





Daniel Abella

Profissional PMP, PMI-ACP e CSM, Gerente de Projetos no Virtus & UniFacisa e Professor do Curso de Sistemas de Informação da UniFacisa. Autor dos livros Gestão A3 (Ágil) e Scrum Arretado.

Python

CRUDs e WebServices com Python

Usando Flask

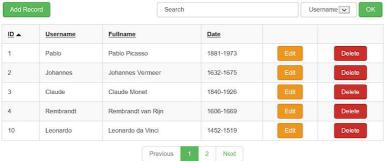
Conteúdo:

- O que é CRUD?
- Operações CRUD no BD
- Criando WebServices com CRUD
- E agora?

0 que é CRUD?

- Um dos termos bastante utilizados na área de Desenvolvimento de Software é CRUD. Mas o que é CRUD?
- Antes de "traduzir" o que significa, observe a tela abaixo.
 - Basicamente, temos uma listagem de usuários (são 5 linhas, uma com Pablo e outras 4 linhas)
 - Em cada uma das linhas, temos um botão que altera os dados de um usuário e um botão oue exclui o usuário.
- Esta funcionalidade inteira (listar, inserir, excluir e editar), chamamos de CRUD. Ou seja, o que acabamos de explicar foi a funcionalidade de CRUD de usuários.
- Mas, afinal, o que é CRUD? CRUD é um acrônimo para C (Create, criar, inserir), R (read, ler), U (Update, atualizar) e D (Delete, Excluir).





- E, caso seja necessário adicionar um novo usuário, clicamos no botão de a tela a seguir é apresentada. Nesta segunda tela, preenchemos os dados de um novo usuário e ao clicar no botão que um novo usuário é criado e voltamos a tela anterior que, ao invés de 5 usuários, agora terá 6.
- ID

 Username

 Pablo

 Fullname

 Pablo Picasso

 Active

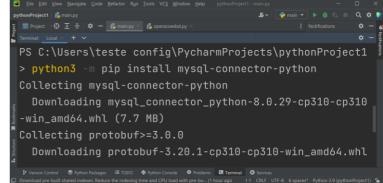
 Country

 Spain

 Role

Operações CRUD no BD

- O pré-requisito é termos o Python e o PIP instalados, após isto, execute o seguinte comando no terminal do PyCharm:
 - python3 -m pip install mysql-connector-python
- Para isso, no PyCharm, clique em Terminal e informe o comando acima, como apresentado a seguir.



- O próximo passo (e bem sugerido) é criar um módulo para guardar as operações no banco de dados. Criaremos o arquivo operacoesbd.py com os seguintes métodos:
 - Método para abrir uma conexão no BD.

```
import mysql.connector

def abrirBancoDados(host,user,password,database):
    return mysql.connector.connect(host_user,
    password_database)
```

 Do mesmo jeito que abrimos uma conexão, precisamos encerrá-la após o uso com o método a seguir.

- Note que, no método que abre a conexão, este retorna uma conexão. Enquanto no método que encerra o banco de dados, recebe a conexão como parâmetro e a encerra.
- Na sequencia, abaixo temos o método que realiza inserção de dados no banco de dados. Note que, precisamos especificar a conexão, a consulta (SQL) e os dados a serem inseridos.

 Para que o exemplo funcione, vamos criar uma tabela chamada produtos

```
1 • ○ create table produtos (
2     id int not null auto_increment primary key,
3     nome varchar(100) not null,
4     descricao varchar(100) not null,
5     preco float not null
6    );
```

- Uma vez criada a tabela e o método acima, podemos usá-los dentro do nosso main.py, conforme exemplo a seguir.
 - · Linha 1: Import do módulo criado antes
 - Linha 3: Criamos a conexão a ser usada
 - Linha 5: Informamos a consulta de INSERT
 - Linha 6: Informamos os dados
 - Linha 7: Chamamos o SQL e os dados usando o método criado
 - Linha 9: Encerramos a conexão criada na linha 3

- No método a seguir, apresentamos todos os registros no banco de dados.
 - Devemos informar a conexão e o SQL da consulta

- Como faço para chamar o método acima? O exemplo está a seguir.
 - Linha 5: Consulta SQL
 - Linha 6: Chamamos o método criado informando a conexão e o SQL (linha 5)
 - Pegamos o resultado e atribuímos a variável resultado
 - Linha 7: Imprimimos o resultado
 - Linha 11: Encerramos a conexão da linha 3

- Seguindo, agora vamos fazer a atualização do BD com o método a seguir.
- Note que, este método retorna a quantidade de linhas que foram atualizadas por meio da consulta.

- Complementamente, o código a seguir realiza o uso do método acima.
 - Linha 7: Chamamos o método criado informando a consulta (linha 5) e os dados (linha 6)

 Por fim, vamos excluir dados do BD, né? O método a seguir faz isto. Note que, é bem parecido com o método de atualizar.

```
def excluirBancoDados(connection, sql, dados):
    cursor = connection.cursor()
    cursor.execute(sql, dados)
    connection.commit()
    linhasAfetadas = cursor.rowcount
    cursor.close()
    return linhasAfetadas
```

 Complementarmente, o código a seguir mostra como chamar o método acima.

O3Criando WebServices que Façam as Operações CRUD

- Na seção anterior, lembram que criamos um módulo com as operações de um banco de dados? Enfim, agora nesta seção, vamos criar o nosso webservice que usa este módulo e possui as seguintes operações:
 - · Linha 10: Método que recebe requisições GET
 - Como todo método dentro de classe, o primeiro parâmetro é um self
 - Note que, na linha 13 colocamos a consulta, que é enviada na linha 14. Na linha 15 encerramos a conexão.
 - Por fim, na linha 16, convertemos o resultado da linha 14 em formato JSON
 - Linha 18: Método que recebe requisições POST
 - Linhas 21,22 e 23: Pegamos os parâmetros enviados no body (corpo) da requisição
 - Na linha 25 temos a consulta, que é enviada na linha 27.
 - Os dois métodos GET e POST (linhas 10 e 17) estão dentro de uma classe chamada Produtos
 - Usamos esta classe para encapsular os 2 métodos
 - Posteriormente, na linha 62, informamos: "pegue os métodos da classe Produtos e faça eles atenderem a requisições em /produtos"
 - De maneira análoga, na linha 43 criamos uma classe para abrigar métodos que recebem parâmetros por URL (tipo, GET ou DELETE passando o ID)

```
from flask_restful import Resource, Api
import operacoesbd
app = Flask(__name__)
   def get(self):
   operacoesbd.insertNoBancoDados(conn,sql,dados)
       'root', '12345', 'testes')

sql = 'SELECT * FROM PRODUTOS where id = ' + id

population
       operacoesbd.encerrarBancoDados(conn)
api.add_resource(Produtos, '/produtos')
api.add_resource(ProdutosPorId, '/produtos/<id>')
```

Por fim, os métodos get (por ID) e delete (por ID) estão nas linhas 45 e 54. Na linha 63, adicionamos esta nova classe, que vai atender a requisições em '/produtos/<id>'.

O4E agora? Acabou?

- Uma recomendação final é que leiam o <u>PEP 8 Style Guide for</u>
 <u>Python Code</u>, pois foi um assunto negligenciado durante esta
 série, pois o objetivo era "startar" sua carreira de desenvolvedor
 usando a linguagem Python.
- Espero que esta série de conteúdos tenham sido de bastante utilidade para você!
- Abraços cordiais, Daniel Abella.