto

```
# Adiciona a camada de interseção ao projeto QGIS para visualização
QgsProject.instance().addMapLayer(intersecao)
```

Resultado do código:



IV. Exemplo 2: Criando buffer em feições

Objetivo: Gerar uma zona de proteção de 5000 metros ao redor dos pontos de pedágio da Dutra.

1. Carregar a camada de pontos

```
# Carrega a camada de pontos dos pedágios
pontos = QgsVectorLayer('D:/pasta/pedagio.gpkg', 'pedágios', 'ogr')
QgsProject.instance().addMapLayer(pontos)
```

2. Criar camada de saída para buffer

```
# Cria uma camada de polígonos em memória para armazenar os buffers
buffer_camada = QgsVectorLayer('Polygon?crs=EPSG:4326', 'buffer', 'memory')

# Obtém o provedor de dados da camada de buffer
provedor_buffer = buffer_camada.dataProvider()

# Adiciona os mesmos atributos da camada original de pontos à camada de buffer
provedor_buffer.addAttributes(pontos.fields())
buffer_camada.updateFields()
```

3. Gerar buffers

```
# Itera sobre cada feição da camada de pontos
for feicao in pontos.getFeatures():
    # Obtém a geometria do ponto
    geom = feicao.geometry()

    # Cria um buffer de aproximadamente 5500 metros e 20 segmentos para suavização
    buffer_geom = geom.buffer(0.05, 20)

    # Cria uma nova feição com os mesmos campos da camada de buffer
    buffer_feicao = QgsFeature(buffer_camada.fields())

    # Define a geometria da nova feição como sendo o buffer
    buffer_feicao.setGeometry(buffer_geom)

    # Copia os atributos da feição original
    buffer_feicao.setAttributes(feicao.attributes())

    # Adiciona a nova feição ao provedor de dados da camada de buffer
    provedor_buffer.addFeatures([buffer_feicao])
```

4. Adicionar camada ao projeto

```
# Adiciona a camada de buffer ao projeto QGIS
QgsProject.instance().addMapLayer(buffer_camada)
```

Resultado do código:

V. Exemplo 3: Obtendo os vizinhos de uma feição

Objetivo: Realizar a operação espacial "toque" (touch) utilizando a camada vetorial de polígonos das Unidades da Federação (UFs). A operação de toque identifica feições que compartilham fronteiras, mas não se sobrepõem.

Conceito: "Touch"

Na álgebra espacial, duas geometrias tocam quando:

- Suas fronteiras possuem pelo menos um ponto em comum;
- Mas seus interiores não se sobrepõem.

Fluxo da operação

1. Carregar a camada de UFs;
2. Criar uma camada em memória para armazenar as relações de toque;
3. Iterar sobre todas as UFs e verificar quais tocam entre si;
4. Para cada par que toca, criar uma linha ligando os centroides dos polígonos que tocam;
5. Armazenar informações dos estados que tocam na camada de saída.

1. Carregar a camada de polígonos

```
# Carregar a camada de polígonos dos estados
camada = QgsVectorLayer('D:/pasta/uf.gpkg', 'estados', 'ogr')

# Adicionar a camada ao projeto
QgsProject.instance().addMapLayer(camada)
```

2. Criar camada de saída para armazenas as conexões de toque

```
# Criar uma camada de linhas em memória para armazenar as conexões de toque
touch_camada = QgsVectorLayer('LineString?crs=EPSG:4326', 'toques', 'memory')

# Obter o provedor de dados da camada
provedor = touch_camada.dataProvider()

# Definir os atributos da camada de saída
provedor.addAttributes([
    QgsField('uf1', QVariant.String),
    QgsField('uf2', QVariant.String)
])
touch_camada.updateFields()
```

3. Gerar as linhas que conectam os centroides dos polígonos vizinhos

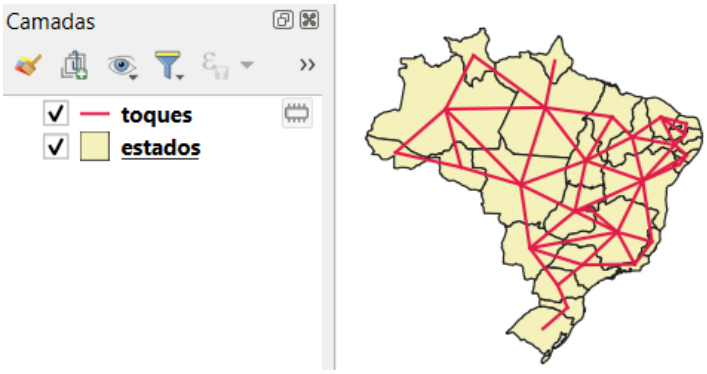
```
# Iterar sobre todas as combinações de estados
feicoes = list(camada.getFeatures())
```

```
for i, feicao1 in enumerate(feicoes):
    geom1 = feicao1.geometry()
    nome1 = feicao1['uf']

    for j in range(i + 1, len(feicoes)):
        feicao2 = feicoes[j]
        geom2 = feicao2.geometry()
        nome2 = feicao2['uf']

        # Verifica se as geometrias tocam
        if geom1.touches(geom2):
            # Criar uma linha entre os centróides dos dois estados
            centroide1 = geom1.centroid().asPoint()
            centroide2 = geom2.centroid().asPoint()
            linha = QgsGeometry.fromPolylineXY([
                QgsPointXY(centroide1),
                QgsPointXY(centroide2)
            ])

            # Criar uma nova feição com a linha
            nova_feicao = QgsFeature(touch_camada.fields())
            nova_feicao.setGeometry(linha)
            nova_feicao.setAttributes([nome1, nome2])

            # Adicionar a feição ao provedor
            provedor.addFeatures([nova_feicao])
```

4. Adicionar camada ao projeto

```
# Adicionar a camada de toques ao projeto
QgsProject.instance().addMapLayer(touch_camada)
```

Resultado do código:



VI. Exemplo 4: Unindo geometrias

Objetivo: Realizar a operação espacial de união (union), também chamada de dissolve ou merge, com o propósito de fundir todos os polígonos das Unidades da Federação (UFs) em uma única geometria composta.

Essa operação é comumente utilizada para:

- Simplificar camadas vetoriais;
- Criar uma representação do território nacional;
- Preparar dados para análises que envolvem limites externos e não divisões internas.

Conceito: União de geometrias

A operação de união consiste em combinar várias geometrias em uma única, eliminando as bordas internas e formando um polígono composto.

1. Carregar a camada de polígonos

```
# Carregar a camada de polígonos dos estados
camada = QgsVectorLayer('D:/pasta/uf.gpkg', 'estados', 'ogr')
```

2. Carregar em uma lista todas as geometrias da camada de entrada

```
# Lista para armazenar todas as geometrias
lista_geometrias = []

# Iterar sobre todas as feições para coletar as geometrias
for feicao in camada.getFeatures():
    geom = feicao.geometry()
    # Adiciona apenas se a geometria for válida
    if geom.isGeosValid():
        lista_geometrias.append(geom)
```

3. Unir as geometrias

```
# Realiza a união (dissolve) de todas as geometrias da lista
# unaryUnion retorna uma única geometria resultante
geom_uniao = QgsGeometry.unaryUnion(lista_geometrias)
```

4. Criar camada de saída para armazenar a feição resultado da união

```
# Criar uma camada em memória para armazenar a união
uniao_camada = QgsVectorLayer('Polygon?crs=EPSG:4326', 'uniao', 'memory')

# Obter o provedor de dados da camada
provedor = uniao_camada.dataProvider()

# Criar um campo para identificação
provedor.addAttributes([QgsField('pais', QVariant.String)])
uniao_camada.updateFields()
```

5. Colocar a feição com a união na camada de saída

```
# Criar a feição da união
uniao_feicao = QgsFeature(uniao_camada.fields())
uniao_feicao.setGeometry(geom_uniao)
uniao_feicao.setAttribute('pais', 'Brasil')

# Adicionar a feição ao provedor
provedor.addFeatures([uniao_feicao])
```

6. Adicionar camada ao projeto

```
# Adicionar a camada de união ao projeto
QgsProject.instance().addMapLayer(uniao_camada)
```

Resultado do código:

VII. Exemplo 5: Exportar camada no formato Geopackage

Objetivo: Salvar a camada do exemplo anterior no formato de arquivo GeoPackage

Para salvar a camada resultante da união das geometrias no formato GeoPackage (GPKG), é necessário utilizar a funcionalidade de exportação de camadas disponível no QGIS através do módulo `QgsVectorFileWriter` (https://qgis.org/pyqgis/3.40/core/QgsVectorFileWriter.html).

O processo de exportação deve contemplar as seguintes etapas:

1. Definir o caminho e o nome do arquivo de saída

O caminho indica a pasta onde o arquivo será salvo e o nome define como será identificado o arquivo no sistema. No caso, o arquivo deverá ser salvo como `pais.gpkg` no diretório `D:/pasta/`.

```
arquivo_saida = 'D:/pasta/pais.gpkg'
```

2. Configurar as opções de exportação

É necessário especificar:

- O **driver** de exportação, que neste caso é "GPKG" para o formato **GeoPackage**;
- O **nome da camada** dentro do arquivo, que pode ser diferente do nome do arquivo. Neste exemplo, será `"pais"`.

```
# Configurações para exportação
options = QgsVectorFileWriter.SaveVectorOptions()
options.driverName = "GPKG"
options.layerName = "pais"  # Nome da camada dentro do GeoPackage
```

3. Executar a exportação da camada

A função `QgsVectorFileWriter.writeAsVectorFormatV2()` permite realizar a exportação com segurança, além de retornar informações sobre o sucesso ou erro do processo. Os parâmetros mais importantes são:

- A camada que será exportada (`uniao_camada`);
- O caminho do arquivo de saída;
- O contexto de transformação de coordenadas do projeto `QgsProject.instance().transformContext()`;
- As opções configuradas no passo anterior.

```
# Executa a exportação da camada de união para o arquivo GeoPackage
QgsVectorFileWriter.writeAsVectorFormatV2(
    uniao_camada,
    arquivo_saida,
    QgsProject.instance().transformContext(),
    options
)
```

Exercícios

Veja os vídeos se tiver dúvidas nos exercícios:

Exercício 1: https://youtu.be/iO9wN0Y3cQY

Exercício 2: https://youtu.be/6-Q7xyVmy6E

Exercício 3: https://youtu.be/XWf60vkaglk

Exercício 4: https://youtu.be/WiGvGRmqGDI

Exercício 5: https://youtu.be/YWDOUyGK5XI

Exercício 1: Fazer um programa para listar as siglas dos estados que interceptam a Bacia Hidrográfica do Paraguai.

Entrada:

- Camada vetorial de bacias hidrográficas (bacia.gpkg).
- Camada vetorial de unidades federativas (uf.gpkg).

Saída:

- Lista dos nomes dos estados que fazem interseção espacial com a bacia cujo atributo NOME seja igual a "Paraguai".

Passos principais:

1. Carregar a camada vetorial de bacias hidrográficas a partir do arquivo bacia.gpkg;

```
bacia = QgsVectorLayer('D:/pasta/bacia.gpkg', 'bacia', 'ogr')
```

2. Carregar a camada vetorial das unidades federativas a partir do arquivo uf.gpkg;

```
uf = QgsVectorLayer('D:/pasta/uf.gpkg', 'uf', 'ogr')
```

3. Inicializar uma variável `bacia_paraguai` como vazia;

```
bacia_paraguai = None
```

4. Para cada feição na camada de bacias:

```
for feicao_bacia in bacia.getFeatures():
```

   - Verificar se o valor do atributo NOME é igual a "Paraguai";

```
if feicao_bacia['NOME'] == 'Paraguai':
```

   - Se verdadeiro:
     - Atribuir a feição encontrada à variável `bacia_paraguai`;

```
bacia_paraguai = feicao_bacia
```

     - Encerrar a busca (otimização).

```
break  # Para otimizar, encerramos a busca após encontrar
```

5. Se `bacia_paraguai` for nula:
   - Exibir a mensagem: `"Bacia com NOME = 'Paraguai' não encontrada"`
6. Caso contrário (se a bacia for encontrada):
   - Obter a geometria da `bacia_paraguai` e armazenar na variável `geom_bacia`.

```
geom_bacia = bacia_paraguai.geometry()
```

7. Inicializar uma lista vazia para armazenar os nomes dos estados.

```
estados_intersectantes = []
```

8. Para cada feição na camada das unidades federativas:
   - Obter a geometria da feição (`geom_uf`);
   - Verificar se `geom_uf` intersecta a geometria `geom_bacia`;
   - Se verdadeiro:
     - Obter o valor do atributo uf na camada de estados da federação;
     - Adicionar esse valor à lista `estados_intersectantes`.
9. Exibir a mensagem: `"Estados que fazem interseção com a Bacia do Paraguai:"`.
10. Para cada estado na lista `estados_intersectantes`:
    - Exibir o nome do estado.

Resposta esperada:

```
Estados que fazem interseção com a Bacia do Paraguai:
GO
MT
MS
```

Exercício 2: Fazer um programa para salvar no arquivo exer02.gpkg somente a feição que possui o nome Paraguai no arquivo bacia.gpkg.

Passos principais:

1. Carregar a camada normalmente;
2. Criar a camada em memória para armazenar a feição filtrada;
3. Iterar sobre todas as feições da camada;
4. Verificar se o valor do atributo NOME é igual a 'Paraguai';
5. Adicionar a feição filtrada à nova camada;
6. Exportar para o arquivo exer02.gpkg.

Exercício 3: Fazer um programa para fazer a diferença espacial entre as geometrias dos arquivos exer02.gpkg e pais.gpkg (resultado do Exemplo 5), e salvar o resultado no arquivo exer03.gpkg.

Passos principais:

1 Carregar as camadas exer02.gpkg e pais.gpkg;

```
camada_exer02 = QgsVectorLayer('D:/pasta/exer02.gpkg', 'exer02', 'ogr')
camada_pais = QgsVectorLayer('D:/pasta/pais.gpkg', 'pais', 'ogr')
```

2 Obter a única geometria de cada camada.

```
geom_exer02 = next(camada_exer02.getFeatures()).geometry()
geom_pais = next(camada_pais.getFeatures()).geometry()
```

A função `getFeatures()`retorna um iterador, que permite acessar as feições de uma camada uma a uma, sem carregar todas na memória ao mesmo tempo.

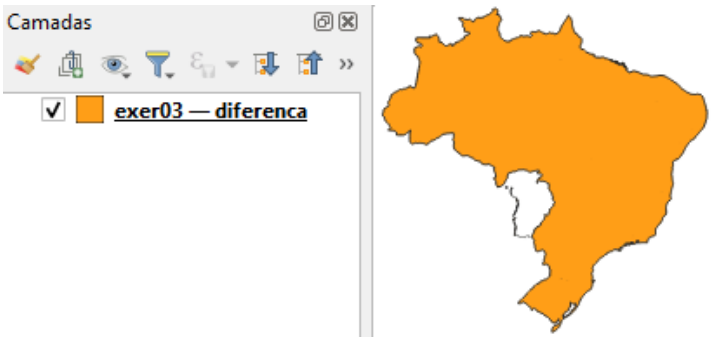
A função `next()` recupera o próximo item de um iterador. Como, no enunciado, sabemos que há apenas uma feição em cada camada, podemos usar `next()` diretamente para obter essa feição de forma simples e eficiente.

3 Realizar a operação de diferença espacial – usando a função difference;

4 Criar uma camada em memória para armazenar o resultado;

5 Exportar o resultado para exer03.gpkg.

Resultado esperado:



Exercício 4: Fazer um programa para obter as cidades que estão contidas na Bacia Hidrográfica do Paraguai, e salvar o resultado no arquivo exer04.gpkg.

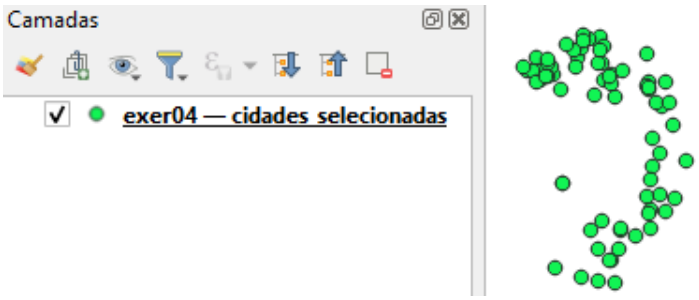
Entrada:

- Camada vetorial com a Bacia Hidrográfica do Paraguai (exer02.gpkg).
- Camada vetorial com os pontos das cidades do pais (cidade.gpkg).

Passos principais:

1 Carregar as camadas: cidades (cidade.gpkg) e exer02;
2 Extrair a geometria da única feição de exer02;
3 Criar camada em memória para armazenar as cidades selecionadas;
4 Iterar sobre cada cidade, verificando com within() se está contida na geometria de exer02;
5 Adicionar as cidades selecionadas na nova camada;
6 Salvar a camada cidades_selecionadas no arquivo exer04.gpkg.

Resultado esperado: 73 cidades.



Exercício 5: Altere o código do Exercício 4 para obter os polígonos dos municípios que estão contidos na Bacia Hidrográfica do Paraguai, e salvar o resultado no arquivo exer05.gpkg.

Entrada:

- Camada vetorial com a Bacia Hidrográfica do Paraguai (exer02.gpkg).
- Camada vetorial com os polígonos dos municípios do pais (municipio.gpkg).

Resultado esperado: 47 municípios.