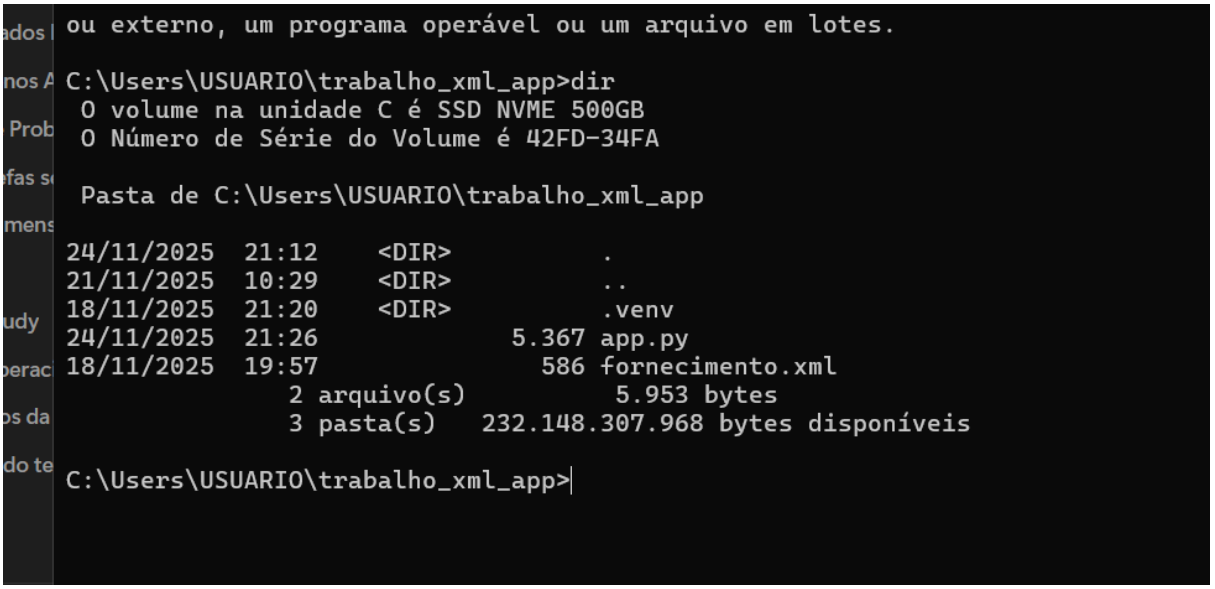
Trabalho Etapa 2 XML

✅ ETAPA 2 – Integração dos Dados do PostgreSQL com o XML usando Python

Primeiro criei a pasta que iria ficar o projeto:

```
C:\Users\USUARIO\trabalho_xml_app
```

Dentro da pasta coloquei fronecimento.xml e a pasta app.py que posteriormente criei no vscode



Aqui podemos ver tyodos os arquivos que posteriormente terão na pasta.
Após a criação do arquivo app.py:

No terminal do vscode rodei esse comando para a criação do ambiente virtual:

```
python -m venv .venv
```

Para ativar ele:

```
.\.venv\Scripts\activate
```

Instalei as dependências necessárias:

```
pip install psycopg2-binary
```

```
PS C:\Users\USUARIO\trabalho_xml_app> .\.venv\Scripts\Activate.ps1
>>
(.venv) PS C:\Users\USUARIO\trabalho_xml_app> pip install psycopg2-binary
>>
Requirement already satisfied: psycopg2-binary in c:\users\usuario\trabalho_xml_app\.venv
\lib\site-packages (2.9.11)
(.venv) PS C:\Users\USUARIO\trabalho_xml_app> dir
>>
```

```
(.venv) PS C:\Users\USUARIO\trabalho_xml_app> dir
>>

    Diretório: C:\Users\USUARIO\trabalho_xml_app


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        18/11/2025     21:20                .venv
-a----        24/11/2025     21:07           5408 app.py
-a----        18/11/2025     19:57            650 fornecedor.xml
-a----        18/11/2025     19:57            586 fornecimento.xml
```

Após isso eu comecei a configuração do banco no postgres

No powershell como admin rodei:

```
psql -U postgres
```

Ai digitei minha senha e entrei no banco.

```
(.venv) C:\Users\USUARIO\trabalho_xml_app>psql -U postgres
Senha para o usuário postgres:

AVISO:  o banco de dados "postgres" possui uma nÃ£o correspondÃªncia de versÃ£o de ordenaÃ§Ã£o
DETALHE:  The database was created using collation version 1539.5,1539.5, but the operating system
.2,1541.2.
DICA:  Rebuild all objects in this database that use the default collation and run ALTER DATABASE p
TION VERSION, or build PostgreSQL with the right library version.
psql (17.4)
ADVERTÊNCIA: A página de código da console (850) difere da página de código do Windows (1252)
             os caracteres de 8 bits podem não funcionar corretamente. Veja a página de
             referência do psql "Notes for Windows users" para obter detalhes.
Digite "help" para obter ajuda.

postgres=# \c trabalho_xml
Agora você está conectado ao banco de dados "trabalho_xml" como usuário "postgres".
trabalho_xml=#
```

Aqui acessei o banco que criei para o trabalho usando o comando:

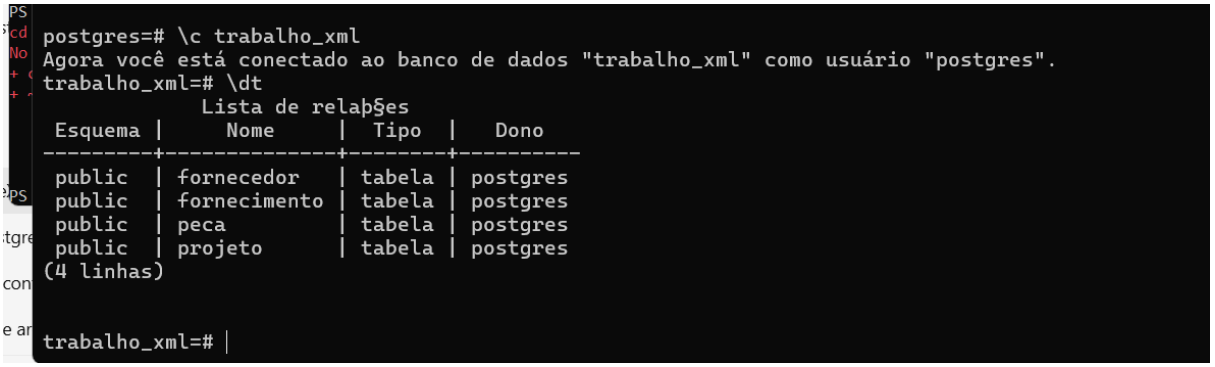
CREATE DATABASE trabalho_xml;

Dentro desse banco subi todo o script que baixei da minha uno, esse aqui:

postcriatab.sql

Para verificar quais tabelas foram criadas:

```
\dt
```



Depois disso eu tive um problema com a senha do postgres entao tive que trocar por algo bem simples, usei:

```
ALTER USER postgres WITH PASSWORD 'postgres';
```

Depois de ter feito tudo isso parti para a criação do arquivo app.py:

```
import psycopg2
import xml.etree.ElementTree as ET

# ============================
# CONFIG DO POSTGRES
# ============================
DB_CONFIG = {
    "host": "localhost",
    "port": 5432,
    "dbname": "trabalho_xml",
```

```
    "user": "postgres",
    "password": "postgres"
}

def get_connection():
    """
    Abre conexão com o PostgreSQL usando a config acima.
    """
    conn = psycopg2.connect(**DB_CONFIG)
    # Se der problema de acentuação, isso ajuda:
    conn.set_client_encoding("UTF8")
    return conn


# ============================
# CARREGAR DADOS DO POSTGRES
# ============================
def carregar_dados_relacionais():
    """
    Lê Fornecedor, Peca e Projeto do banco e guarda em dicionários.
    As chaves vão ser F1, P1, J1 para casar DIRETO com o XML.
    """
    conn = get_connection()
    cur = conn.cursor()

    # ---------- Fornecedor ----------
    cur.execute("SELECT cod_fornec, fnome, status, cidade FROM Forneced
or;")
    fornecedores = {}
    for cod, nome, status, cidade in cur.fetchall():
        # cod é numeric, convertemos pra int e montamos F1, F2, ...
        chave = f"F{int(cod)}"
        fornecedores[chave] = {
            "codigo": int(cod),
            "nome": nome,
            "status": int(status) if status is not None else None,
            "cidade": cidade,
        }
```

```
    # ---------- Peca ----------
    cur.execute("SELECT cod_peca, pnome, cor, peso, cdade FROM Peca;")
    pecas = {}
    for cod, nome, cor, peso, cdade in cur.fetchall():
        chave = f"P{int(cod)}"
        pecas[chave] = {
            "codigo": int(cod),
            "nome": nome,
            "cor": cor,
            "peso": float(peso) if peso is not None else None,
            "cidade": cdade,
        }

    # ---------- Projeto ----------
    cur.execute("SELECT cod_proj, jnome, cidade FROM Projeto;")
    projetos = {}
    for cod, nome, cidade in cur.fetchall():
        chave = f"J{int(cod)}"
        projetos[chave] = {
            "codigo": int(cod),
            "nome": nome,
            "cidade": cidade,
        }

    cur.close()
    conn.close()
    return fornecedores, pecas, projetos


# ============================
# CARREGAR FORNECIMENTOS DO XML
# ============================
def carregar_fornecimentos_xml(caminho_xml: str):
    """
    Lê o arquivo fornecimento.xml no formato:

    <dados>
```

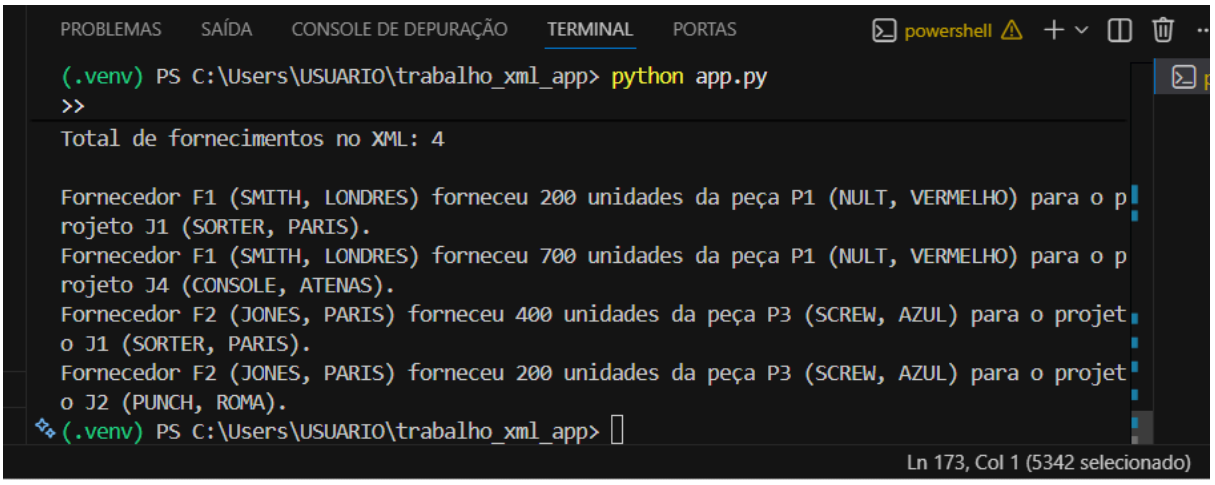
```
    <fornecimento>
      <Cod_Fornec>F1</Cod_Fornec>
      <Cod_Peca>P1</Cod_Peca>
      <Cod_Proj>J1</Cod_Proj>
      <Quantidade>200</Quantidade>
    </fornecimento>
    ...
  </dados>
  """
  tree = ET.parse(caminho_xml)
  root = tree.getroot()

  fornecimentos = []
  for f in root.findall("fornecimento"):
      cod_fornec = f.findtext("Cod_Fornec")
      cod_peca = f.findtext("Cod_Peca")
      cod_proj = f.findtext("Cod_Proj")
      qtd_text = f.findtext("Quantidade")

      quantidade = int(qtd_text) if qtd_text is not None else 0

      fornecimentos.append({
          "cod_fornec": cod_fornec,   # ex: F1
          "cod_peca": cod_peca,       # ex: P1
          "cod_proj": cod_proj,       # ex: J1
          "quantidade": quantidade,
      })

  return fornecimentos


# ============================
# GERAR RELATÓRIO INTEGRADO
# ============================
def gerar_relatorio(fornecedores, pecas, projetos, fornecimentos):
  """
  Para cada fornecimento do XML, faz a junção com os dados do Postgre
s.
```

```
    Só imprime quando fornecedor, peça e projeto existem no banco.
    """
    print("=== RELATÓRIO INTEGRANDO XML + POSTGRES ===\n")
    print(f"Total de fornecimentos no XML: {len(fornecimentos)}\n")

    for f in fornecimentos:
        cod_fornec_xml = f["cod_fornec"]  # F1, F2...
        cod_peca_xml = f["cod_peca"]      # P1, P2...
        cod_proj_xml = f["cod_proj"]      # J1, J2...
        qtd = f["quantidade"]

        # Se qualquer um não existir no banco, pula
        if cod_fornec_xml not in fornecedores:
            print(f"[IGNORADO] Fornecedor {cod_fornec_xml} não existe no ba
nco.")
            continue

        if cod_peca_xml not in pecas:
            print(f"[IGNORADO] Peça {cod_peca_xml} não existe no banco.")
            continue

        if cod_proj_xml not in projetos:
            print(f"[IGNORADO] Projeto {cod_proj_xml} não existe no banco.")
            continue

        forn = fornecedores[cod_fornec_xml]
        peca = pecas[cod_peca_xml]
        proj = projetos[cod_proj_xml]

        print(
            f"Fornecedor {cod_fornec_xml} ({forn['nome']}, {forn['cidade']}) "
            f"forneceu {qtd} unidades da peça {cod_peca_xml} "
            f"({peca['nome']}, {peca['cor']}) "
            f"para o projeto {cod_proj_xml} ({proj['nome']}, {proj['cidade']})."
        )


def main():
```

```
    # 1) Carrega dados relacionais (Postgres)
    fornecedores, pecas, projetos = carregar_dados_relacionais()

    # 2) Carrega fornecimentos do XML
    # Certifique-se que "fornecimento.xml" está na MESMA PASTA do app.p
y
    fornecimentos = carregar_fornecimentos_xml("fornecimento.xml")

    # 3) Gera a junção / integração
    gerar_relatorio(fornecedores, pecas, projetos, fornecimentos)


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

Após isso eu dei um CTRL + S e rodei no terminal do vscode com o venv ativado:

```
python app.py
```



O resultado da foto é esse:

```
(.venv) PS C:\Users\USUARIO\trabalho_xml_app> python app.py
```

```
=== RELATÓRIO INTEGRANDO XML + POSTGRES ===
```

```
Total de fornecimentos no XML: 4
```

```
Fornecedor F1 (SMITH, LONDRES) forneceu 200 unidades da peça P1 (NUL
T, VERMELHO) para o projeto J1 (SORTER, PARIS).
Fornecedor F1 (SMITH, LONDRES) forneceu 700 unidades da peça P1 (NUL
T, VERMELHO) para o projeto J4 (CONSOLE, ATENAS).
Fornecedor F2 (JONES, PARIS) forneceu 400 unidades da peça P3 (SCRE
W, AZUL) para o projeto J1 (SORTER, PARIS).
Fornecedor F2 (JONES, PARIS) forneceu 200 unidades da peça P3 (SCRE
W, AZUL) para o projeto J2 (PUNCH, ROMA).
(.venv) PS C:\Users\USUARIO\trabalho_xml_app>
```