

RELATÓRIO

ATIVIDADE AZURE

11 DE MARÇO

Criado por:
Stephany Mackyne

GAMA  ACADEMY


accenture

Atividade 1 – Azure (Equivalente a Atividade 09)

Descrição da atividade

Leitura de dados da API do IBGE, com o uso do package requests, para escrever em Banco de Dados relacional alocado no Azure os seguintes dados: UFs, Cidades (relacionadas com UFs) e Distritos (relacionados com Cidades).

Programas e plataformas utilizadas

- Microsoft Azure
- DBeaver
- Visual Studio Code
- Site IBGE (Instituto Brasileiro de Geografia Aplicada)
- Site Nylas

Bibliotecas

- pyodbc
- requests

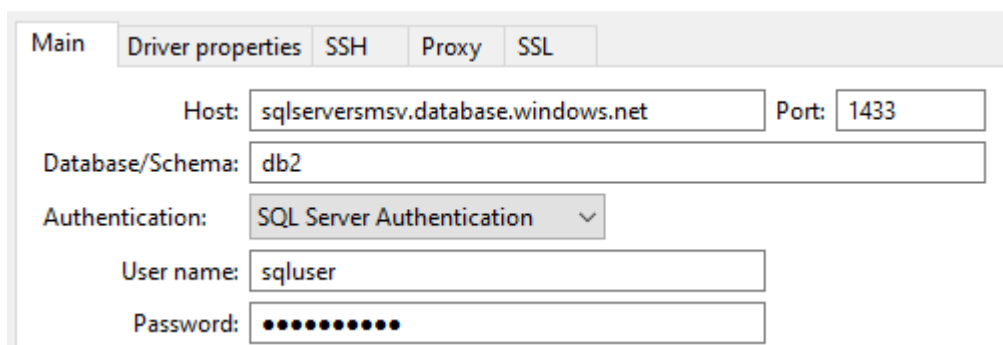
Linguagens de programação

- SQL ANSI
- Python

Metodologia

Inicialmente, é criada uma conta no [portal da Azure](#), preferencialmente de teste gratuito para que não haja cobranças inesperadas, e, logo após, realiza-se login. É criado um recurso de banco de dados, o escolhido na Atividade foi o **SQL Database**, para **SQL Server**, seguindo todas as instruções contidas na Atividade 04.

Com isso, é utilizado o software cliente SQL **DBeaver 7.3.5** para a criação das tabelas do banco alocado no Azure. A conexão com o banco é realizada a partir dos seguintes dados:



The image shows the 'Driver properties' tab in the DBeaver application. The fields are configured as follows:

- Host:** sqlserversmsv.database.windows.net
- Port:** 1433
- Database/Schema:** db2
- Authentication:** SQL Server Authentication (selected from a dropdown menu)
- User name:** sqluser
- Password:** [masked with dots]

em que **Host** é a url do servidor, **Database** o nome do banco de dados e **User name** e **Password** os dados de acesso ao banco referentes a usuário e senha, respectivamente.

No DBeaver, é gerado o seguinte código para criação das tabelas que ficarão armazenadas no banco. A linguagem utilizada foi a SQL ANSI. São criadas três tabelas, estados, município e distritos, em que o “id” de estados é chave estrangeira para municípios e o “id” de municípios é chave estrangeira para distritos.

```
use db2;

create table estados
(
    id int not null,
    sigla varchar(3) not null,
    nome varchar(25) not null,
    constraint pk_estados primary key (id)
);

create table municipios
(
    id int not null,
    nome varchar(45) not null,
    id_estado int not null,
    constraint pk_municipios primary key (id),
    constraint fk_municipios_estados foreign key (id_estado)
    references estados (id)
);

create table distritos
(
    id int not null,
    nome varchar(45) not null,
    id_municipio int not null,
    constraint pk_distritos primary key (id),
    constraint fk_distritos_municipios foreign key (id_municipio)
    references municipios (id)
);
```

O código Python é escrito na IDE Visual Studio Code, na versão 3.9.1 e com o download do módulo python requests, para interação com REST API's, e do pyodbc, para acesso a sistemas gerenciadores de bancos de dados.

Inicia-se importando as duas bibliotecas instaladas. Depois são inseridas as especificações do banco para conexão do código pelo pyodbc e, abaixo, **conn.cursor()** cria um cursor que permite executar comandos SQL.

```
import requests
import pyodbc

conn = pyodbc.connect('Driver={ODBC Driver 17 for SQL Server};'
                      'Server=sqlserversmsv.database.windows.net;'
                      'Database=db2;'
                      'UID=sqluser;'
                      'PWD=Pa$$wOrd1234;')

cursor = conn.cursor()
```

O comando query é para ordenação dos nomes por ordem alfabética. O comando **try** é utilizado para tratamento de erros e exceções. O comando **requests.get()** faz uma consulta **get** na API do [IBGE](https://servicodados.ibge.gov.br/api/v1/localidades/estados). O primeiro laço **for** faz a leitura dos dados dos estados (id, nome e sigla) e, para cada item/estado, o **cursor.execute()** já insere os dados obtidos como registro no banco e o **cursor.commit()** salva as alterações no Azure.

```
query = {"orderBy": "nome"}

try:
    responseEstados = requests.get("https://servicodados.ibge.gov.br/api/v1/localidades/estados", params=query)
    estados = responseEstados.json()
    for estado in estados:
        idEstado = estado["id"]
        nomeEstado = estado["nome"]
        siglaEstado = estado["sigla"]

        cursor.execute("INSERT INTO ESTADOS VALUES (?, ?, ?)", idEstado, siglaEstado, nomeEstado)
    cursor.commit()
except:
    print("Ocorreu um erro! Os dados dos Estados não foram inseridos no banco. \n")
else:
    print("Dados dos Estados inseridos com sucesso! \n")
```

No segundo laço **for**, o comando **try** realiza a mesma função da seção de código acima, e faz a leitura dos municípios (id, nome, id do estado referente), inserindo-os como registros no banco de dados alocado no Azure.

```
try:
    responseMunicipios = requests.get(f"https://servicodados.ibge.gov.br/api/v1/localidades/municipios", params=query)
    municipios = responseMunicipios.json()
    for municipio in municipios:
        idMunicipio = municipio["id"]
        nomeMunicipio = municipio["nome"]
        idEstado = municipio["microrregiao"]["mesorregiao"]["UF"]["id"]

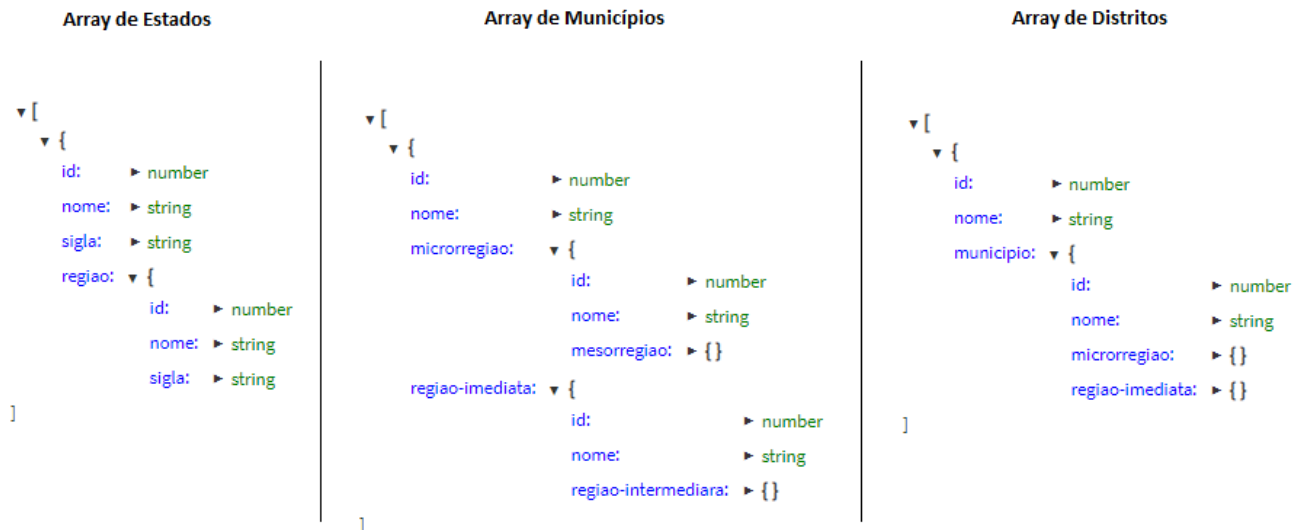
        cursor.execute("INSERT INTO MUNICIPIOS VALUES (?, ?, ?)", idMunicipio, nomeMunicipio, idEstado)
    cursor.commit()
except:
    print("Ocorreu um erro! Os dados dos Municipios não foram inseridos no banco. \n")
else:
    print("Dados dos Municipios inseridos com sucesso! \n")
```

O comando **try** no terceiro laço **for**, além de imprimir o status de erro ou não, imprime também uma mensagem de que o código foi finalizado e parou de rodar, independente se ocorreram erros ou não. A leitura da API retorna os dados de distritos (id, nome, id do município referente), inserindo-os automaticamente como registros no banco. E, por fim, o comando **conn.close()** encerra a conexão com o banco.

```
try:
    responseDistritos = requests.get(f"https://servicodados.ibge.gov.br/api/v1/localidades/distritos", params=query, timeout=300)
    distritos = responseDistritos.json()
    for distrito in distritos:
        idDistrito = distrito["id"]
        nomeDistrito = distrito["nome"]
        idMunicipio = distrito["municipio"]["id"]
        cursor.execute("INSERT INTO DISTRITOS VALUES (?, ?, ?)", idDistrito, nomeDistrito, idMunicipio)
    cursor.commit()
except:
    print("Ocorreu um erro! Os dados dos Distritos não foram inseridos no banco. \n")
else:
    print("Dados dos Distritos inseridos com sucesso! \n")
finally:
    print("Programa finalizado!")

conn.close()
```

A ilustração abaixo é a definida na documentação do site do IBGE, dividida pelo **.json** de estados, municípios e distritos, o que ajuda a identificar a herança de cada objeto.



Situações principais em que o programa não funcionaria

- Instabilidade do sistema de nuvem da Azure
- Instabilidade da API do IBGE
- Modificação da API do IBGE
- Inconsistência dos dados
- Dentre outros

Melhorias futuras

- Inserção de mais colunas nas tabelas, como dados de região, microrregião, região-imediata, etc.
- Criação de novos relacionamentos
- Dentre outros

Comentários

A complexidade do programa foi adequada para exercitar os conceitos iniciais apresentados nas aulas.