ATIVIDADE 08

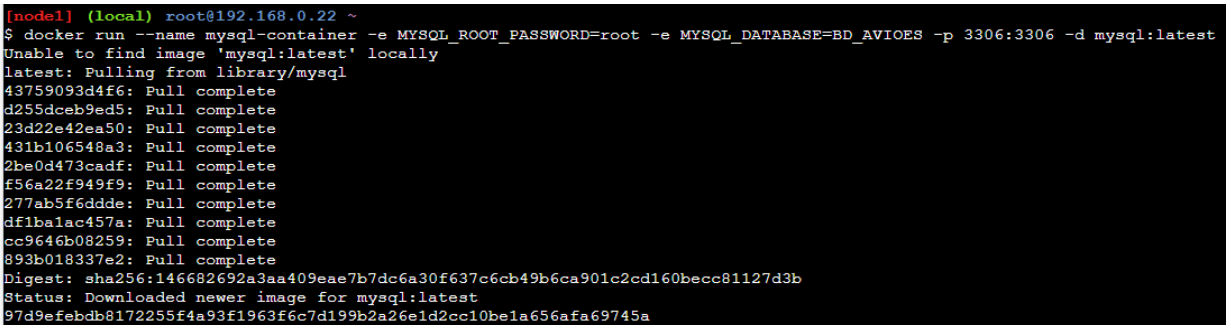
ALUNO: Francisco Thiago Cordeiro de Brito

ENUNCIADO: Implementar os CRUDs usando o Python (Aula do dia 10/02/2024) das tabelas do banco de dados criado na atividade-07. Tarefa em dupla postado no GitHub de cada participante com o link do Github referenciado no Google Classroom.

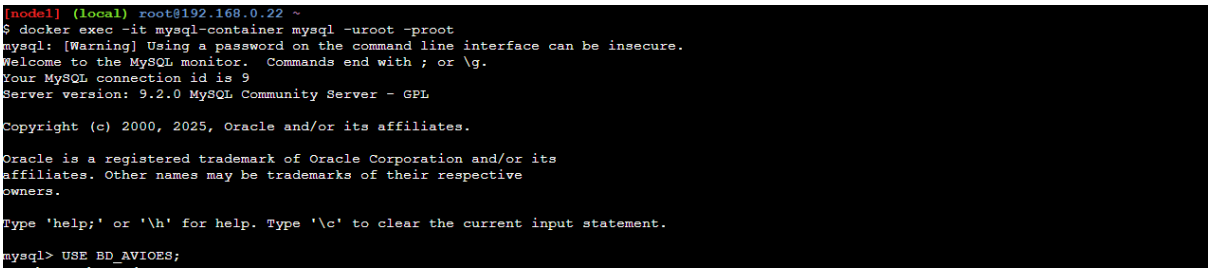
ATIVIDADE

1. Criamos e entramos num container docker

docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=BD_LOJA -p 3306:3306 -d mysql:latest



docker exec -it mysql-container mysql -uroot -proot



2. Criamos o banco de dados nesta instância

```
USE BD_AVIOES;

CREATE TABLE TB_ESCRITORIOS (
id int NOT NULL AUTO_INCREMENT,
cidade text
phone text,
endereco_pt1 text,
endereco_pt2 text,
estado text, pais text,
codigo_postal text,
territorio text,
PRIMARY KEY (id) );

CREATE TABLE TB_FUNCIONARIOS (
id int NOT NULL AUTO_INCREMENT,
nome_ultimo varchar(100),
 nome_primeiro varchar(100),
 extensao varchar(10),
 email varchar(255),
escritorio_id int,
relatorios_para_funcionario_id int,
trabalho varchar(100),
PRIMARY KEY (id),
 FOREIGN KEY (escritorio_id)
REFERENCES TB_ESCRITORIOS (id),
FOREIGN KEY (relatorios_para_funcionario_id) REFERENCES TB_FUNCIONARIOS (id) );

CREATE TABLE TB_CLIENTES (
id int NOT NULL AUTO_INCREMENT,
 nome text, nome_ultimo varchar(100),
 nome_primeiro varchar(100),
telefone text, endereco_pt1 varchar(255),
endereco_pt2 varchar(255),
cidade varchar(50),
estado varchar(50),
 codigo_postal varchar(20),
 pais varchar(50),
funcionario_id int,
limite_credito double,
 PRIMARY KEY (id),
FOREIGN KEY (funcionario_id) REFERENCES TB_FUNCIONARIOS (id) );
CREATE TABLE TB_LINHAS_PRODUTOS (
id int NOT NULL AUTO_INCREMENT,
descricao text,
 descricao_html longtext,
image text,
 PRIMARY KEY (id) );

CREATE TABLE TB_PRODUTOS (
```

```
id int NOT NULL AUTO_INCREMENT,
nome text,
 linha_produto_id int,
 escala text, fornecedor text,
descricao text,
quantidade_estoque int,
preco double, msrp double,
PRIMARY KEY (id),
 FOREIGN KEY (linha_produto_id) REFERENCES TB_LINHAS_PRODUTOS (id) );

 CREATE TABLE TB_PEDIDOS (
id int NOT NULL AUTO_INCREMENT,
data_pedido date,
data_entrega date,
 data_envio date,
status text, comentarios text,
cliente_id int,
PRIMARY KEY (id),
FOREIGN KEY (cliente_id) REFERENCES TB_CLIENTES (id) );

CREATE TABLE TB_DETALHES_PEDIDO (
pedido_id int NOT NULL,
 produto_id int NOT NULL,
quantidade_pedida int,
preco_unitario double,
numero_linha_pedido int,
PRIMARY KEY (pedido_id, produto_id),
 FOREIGN KEY (pedido_id) REFERENCES TB_PEDIDOS (id), FOREIGN KEY (produto_id)
REFERENCES TB_PRODUTOS (id) );

CREATE TABLE TB_PAGAMENTOS ( id int NOT NULL AUTO_INCREMENT, cliente_id int NOT NULL,
data_pagamento date, valor double, PRIMARY KEY (id, cliente_id), FOREIGN KEY (cliente_id)
REFERENCES TB_CLIENTES (id) );
```

3. Criamos, entramos e instalamos o conector no ambiente virtual (outra instância)

python -m venv myenv
source myenv/bin/activate
pip install mysql-connector-python

```
[node2] (local) root@192.168.0.17 ~
$ python -m venv myenv
[node2] (local) root@192.168.0.17 ~
$ source myenv/bin/activate
(myenv) [node2] (local) root@192.168.0.17 ~
$ pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.2.0-py2.py3-none-any.whl.metadata (6.0 kB)
Downloading mysql_connector_python-9.2.0-py2.py3-none-any.whl (398 kB)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.2.0

[notice] A new release of pip is available: 24.2 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
(myenv) [node2] (local) root@192.168.0.17 ~
$
```

4. Criamos o arquivo.py

vi app.py

```python
import mysql.connector
from mysql.connector import Error
from datetime import date

def create_connection():
    """Conecta ao banco de dados MySQL."""
    connection = None
    try:
        connection = mysql.connector.connect(
            host='192.168.0.17',
            port='3306',
            user='root',
            password='root',
            database='BD_AVIOES'
        )
        print("Conexão com o banco de dados estabelecida")
    except Error as e:
        print(f"Erro ao conectar ao banco de dados: {e}")
    return connection

# Funções CRUD para TB_ESCRITORIOS
def create_escritorio(connection, cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio):
    cursor = connection.cursor()
    query = """INSERT INTO TB_ESCRITORIOS (cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio)
               VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"""
    values = (cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio)
    cursor.execute(query, values)
    connection.commit()
    print("Escritório criado")

def read_escritorios(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_ESCRITORIOS")
    return cursor.fetchall()

def update_escritorio(connection, id, cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio):
    cursor = connection.cursor()
    query = """UPDATE TB_ESCRITORIOS SET cidade=%s, phone=%s, endereco_pt1=%s, endereco_pt2=%s, estado=%s, pais=%s, codigo_postal=%s, territorio=%s WHERE id=%s"""
    values = (cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio, id)
    cursor.execute(query, values)
    connection.commit()
```

```python
def update_escritorio(connection, id, cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio):
    values = (cidade, phone, endereco_pt1, endereco_pt2, estado, pais, codigo_postal, territorio, id)
    cursor.execute(query, values)
    connection.commit()
    print("Escritório atualizado")

def delete_escritorio(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_ESCRITORIOS WHERE id=%s", (id,))
    connection.commit()
    print("Escritório removido")

# Funções CRUD para TB_FUNCIONARIOS
def create_funcionario(connection, nome_ultimo, nome_primeiro, extensao, email, escritorio_id, relatorios_para_funcionario_id, trabalho):
    cursor = connection.cursor()
    query = """INSERT INTO TB_FUNCIONARIOS (nome_ultimo, nome_primeiro, extensao, email, escritorio_id, relatorios_para_funcionario_id, trabalho)
    |   |   |   VALUES (%s, %s, %s, %s, %s, %s, %s)"""
    values = (nome_ultimo, nome_primeiro, extensao, email, escritorio_id, relatorios_para_funcionario_id, trabalho)
    cursor.execute(query, values)
    connection.commit()
    print("Funcionário criado")

def read_funcionarios(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_FUNCIONARIOS")
    return cursor.fetchall()

def update_funcionario(connection, id, nome_ultimo, nome_primeiro, extensao, email, escritorio_id, relatorios_para_funcionario_id, trabalho):
    cursor = connection.cursor()
    query = """UPDATE TB_FUNCIONARIOS SET nome_ultimo=%s, nome_primeiro=%s, extensao=%s, email=%s, escritorio_id=%s, relatorios_para_funcionario_id=%s, trabalho=%s
    WHERE id=%s"""
    values = (nome_ultimo, nome_primeiro, extensao, email, escritorio_id, relatorios_para_funcionario_id, trabalho, id)
    cursor.execute(query, values)
    connection.commit()
    print("Funcionário atualizado")

def delete_funcionario(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_FUNCIONARIOS WHERE id=%s", (id,))
    connection.commit()
    print("Funcionário removido")
```

```python
def delete_funcionario(connection, id):
    print("Funcionário removido")

# Funções CRUD para TB_CLIENTES
def create_cliente(connection, nome, nome_ultimo, nome_primeiro, telefone, endereco_pt1, endereco_pt2, cidade, estado, codigo_postal, pais, funcionario_id, limite_credito):
    cursor = connection.cursor()
    query = """INSERT INTO TB_CLIENTES (nome, nome_ultimo, nome_primeiro, telefone, endereco_pt1, endereco_pt2, cidade, estado, codigo_postal, pais, funcionario_id,
    limite_credito)
    |   |   |   VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"""
    values = (nome, nome_ultimo, nome_primeiro, telefone, endereco_pt1, endereco_pt2, cidade, estado, codigo_postal, pais, funcionario_id, limite_credito)
    cursor.execute(query, values)
    connection.commit()
    print("Cliente criado")

def read_clientes(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_CLIENTES")
    return cursor.fetchall()

def update_cliente(connection, id, nome, nome_ultimo, nome_primeiro, telefone, endereco_pt1, endereco_pt2, cidade, estado, codigo_postal, pais, funcionario_id, limite_credito):
    cursor = connection.cursor()
    query = """UPDATE TB_CLIENTES SET nome=%s, nome_ultimo=%s, nome_primeiro=%s, telefone=%s, endereco_pt1=%s, endereco_pt2=%s, cidade=%s, estado=%s, codigo_postal=%s,
    pais=%s, funcionario_id=%s, limite_credito=%s WHERE id=%s"""
    values = (nome, nome_ultimo, nome_primeiro, telefone, endereco_pt1, endereco_pt2, cidade, estado, codigo_postal, pais, funcionario_id, limite_credito, id)
    cursor.execute(query, values)
    connection.commit()
    print("Cliente atualizado")

def delete_cliente(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_CLIENTES WHERE id=%s", (id,))
    connection.commit()
    print("Cliente removido")

# Funções CRUD para TB_LINHAS_PRODUTOS
def create_linha_produto(connection, descricao, descricao_html, image):
    cursor = connection.cursor()
    query = """INSERT INTO TB_LINHAS_PRODUTOS (descricao, descricao_html, image)
    |   |   |   VALUES (%s, %s, %s)"""
    cursor.execute(query, (descricao, descricao_html, image))
    connection.commit()
```

```python
def create_linha_produto(connection, descricao, descricao_html, image):
    print("Linha de produto criada")

def read_linhas_produtos(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_LINHAS_PRODUTOS")
    return cursor.fetchall()

def update_linha_produto(connection, id, descricao, descricao_html, image):
    cursor = connection.cursor()
    query = """UPDATE TB_LINHAS_PRODUTOS SET descricao=%s, descricao_html=%s, image=%s WHERE id=%s"""
    cursor.execute(query, (descricao, descricao_html, image, id))
    connection.commit()
    print("Linha de produto atualizada")

def delete_linha_produto(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_LINHAS_PRODUTOS WHERE id=%s", (id,))
    connection.commit()
    print("Linha de produto removida")

# Funções CRUD para TB_PRODUTOS
def create_produto(connection, nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp):
    cursor = connection.cursor()
    query = """INSERT INTO TB_PRODUTOS (nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp)
               VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"""
    values = (nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp)
    cursor.execute(query, values)
    connection.commit()
    print("Produto criado")

def read_produtos(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PRODUTOS")
    return cursor.fetchall()

def update_produto(connection, id, nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp):
    cursor = connection.cursor()
    query = """UPDATE TB_PRODUTOS SET nome=%s, linha_produto_id=%s, escala=%s, fornecedor=%s, descricao=%s, quantidade_estoque=%s, preco=%s, msrp=%s WHERE id=%s"""
    values = (nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp, id)
```

```python
def update_produto(connection, id, nome, linha_produto_id, escala, fornecedor, descricao, quantidade_estoque, preco, msrp):
    cursor.execute(query, values)
    connection.commit()
    print("Produto atualizado")

def delete_produto(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_PRODUTOS WHERE id=%s", (id,))
    connection.commit()
    print("Produto removido")

# Funções CRUD para TB_PEDIDOS
def create_pedido(connection, data_pedido, data_entrega, data_envio, status, comentarios, cliente_id):
    cursor = connection.cursor()
    query = """INSERT INTO TB_PEDIDOS (data_pedido, data_entrega, data_envio, status, comentarios, cliente_id)
               VALUES (%s, %s, %s, %s, %s, %s)"""
    values = (data_pedido, data_entrega, data_envio, status, comentarios, cliente_id)
    cursor.execute(query, values)
    connection.commit()
    print("Pedido criado")

def read_pedidos(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PEDIDOS")
    return cursor.fetchall()

def update_pedido(connection, id, data_pedido, data_entrega, data_envio, status, comentarios, cliente_id):
    cursor = connection.cursor()
    query = """UPDATE TB_PEDIDOS SET data_pedido=%s, data_entrega=%s, data_envio=%s, status=%s, comentarios=%s, cliente_id=%s WHERE id=%s"""
    values = (data_pedido, data_entrega, data_envio, status, comentarios, cliente_id, id)
    cursor.execute(query, values)
    connection.commit()
    print("Pedido atualizado")

def delete_pedido(connection, id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_PEDIDOS WHERE id=%s", (id,))
    connection.commit()
    print("Pedido removido")

# Funções CRUD para TB_DETALHES_PEDIDO
```

```python
def create_detalhe_pedido(connection, pedido_id, produto_id, quantidade_pedida, preco_unitario, numero_linha_pedido):
    query = """INSERT INTO TB_DETALHES_PEDIDO (pedido_id, produto_id, quantidade_pedida, preco_unitario, numero_linha_pedido)
    VALUES (%s, %s, %s, %s, %s)"""
    values = (pedido_id, produto_id, quantidade_pedida, preco_unitario, numero_linha_pedido)
    cursor.execute(query, values)
    connection.commit()
    print("Detalhe do pedido criado")

def read_detalhes_pedido(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_DETALHES_PEDIDO")
    return cursor.fetchall()

def update_detalhe_pedido(connection, pedido_id, produto_id, quantidade_pedida, preco_unitario, numero_linha_pedido):
    cursor = connection.cursor()
    query = """UPDATE TB_DETALHES_PEDIDO SET quantidade_pedida=%s, preco_unitario=%s, numero_linha_pedido=%s WHERE pedido_id=%s AND produto_id=%s"""
    values = (quantidade_pedida, preco_unitario, numero_linha_pedido, pedido_id, produto_id)
    cursor.execute(query, values)
    connection.commit()
    print("Detalhe do pedido atualizado")

def delete_detalhe_pedido(connection, pedido_id, produto_id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_DETALHES_PEDIDO WHERE pedido_id=%s AND produto_id=%s", (pedido_id, produto_id))
    connection.commit()
    print("Detalhe do pedido removido ")

# Funções CRUD para TB_PAGAMENTOS
def create_pagamento(connection, cliente_id, data_pagamento, valor):
    cursor = connection.cursor()
    query = """INSERT INTO TB_PAGAMENTOS (cliente_id, data_pagamento, valor)
    VALUES (%s, %s, %s)"""
    cursor.execute(query, (cliente_id, data_pagamento, valor))
    connection.commit()
    print("Pagamento criado")

def read_pagamentos(connection):
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM TB_PAGAMENTOS")
    return cursor.fetchall()
```

```python
def update_pagamento(connection, id, cliente_id, data_pagamento, valor):
    query = """UPDATE TB_PAGAMENTOS SET data_pagamento=%s, valor=%s WHERE id=%s AND cliente_id=%s"""
    cursor.execute(query, (data_pagamento, valor, id, cliente_id))
    connection.commit()
    print("Pagamento atualizado")

def delete_pagamento(connection, id, cliente_id):
    cursor = connection.cursor()
    cursor.execute("DELETE FROM TB_PAGAMENTOS WHERE id=%s AND cliente_id=%s", (id, cliente_id))
    connection.commit()
    print("Pagamento removido")

def main():
    connection = create_connection()
    if connection is None:
        return

    # CREATE de cada tabela
    create_escritorio(connection, "Rio de Janeiro", "21999999999", "Avenida Atlântica", "500", "RJ", "Brasil", "22021001", "Sudeste")
    create_funcionario(connection, "Mendes", "Carlos", "202", "carlos.mendes@email.com", 1, None, "Gerente de Vendas")
    create_cliente(connection, "Fernanda Costa", "Costa", "Fernanda", "21988887777", "Rua das Palmeiras", "45", "Rio de Janeiro", "RJ", "22031002", "Brasil", 1, 7000.00)
    create_linha_produto(connection, "Miniaturas de Aviões", "<p>Réplicas detalhadas de aviões comerciais e militares</p>", "miniaturas_avioes.jpg")
    create_produto(connection, "Boeing 747 Escala 1:200", 1, "1:200", "SkyModels", "Réplica detalhada do Boeing 747", 50, 349.99, 429.99)
    create_pedido(connection, date(2024, 3, 1), date(2024, 3, 12), date(2024, 3, 5), "Processando", "Pedido em separação", 1)
    create_detalhe_pedido(connection, 1, 1, 1, 349.99, 1)
    create_pagamento(connection, 1, date(2024, 3, 5), 349.99)

    # READ de cada tabela
    print("\nREAD de todas as tabelas:")
    print("\nEscritórios: ", read_escritorios(connection))
    print("\nFuncionários: ", read_funcionarios(connection))
    print("\nClientes: ", read_clientes(connection))
    print("\nLinhas de Produtos: ", read_linhas_produtos(connection))
    print("\nProdutos: ", read_produtos(connection))
    print("\nPedidos: ", read_pedidos(connection))
    print("\nDetalhes de Pedido: ", read_detalhes_pedido(connection))
    print("\nPagamentos: ", read_pagamentos(connection))

    # UPDATE de cada tabela
    update_escritorio(connection, 1, "Rio de Janeiro", "21999999999", "Avenida Atlântica", "800", "RJ", "Brasil", "22031002", "Sudeste")
    update_funcionario(connection, 1, "Mendes", "Carlos Eduardo", "202", "carlos.mendes@email.com", 1, None, "Gerente de Operações")
```

```python
    # DELETE de cada tabela
    delete_pagamento(connection, 1, 1)
    delete_detalhe_pedido(connection, 1, 1)
    delete_pedido(connection, 1)
    delete_produto(connection, 1)
    delete_linha_produto(connection, 1)
    delete_cliente(connection, 1)
    delete_funcionario(connection, 2)
    delete_escritorio(connection, 1)

    # Fechar conexão
    connection.close()

if __name__ == "__main__":
    main()
```

5. Execute o arquivo

python app.py

```
$ python app.py
Conexão com o banco de dados estabelecida
Escritório criado
Funcionário criado
Cliente criado
Linha de produto criada
Produto criado
Pedido criado
Detalhe do pedido criado
Pagamento criado
```

```
READ de todas as tabelas:

Escritórios: [(1, "Rio de Janeiro", "21999999999", "Avenida Atlântica", "500",
"RJ", "Brasil", "22021001", "Sudeste")]

Funcionários: [(1, "Mendes", "Carlos", "202", "carlos.mendes@email.com", 1, None,
"Gerente de Vendas")]


Clientes: [(1, "Fernanda Costa", "Costa", "Fernanda", "21988887777", "Rua das
Palmeiras", "45", "Rio de Janeiro", "RJ", "22031002", "Brasil", 1, 7000.00)]

Linhas de Produtos: [(1, "Miniaturas de Aviões", "<p>Réplicas detalhadas de aviões
comerciais e militares</p>", "miniaturas_avioes.jpg")]

Produtos: [(1, "Boeing 747 Escala 1:200", 1, "1:200", "SkyModels", "Réplica
detalhada do Boeing 747", 50, 349.99, 429.99)]
```

```
Produtos: [(1, "Boeing 747 Escala 1:200", 1, "1:200", "SkyModels", "Réplica
detalhada do Boeing 747", 50, 349.99, 429.99)]

Pedidos: [(1, date(2024, 2, 2), date(2024, 2, 11), date(2024, 2, 6), "Entregue",
"Entrega expressa", 2)
]

Detalhes de Pedido: [(1, 1, 1, 349.99, 1)]

Pagamentos: [(1, 1, date(2024, 3, 5), 349.99)]
```

```
Escritório atualizado
Funcionário atualizado
Cliente atualizado
Linha de produto atualizada
Produto atualizado
Pedido atualizado
Detalhe do pedido atualizado
Pagamento atualizado
```

```
Escritório removido
Funcionário removido
Cliente removido
Linha de produto removida
Produto removido
Pedido removido
Detalhe do pedido removido
Pagamento removido
```