

EDC - Graduação em Engenharia de Computação
Projeto de Bloco: Fundamentos de Dados

TP4 - API Python com Looker



Professor: Alcione Santos Dolavale.
Aluna: Letícia Ferreira Augusto Alves Gomes.



Instituto Infnet.
Brasil, 24 de fevereiro de 2025.
Professor: Alcione Santos Dolavale.
Aluna: Letícia Gomes.
Projeto Bloco: Fundamentos Dados.

Sumário

1. Introdução	05
1.1. Breve Descrição do Código	05
1.2. Propósito do Projeto	05
1.3. Localização do Projeto	05
1.4. Estrutura Diagrama ER	05
2. Análise do Código	06
2.1. Diretório Raiz do Projeto	06
2.2. <code>criacao_csv.py</code>	07
2.2.1. Importação do pandas	07
2.2.2. Lista Inicializadas	07
2.2.3. Função <code>addFuncionario()</code>	07
2.2.4. Função <code>addCargo()</code>	07
2.2.5. Função <code>addDepartamento()</code>	08
2.2.6. Função <code>addHistoricoSalarial()</code>	08
2.2.7. Função <code>addDependente()</code>	08
2.2.8. Função <code>addProjeto()</code>	09
2.2.9. Função <code>addRecurso()</code>	09
2.2.10. Populando Listas	09
2.2.11. Função <code>InserirDadosCsv()</code>	10
2.2.12. Parâmetro Função <code>inserirDadosCsv()</code>	10
2.3. <code>conexao_db.py</code>	11
2.3.1. Importação do SQLite3	11
2.3.2. Função <code>conectar()</code>	11
2.3.3. Função <code>desconectar()</code>	11
2.4. <code>modelos.py</code>	12
2.4.1. Sequências de Escape ANSI	12
2.4.2. Classe Metadado	12



2.5. crud.py	14
2.5.1. Importação de Módulos	14
2.5.2. Gerenciando Conexão	14
2.5.3. Função lerDadosCsv()	14
2.5.4. Função consultar_tabelas()	15
2.5.5. Função criarEndpoint()	17
2.5.6. Função criar_tabelas()	18
2.5.7. Função criar_insert()	18
2.5.8. Execução do fastapi	19
2.6. criacao_tabelas.py	20
2.6.1. Importação de Módulo	20
2.6.2. Função executar_tabelas()	20
2.6.3. Leitura de Dados	24
2.6.4. Inserção de Dados	24
2.7. comandos.py	28
2.7.1. Importação de Módulos	28
2.7.2. Chamada da Função executar_tabelas()	28
2.7.3. Sequências de Escape ANSI	28
2.7.4. TP1.1	29
2.7.5. TP1.2	30
2.7.6. TP1.3	31
2.7.7. TP1.4	32
2.7.8. TP1.5	33
2.7.9. Desconectar Base de Dados	33
2.8. requirements.txt	34
2.8.1. dependências	34
2.9. render.yaml	34
2.9.1. services	34
3. Deploy	35
3.1. Github	35
3.2. Render	36



4. Looker Studio	39
4.1. Conectando TPs	39
4.2. Capa Dashboard	40
4.3. TP4.1	41
4.4. TP4.3	41
4.5. TP4.4	42

Introdução

◆ Breve Descrição

Este projeto foi feito a partir das ferramentas: **SQLite3** e **Python**, com dados de uma empresa fictícia de tecnologia, contendo as tabelas:

- ◆ **tb_Funcionario:** 21 dados fictícios de funcionários da empresa;
- ◆ **tb_Cargo:** 13 dados fictícios de cargos da empresa;
- ◆ **tb_Departamento:** 5 dados fictícios de departamentos da empresa;
- ◆ **tb_HistoricoSalarial:** 126 dados fictícios do histórico salarial da empresa nos últimos 6 meses;
- ◆ **tb_Dependente:** 44 dados fictícios de dependentes de funcionários.
- ◆ **tb_Projeto:** 11 dados fictícios de projetos da empresa;
- ◆ **tb_Recurso:** 16 dados fictícios de recursos de projetos da empresa.

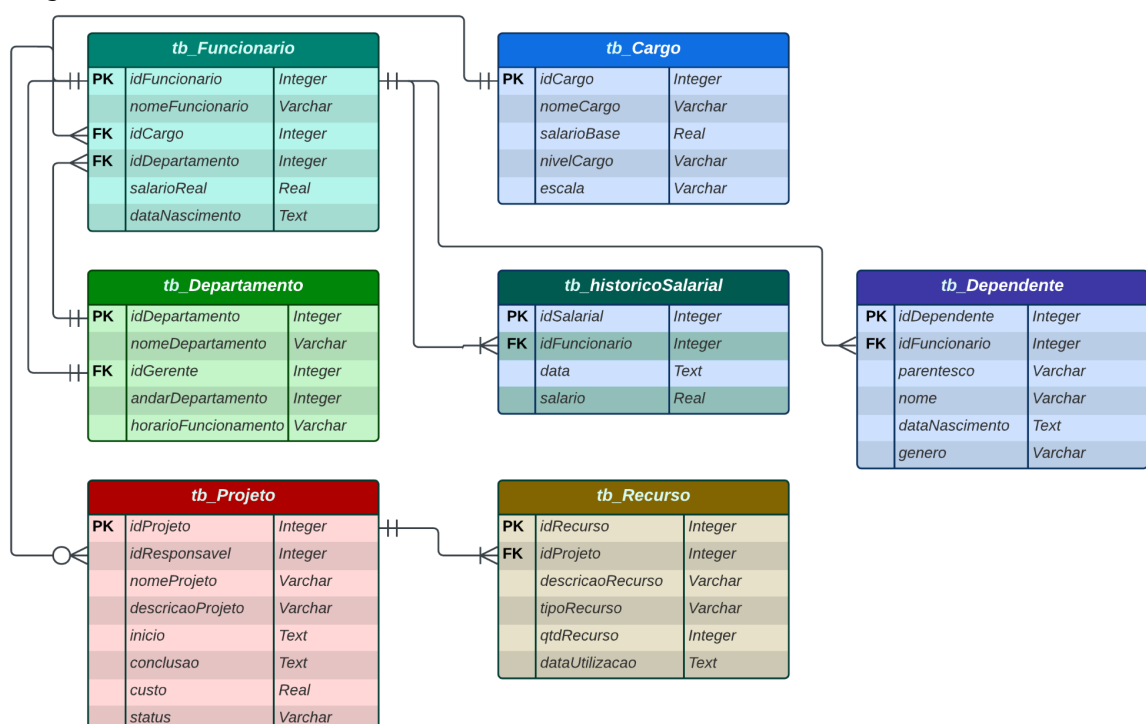
◆ Localização dos Dados

Os dados criados e consultas foram separados entre os arquivos: [comandos.py](#), [conexao_db.py](#), [criacao_tabelas.py](#), [criacao_csv.py](#), [crud.py](#) e [modelos.py](#)

- **Repositório:**
https://github.com/LeticiaFAAGomes/PB_TP4
- **TPs no Render:**
<https://pb-tp4.onrender.com/tp4.1> | <https://pb-tp4.onrender.com/tp4.3> | <https://pb-tp4.onrender.com/tp4.4>
- **Dashboard no Looker:**
<https://lookerstudio.google.com/reporting/24c86f5c-b4ce-495c-b7fd-463de6b71bf5>

◆ Estrutura Diagrama ER

Diagrama **Entidade-Relacionamento** dos dados utilizados.



Análise do Código

A seguir estão informações sobre todo o projeto desenvolvido pela aluna.

◆ Diretório Raiz do Projeto

O projeto foi desenvolvido com a estrutura abaixo:

Diretório Raiz

- TP4- PB
 - api
 - tabelas
 - tb_cargo.csv
 - tb_departamento.csv
 - tb_dependente.csv
 - tb_funcionario.csv
 - tb_historicoSalarial.csv
 - tb_projeto.csv
 - tb_recurso.csv
 - comandos.py
 - conexao_db.py
 - criacao_csv.py
 - criacao_tabelas.py
 - crud.py
 - modelos.py
 - .gitignore
 - db_empresaTech.db
 - README.md
 - render.yaml
 - requirements.txt

Letícia_Ferreira_Augusto_Alves_Gomes__PB__TP3/

- tabelas** ← Pasta onde os arquivos **.CSV** estão **armazenados**;
 - tb_cargo.csv** ← Tabela de **Cargos**;
 - tb_departamento.csv** ← Tabela de **Departamentos**;
 - tb_dependente.csv** ← Tabela de **Dependentes**;
 - tb_funcionario.csv** ← Tabela de **Funcionários**;
 - tb_historicoSalarial.csv** ← Tabela de **Históricos Salariais**.
- api** ← Pasta onde os arquivos **.py** estão **armazenados**;
 - comandos.py** ← Arquivo onde todas as **consultas** foram desenvolvidas;
 - conexao_db.py** ← Arquivo que gerencia a **conexão** com o banco de dados do **SQLite3**.
 - criacao_csv.py** ← Arquivo onde houve a **criação** dos **.CSV** armazenados na pasta **tabelas**;
 - criacao_tabelas.py** ← Arquivo que gerencia na **criação** e **população** dos dados no **SQLite3**;
 - crud.py** ← Arquivo que gerencia a **execução** das consultas **SQL**;
 - modelos.py** ← Arquivo que armazena **classes** utilizadas nas **consultas**.
- db_empresa_tech.db** ← Arquivo do banco de dados do **SQLite3**;
- README.md** ← Arquivo **markdown**;
- render.yaml** ← Arquivo **yaml** para **deploy** no **render.com**;
- requirements.txt** ← Arquivo com as **dependências** a serem **instaladas**.

◆ criacao_csv.py

- Importação do pandas

1. **Biblioteca pandas:**

Biblioteca utilizada para manipular arquivos como **.CSV**.

Importação do pandas

```
import pandas as pd
```

- Listas Inicializadas

Inicialização de listas:

Listas inicializadas

```
funcionarios, cargos, departamentos, historicoSalarial, dependentes,  
projetos, recursos = [], [], [], [], [], [], []
```

- Função addFuncionario()

Esta função adiciona os dados de um funcionário na lista de funcionários:

Função addFuncionario()

```
def addFuncionario(nomeFuncionario, idCargo, idDepartamento,  
salarioReal, dataNascimento):  
    funcionarios.append({'idFuncionario':len(funcionarios)+1,  
                        'nomeFuncionario':nomeFuncionario,  
                        'idCargo': idCargo,  
                        'idDepartamento': idDepartamento,  
                        'salarioReal': salarioReal,  
                        'dataNascimento': dataNascimento})
```

- Função addCargo()

Esta função adiciona os dados de um cargo na lista de cargos:

Função addCargo()

```
def addCargo(nomeCargo, salarioBase, nivelCargo, escala):  
    cargos.append({'idCargo':len(cargos)+1,  
                  'nomeCargo':nomeCargo,  
                  'salarioBase':salarioBase,  
                  'nivelCargo':nivelCargo,  
                  'escala':escala})
```

- **Função addDepartamento()**

Esta função adiciona os dados de um departamento na lista de departamentos:

Função addDepartamento()

```
def addDepartamento(nomeDepartamento, idGerente, andarDepartamento,
    horarioFuncionamento):
    departamentos.append({'idDepartamento':len(departamentos)+1,
        'nomeDepartamento':nomeDepartamento,
        'idGerente':idGerente,
        'andarDepartamento':andarDepartamento,
        'horarioFuncionamento':horarioFuncionamento})
```

- **Função addHistoricoSalarial()**

Esta função adiciona os dados de um histórico salarial na lista de historicoSalarial:

Função addHistoricoSalarial()

```
def addHistoricoSalarial(idFuncionario, data, salario):
    historicoSalarial.append({'idSalarial':len(historicoSalarial)+1,
        'idFuncionario':idFuncionario,
        'data':data,
        'salario':salario})
```

- **Função addDependente()**

Esta função adiciona os dados de um dependente na lista de dependentes:

Função addDependente()

```
def addDependente(idFuncionario, parentesco, nome, idade, genero):
    dependentes.append({'idDependente':len(dependentes)+1,
        'idFuncionario':idFuncionario,
        'parentesco':parentesco,
        'nome':nome,
        'idade':idade,
        'genero':genero})
```


- **Função addProjeto()**

Esta função adiciona os dados de um projeto na lista de projetos:

Função addProjeto()

```
def addProjeto(idResponsavel, nomeProjeto, descricao, inicio,
               conclusao, custo, status):
    projetos.append({'idProjeto':len(projetos)+1,
                    'idResponsavel':idResponsavel,
                    'nomeProjeto':nomeProjeto,
                    'descricaoProjeto':descricao,
                    'inicio':inicio,
                    'conclusao':conclusao,
                    'custo':custo,
                    'status':status})
```

- **Função addRecurso()**

Esta função adiciona os dados de um recurso na lista de recursos:

Função addRecurso()

```
def addRecurso(idProjeto, descricao, tipo, qtd, dataUtilizacao):
    recursos.append({'idRecurso':len(recursos)+1,
                    'idProjeto':idProjeto,
                    'descricaoRecurso':descricao,
                    'tipoRecurso':tipo,
                    'qtdRecurso':qtd,
                    'dataUtilizacao':dataUtilizacao})
```

- **Populando Listas**

Depois cada **dado** é passado como [parâmetro](#) para as **funções correspondentes**:

Populando Listas

```
addFuncionario(nomeFuncionario, idCargo, idDepartamento, salarioReal,
               dataNascimento)
addCargo(nomeCargo, salarioBase, nivelCargo, escala)
addDepartamento(nomeDepartamento, idGerente, andarDepartamento,
                horarioFuncionamento)
addHistoricoSalarial(idFuncionario, data, salario)
addDependente(idFuncionario, parentesco, nome, idade, genero)
addProjeto(idResponsavel, nomeProjeto, descricao, inicio, conclusao,
           custo, status)
addRecurso(idProjeto, descricao, qtd, dataUtilizacao)
```



- **Função `inserirDadosCsv()`**

Esta função cria um arquivo **.CSV** e insere os dados **armazenados** de uma lista nesse arquivo **.CSV** :

Função `inserirDadosCsv()`

```
def inserirDadosCsv(nomeArquivo, lista, *campos):  
    df = pd.DataFrame(lista)  
    df.to_csv(f'api/tabelas/tb_{nomeArquivo}.csv',  
             encoding='UTF-8', index=False)  
    print(f'{nomeArquivo.title()}s adicionados com sucesso!\n')
```

- **Parâmetro Função `inserirDadosCsv()`**

Depois cada nome de **arquivo**, **lista** de **dados e campos** são passados como **parâmetro** para a **função `inserirDadosCsv()`**:

Função `inserirDadosCsv()`

```
inserirDadosCsv(nomeArquivo, lista, *campos)
```

◆ conexao_db.py

● Importação do SQLite3

A biblioteca `sqlite3` foi utilizada para a **conexão** entre o **projeto** e o banco de dados **SQLite3**.

Importação do SQLite3

```
import sqlite3
```

● Função conectar()

A função `conectar()` estabelece uma **conexão ativa** com o banco de dados **SQLite3** a partir do nome do base de dados. Caso não haja nenhum erro, a função retornará a instância para **conexão ativa** armazenada na variável `conn`, senão a **conexão não** será **estabelecida**.

Função `conectar()`

```
def conectar():
    try:
        conn = sqlite3.connect('db_empresaTech.db')
        return conn
    except Exception as ex:
        print(ex)
        exit()
```

● Função desconectar()

A função `desconectar()` **salva** as alterações e encerra a **conexão ativa** estabelecida pela função `conectar()` — caso ela esteja **ativa** — com o banco de dados **SQLite3**.

Função `desconectar()`

```
def desconectar(conn):
    if (conn):
        conn.commit()
        conn.close()
```

◆ modelos.py

- Sequências de Escape ANSI

Estas **variáveis** armazenam **sequências de escape ANSI** para formatar a **cor** das **strings** da **classe Metadado**.

Cores para formatação

```
azul, negrito, reset = '\x1b[38;5;117m', '\033[1m', '\x1b[22m'
```

- Classe Metadado

Esta classe **inicializa** todos os dados de **Metadado** que foram **instanciados** com **palavras-chave**:

Dado	Tipo
**dados	<i>dict</i>

Inicialização da Classe **Metadado**

```
class Metadado:  
    def __init__(self, **dados):  
        self.dados = dados
```

Esta esta função **encontra** o tamanho **máximo** de **caracteres** que uma lista pode **ocupar** a partir de uma **lista de dicionário** e **palavras-chave**.

Função *encontrarTamanho()* da Classe **Metadado**

```
def encontrarTamanho(self, lista, cabecalho):  
    return max(len(str(dado.dados.get(cabecalho, ''))) for dado  
in lista) + 4
```

Esta esta função retorna uma **tabela** dos **dados** em formato de **string**, identificando o **cabeçalho** dos dicionários com o **método** `.keys()`, depois um **loop** é iniciado para encontrar o **tamanho** de cada **dicionário** a partir da **lista de dicionários** e as **respectivas palavras-chave** com a função `encontrarTamanho()`.

Depois os cabeçalhos da tela são **formatados** com seus respectivos **tamanhos**. A **quantidade** de **caracteres armazenada** na no **dicionario tamanhos** é **somada**. A lista começa a ser **formatada** e **passada** para a lista **resultado** até ser **retornada**.

Função `formatar()` da Classe `Metadado`

```
def formatar(self, lista):
    cabecalhos = lista[0].dados.keys()
    tamanhos = {cabecalho: self.encontrarTamanho(lista, cabecalho) for cabecalho in
                cabecalhos}
    cabecalhos_formatados = ' | ' + ' | '.join(f'{cabecalho:{tamanhos[cabecalho]}}'
                                                for cabecalho in cabecalhos) + ' | '
    qtdCaracteres = sum([tamanhos[nome]+3 for nome in cabecalhos])-1
    resultado = []
    resultado.append(f'{negrito}{azul} ┌{"—"*qtdCaracteres}┐ \n{
                        cabecalhos_formatados}\n└{"—"*qtdCaracteres}┘ {reset}')
    for ocorrencia in lista:
        campos = ' | ' + ' | '.join(f'{str(ocorrencia.dados.get(nome,""))}:{
                        tamanhos[nome]}}' for nome in cabecalhos) + ' | '
        resultado.append(campos)
    resultado.append(f'└{"—"*qtdCaracteres}┘ ')
    return '\n'.join(resultado)
```

◆ crud.py

● Importação de Módulos

1. **Biblioteca pandas:**
Biblioteca utilizada para manipular os arquivos como **.CSV**.
2. **Biblioteca fastapi:**
Biblioteca utilizada para construir a **API** com **Python**.
3. **Módulo [conexao_db](#):**
Arquivo que intermedia a [conexão](#) entre o **Python** e o **SQLite3**.
4. **Módulo [modelos](#):**
Arquivo que ajuda a **estruturar** o código de maneira **modular**.

Importação de Módulos

```
from api.conexao_db import *  
from api.modelos import *  
import pandas as pd  
from fastapi import FastAPI
```

● Gerenciando Conexão

1. **app:**
Esta **variável** representa uma **instância** com o **fastapi**.
2. **conn:**
Esta **variável** representa o **objeto de conexão** com o **banco de dados**.
3. **cursor:**
Esta variável representa o **objeto** utilizado para **manipular comandos** de dados **SQL**.

Gerenciando Conexão

```
app = FastAPI()  
conn = conectar()  
cursor = conn.cursor()
```

● Função lerDadosCsv()

Esta função **lê** os dados dos arquivos **.CSV armazenados** na pasta **tabelas**:

```
def LerDadosCsv(arquivo):
```



Este trecho representa uma **expressão** que **lê os arquivos .CSV** — formatados em **UTF-8** — com o **pandas** a partir de sua **função read_csv()**, e os **armazena** em um **dicionário** com a **função to_dict()**:

```
dados = pd.read_csv(f'api/tabelas/{arquivo}',  
encoding='UTF-8').to_dict(orient='records')
```

Após os dados serem **armazenados**, a função irá **retorná-los**:

```
return dados
```

Função *LerDadosCsv()*

```
def LerDadosCsv(arquivo):  
    dados = pd.read_csv(f'api/tabelas/{arquivo}',  
                        encoding='UTF-8').to_dict(orient='records')  
  
    return dados
```

- **Função consultar_tabelas()**

Esta função realiza **consultas SQL** a partir da **query** passada como **parâmetro**:

```
def consultar_tabelas(comando, arquivo=None):
```

Este bloco é utilizado para tratar na **captura de erros** caso ocorram:

```
try:  
    pass  
except Exception as ex:  
    print(ex)
```

Estas variáveis são **inicializadas** como uma **lista**:

```
lista, armazenamento = [], []
```

Este trecho **executa** a **query** que foi passada como **parâmetro**:

```
cursor.execute(comando)
```

Esta variável **armazena** os **resultados encontrados** do banco de dados:

```
informacoes = cursor.fetchall()
```



Esta variável tem uma “**list comprehension**” que **busca os nomes das colunas** e as **armazena** em formato de **lista**

```
nomeColunas = [nome[0] for nome in cursor.description]
```

Um **loop for** é **inicializado** para passar por cada **dado** dentro de **informações**:

```
for informacao in informacoes:
```

Cada dado será passado como um **dicionário** — **nomeColunas** e **informacao** foram **combinados** com **zip()** — e adicionado na variável **lista** que **contem a classe dados**.

```
dados = {coluna: resultado for coluna, resultado in
zip(nomeColunas, informacao)}
lista.append(Metadado(**dados))
```

Caso o parâmetro de **arquivo** seja **válido**, os dados serão passados para **armazenamento** e a função **criarEndpoint()** será **chamada**:

```
if arquivo:
    armazenamento.append(dados)
    criarEndpoint(armazenamento, arquivo)
```

Após os blocos acima serem **executados**, a função irá **retornar** a resposta da **classe Metadado** com a função **formatar()** na variável **lista**:

```
return Metadado().formatar(lista) + '\n'
```

Função consultar_tabelas()

```
def consultar_tabelas(comando, arquivo=None):
    lista, armazenamento = [], []
    cursor.execute(comando)
    informacoes = cursor.fetchall()
    nomeColunas = [nome[0] for nome in cursor.description]
    try:
        for informacao in informacoes:
            dados = {coluna: resultado for coluna, resultado in zip(nomeColunas,
informacao)}
            lista.append(Metadado(**dados))
            if arquivo:
                armazenamento.append(dados)
                criarEndpoint(armazenamento, arquivo)
    except Exception as ex:
        print(ex)
    return Metadado().formatar(lista) + '\n'
```


- **Função criarEndpoint()**

Esta função **cria endpoints** da **API** a partir de uma **lista** e o **nome** do **endpoint**.

```
def criarEndpoint(armazenamento, arquivo):
```

Este trecho **inicia** o **endpoint** com o **nome** passado.

```
@app.get(f'/{arquivo}')
```

Neste trecho **uma função assíncrona** é inicializada:

```
async def get_consultas():
```

Após os dados serem **armazenados**, a função irá **retorná-los** como **resposta**:

```
return armazenamento
```

Função `criarEndpoint()`

```
def criarEndpoint(armazenamento, arquivo):  
    @app.get(f'/{arquivo}')    async def get_consultas():  
        return armazenamento
```

- **Função criar_tabelas()**

Esta função **cria tabelas SQL** a partir da **query** passada como **parâmetro**:

```
def criar_tabelas(comando):
```

Este bloco é utilizado para **tratar na captura de erros** caso ocorram:

```
    try:
        pass
    except Exception as ex:
        print(ex)
```

Este trecho **executa** a **query** que foi passada como **parâmetro**:

```
    cursor.execute(comando)
```

Função criar_tabelas()

```
def criar_tabelas(comando):
    try:
        cursor.execute(comando)

    except Exception as ex:
        print(ex)
```

- **Função criar_insert()**

Esta função **cria INSERTs** nas **tabelas SQL** a partir da **query** passada como **parâmetro** e seus devidos **valores**:

```
def criar_insert(comando, valores):
```

Este bloco é utilizado para **tratar na captura de erros** caso ocorram:

```
    try:
        pass
    except Exception as ex:
        print(ex)
```

Este trecho **executa** a **query** que foi passada como parâmetro **múltiplas vezes**:

```
    cursor.executemany(comando, valores)
```

Função `criar_insert()`

```
def criar_insert(comando, valores):  
    try:  
        cursor.executemany(comando, valores)  
  
    except Exception as ex:  
        print(ex)
```

- **Execução do fastapi**

Neste trecho executa um **aplicativo fastapi** na porta **8000**.

1. **Biblioteca uvicorn:**

Biblioteca utilizada para iniciar um servidor.

```
if __name__ == '__main__':  
    import uvicorn  
    uvicorn.app(app, host='localhost', port=8000)
```

◆ criacao_tabelas.py

• Importação de Módulo

1. Módulo [crud](#):

Arquivo que é responsável por executar **comandos SQL**.

Importação de Módulo

```
from crud import *
```

• Função executar_tabelas()

Esta função é responsável pela **execução das tabelas SQL**:

função executar_tabelas()

```
def executar_tabelas():
```

Este trecho **desativa** as **chaves estrangeiras** — caso estejam **ativadas** — para não gerar **conflitos** entre as **tabelas** a serem **criadas**:

Desativação de Chaves Estrangeiras

```
cursor.execute("PRAGMA foreign_keys = OFF")
```

Este bloco manda a **query** que cria a tabela [tb_Cargo](#) como **parâmetro** da função **criar_tabelas()**:

Criação da tabela tb_Cargo

```
criar_tabelas('''
    CREATE TABLE IF NOT EXISTS tb_Cargo (

        idCargo      INTEGER PRIMARY KEY AUTOINCREMENT,
        nomeCargo     VARCHAR(30),
        salarioBase   REAL,
        nivelCargo    VARCHAR(30),
        escala        VARCHAR(30)

    );
''')
```



Este bloco manda a **query** que cria a tabela [tb_Departamento](#) como **parâmetro** da função `criar_tabelas()`:

Criação da tabela tb_Departamento

```
criar_tabelas("""
    CREATE TABLE IF NOT EXISTS tb_Departamento (

        idDepartamento      INTEGER PRIMARY KEY AUTOINCREMENT,
        nomeDepartamento    VARCHAR(30),
        idGerente             INTEGER,
        andarDepartamento   INTEGER,
        horarioFuncionamento VARCHAR(30),

        FOREIGN KEY (idGerente) REFERENCES tb_Funcionario(idFuncionario)
    );
""")
```

Este bloco manda a **query** que cria a tabela [tb_Funcionario](#) como **parâmetro** da função `criar_tabelas()`:

Criação da tabela tb_Funcionario

```
criar_tabelas('''
    CREATE TABLE IF NOT EXISTS tb_Funcionario (

        idFuncionario      INTEGER PRIMARY KEY AUTOINCREMENT,
        nomeFuncionario    VARCHAR(50),
        idCargo             INTEGER,
        idDepartamento     INTEGER,
        salarioReal         REAL,
        dataNascimento      TEXT,

        FOREIGN KEY (idCargo) REFERENCES tb_Cargo(idCargo),
        FOREIGN KEY (idDepartamento) REFERENCES tb_Departamento(idDepartamento)
    );
''')
```



Este bloco manda a **query** que cria a tabela [tb_historicoSalarial](#) como **parâmetro** da função **criar_tabelas()**:

Criação da tabela tb_historicoSalarial

```
criar_tabelas('''
    CREATE TABLE IF NOT EXISTS tb_historicoSalarial (

        idSalarial      INTEGER PRIMARY KEY AUTOINCREMENT,
        idFuncionario   INTEGER,
        data            TEXT,
        salario         REAL,

        FOREIGN KEY (idFuncionario) REFERENCES tb_Funcionario(idFuncionario)
    );
''')
```

Este bloco manda a **query** que cria a tabela [tb_Dependente](#) como **parâmetro** da função **criar_tabelas()**:

Criação da tabela tb_Dependente

```
criar_tabelas('''
    CREATE TABLE IF NOT EXISTS tb_Dependente (

        idDependente   INTEGER PRIMARY KEY AUTOINCREMENT,
        idFuncionario   INTEGER,
        parentesco     VARCHAR(10),
        nome           VARCHAR(50),
        idade          TEXT,
        genero         VARCHAR(15),

        FOREIGN KEY (idFuncionario) REFERENCES tb_Funcionario(idFuncionario)
    );
''')
```



Este bloco manda a **query** que cria a tabela [tb_Projeto](#) como **parâmetro** da função **criar_tabelas()**:

Criação da tabela tb_Projeto

```
criar_tabelas(''  
    CREATE TABLE IF NOT EXISTS tb_Projeto (  
  
        idProjeto          INTEGER PRIMARY KEY AUTOINCREMENT,  
        idResponsavel      INTEGER,  
        nomeProjeto        VARCHAR(50),  
        descricaoProjeto   VARCHAR(150),  
        inicio             TEXT,  
        conclusao          TEXT,  
        custo              REAL(10,2),  
        status             VARCHAR(15),  
  
        FOREIGN KEY (idResponsavel) REFERENCES  
tb_Funcionario(idFuncionario)  
    );  
''')
```

Este bloco manda a **query** que cria a tabela [tb_Recurso](#) como **parâmetro** da função **criar_tabelas()**:

Criação da tabela tb_Recurso

```
criar_tabelas(''  
    CREATE TABLE IF NOT EXISTS tb_Recurso (  
  
        idRecurso          INTEGER PRIMARY KEY AUTOINCREMENT,  
        idProjeto          INTEGER,  
        descricaoRecurso   VARCHAR(40),  
        tipoRecurso        VARCHAR(15),  
        qtdRecurso         INTEGER,  
        dataUtilizacao     TEXT,  
  
        FOREIGN KEY (idProjeto) REFERENCES tb_Projeto(idProjeto)  
    );  
''')
```

- **Leitura de Dados**

Este trecho **lê todas as linhas** de cada um dos arquivos **.CSV**:

Lista armazenada	.CSV
funcionarios	tb_funcionario.csv
cargos	tb_cargo.csv
departamentos	tb_departamento.csv
salarios	tb_historicoSalarial.csv
dependentes	tb_dependente.csv
projetos	tb_projeto.csv
recursos	tb_recurso.csv

Leitura de dados

```
funcionarios = lerDadosCsv('tb_funcionario.csv')
cargos = lerDadosCsv('tb_cargo.csv')
departamentos = lerDadosCsv('tb_departamento.csv')
salarios = lerDadosCsv('tb_historicoSalarial.csv')
dependentes = lerDadosCsv('tb_dependente.csv')
projetos = lerDadosCsv('tb_projeto.csv')
recursos = lerDadosCsv('tb_recurso.csv')
```

- **Inserção de Dados**

Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **cargos** na tabela **tb_cargo** que foi **criada anteriormente**:

Inserção de dados na tabela tb_cargo

```
criar_insert('INSERT IGNORE INTO tb_Cargo(idCargo, nomeCargo,
salarioBase, nivelCargo, escala) VALUES(%(idCargo)s, %(nomeCargo)s,
%(salarioBase)s, %(nivelCargo)s, %(escala)s);',
            [{'idCargo':cargo['idCargo'],
              'nomeCargo':cargo['nomeCargo'],
              'salarioBase':cargo['salarioBase'],
              'nivelCargo':cargo['nivelCargo'],
              'escala':cargo['escala']} for cargo in cargos])
```




Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **departamentos** na tabela **tb_Departamento** que foi **criada anteriormente**:

Inserção de dados na tabela tb_Departamento

```
criar_insert(''INSERT IGNORE INTO
tb_Departamento(idDepartamento, nomeDepartamento, idGerente,
andarDepartamento, horarioFuncionamento)
VALUES(%(idDepartamento)s, %(nomeDepartamento)s, %(idGerente)s,
%(andarDepartamento)s, %(horarioFuncionamento)s);'',
[{'idDepartamento':departamento['idDepartamento'],
'nomeDepartamento':departamento['nomeDepartamento'],
'idGerente':departamento['idGerente'],
'andarDepartamento':departamento['andarDepartamento'],
'horarioFuncionamento':departamento['horarioFuncionamento']}
for departamento in departamentos])
```

Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **funcionarios** na tabela **tb_Funcionario** que foi **criada anteriormente**:

Inserção de dados na tabela tb_Funcionario

```
criar_insert('INSERT IGNORE INTO tb_Funcionario(idFuncionario,
nomeFuncionario, idCargo, idDepartamento, salarioReal,
dataNascimento) VALUES(%(idFuncionario)s, %(nomeFuncionario)s,
%(idCargo)s, %(idDepartamento)s, %(salarioReal)s,
%(dataNascimento)s);',
[{'idFuncionario':funcionario['idFuncionario'],
'nomeFuncionario':funcionario['nomeFuncionario'],
'idCargo':funcionario['idCargo'],
'idDepartamento':funcionario['idDepartamento'],
'salarioReal':funcionario['salarioReal'],
'dataNascimento':funcionario['dataNascimento']}
for funcionario in funcionarios])
```



Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **salarios** na tabela **tb_historicoSalarial** que foi **criada anteriormente**:

Inserção de dados na tabela tb_historicoSalarial

```
criar_insert('''INSERT IGNORE INTO
tb_historicoSalarial(idSalarial, idFuncionario, data, salario)
VALUES(%(idSalarial)s, %(idFuncionario)s, %(data)s,
%(salario)s);''',
            [{ 'idSalarial': historicoSalarial['idSalarial'],
              'idFuncionario': historicoSalarial['idFuncionario'],
              'data': historicoSalarial['data'],
              'salario': historicoSalarial['salario'] } for
historicoSalarial in salarios])
```

Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **dependentes** na tabela **tb_Dependente** que foi **criada anteriormente**:

Inserção de dados na tabela tb_Dependente

```
criar_insert('''INSERT IGNORE INTO tb_Dependente(idDependente,
idFuncionario, parentesco, nome, idade, genero)
VALUES(%(idDependente)s, %(idFuncionario)s, %(parentesco)s,
%(nome)s, %(idade)s, %(genero)s);''',
            [{ 'idDependente': dependente['idDependente'],
              'idFuncionario': dependente['idFuncionario'],
              'parentesco': dependente['parentesco'],
              'nome': dependente['nome'],
              'idade': dependente['idade'],
              'genero': dependente['genero'] } for dependente in
dependentes])
```



Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **projetos** na tabela **tb_Projeto** que foi **criada anteriormente**:

Inserção de dados na tabela tb_Projeto

```
criar_insert('''INSERT OR IGNORE INTO tb_Projeto(idProjeto,
idResponsavel, nomeProjeto, descricaoProjeto, inicio, conclusao, custo,
status)
VALUES(:idProjeto, :idResponsavel, :nomeProjeto, :descricaoProjeto,
:inicio, :conclusao, :custo, :status);''',
[{'idProjeto':projeto['idProjeto'],
'idResponsavel':projeto['idResponsavel'],
'nomeProjeto':projeto['nomeProjeto'],
'descricaoProjeto':projeto['descricaoProjeto'],
'inicio':projeto['inicio'],
'conclusao':projeto['conclusao'],
'custo':projeto['custo'],
'status':projeto['status']} for projeto in projetos])
```

Este bloco manda a **query** que adiciona todos os dados **armazenados** na lista **recursos** na tabela **tb_Recurso** que foi **criada anteriormente**:

Inserção de dados na tabela tb_Recurso

```
criar_insert('''INSERT OR IGNORE INTO tb_Recurso(idRecurso,
idProjeto, descricaoRecurso, tipoRecurso, qtdRecurso, dataUtilizacao)
VALUES(:idRecurso, :idProjeto, :descricaoRecurso, :tipoRecurso,
:qtdRecurso, :dataUtilizacao);''',
[{'idRecurso':recurso['idRecurso'],
'idProjeto':recurso['idProjeto'],
'descricaoRecurso':recurso['descricaoRecurso'],
'tipoRecurso':recurso['tipoRecurso'],
'qtdRecurso':recurso['qtdRecurso'],
'dataUtilizacao':recurso['dataUtilizacao']} for recurso in recursos])
```

Este trecho **ativa** as **chaves estrangeiras** para **garantir** a **integridade** dos **dados**:

```
cursor.execute("PRAGMA foreign_keys = ON")
```

◆ comandos.py

- Importação de Módulos

1. Módulo [crud](#):

Arquivo que é responsável por **executar comandos SQL**.

2. Módulo [criacao_tabelas](#):

Arquivo que é responsável pela **criação das tabelas SQL**.

Importação de Módulos

```
from api.crud import *  
from api.criacao_tabelas import executar_tabelas
```

- Chamada da Função executar_tabelas()

Função [executar_tabelas\(\)](#) foi chamada.

Sequências de Escape ANSI

```
executar_tabelas()
```

- Sequências de Escape ANSI

Esta **variável** armazena **sequências de escape ANSI** para formatar a **cor** das **strings** como amarelo.

Sequências de Escape ANSI

```
amarelo = '\x1b[38;5;229m'
```



● TP4.1

Este trecho representa uma **consulta SQL** para saber a **média por departamento** dos **funcionários** responsáveis por projetos **concluídos**. Para realizar a consulta foi necessário selecionar o dados:

Tabela	Dados
tb_funcionario	nomeFuncionario
tb_HistoricoSalarial	salario, data
tb_Projeto	status
tb_departamento	nomeDepartamento

Trazer a média dos salários (atual) dos funcionários responsáveis por projetos concluídos, agrupados por departamento.

```
print(f'{amarelo}1. Trazer a média dos salários (atual) dos funcionários responsáveis por projetos concluídos, agrupados por departamento.')
```

```
tp4_1 = consultar_tabelas("""
    SELECT D.nomeDepartamento AS Departamento,
           ROUND(AVG(H.salario)) AS Media
    FROM tb_Funcionario F
    JOIN (
        SELECT idFuncionario, salario
        FROM tb_HistoricoSalarial H
        WHERE H.data = (
            SELECT MAX(data)
            FROM tb_HistoricoSalarial
            WHERE idFuncionario = H.idFuncionario
        )
    ) H ON H.idFuncionario = F.idFuncionario
    JOIN tb_Projeto P ON P.idResponsavel = F.idFuncionario
    JOIN tb_Departamento D ON D.idDepartamento = F.idDepartamento
    WHERE P.status = 'Concluído'
    GROUP BY D.nomeDepartamento;
""", 'tp4.1')
```

```
print(tp4_1)
```

Printscreen do resultado da consulta

```
PS C:\Users\Leticia\Downloads\TP4- PB> python -m uvicorn api.comandos:app --reload
INFO: Will watch for changes in these directories: ['C:\\Users\\Leticia\\Downloads\\TP4- PB']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [51732] using StatReload
```

1. Trazer a média dos salários (atual) dos funcionários responsáveis por projetos concluídos, agrupados por departamento.

Departamento	Media
Financeiro	12000.0
Marketing	5000.0
TI	21500.0



- TP4.2

Este trecho representa uma **consulta SQL** para saber quais são os **três recursos** mais **usados** nos **projetos**. Para realizar a consulta foi necessário selecionar o dados:

Tabela	Dados
tb_recurso	descricaoRecurso, qtdRecurso

Identificar os três recursos materiais mais usados nos projetos, listando a descrição do recurso e a quantidade total usada.

```
print(f'{amarelo}2. Identificar os três recursos materiais mais usados  
nos projetos, listando a descrição do recurso e a quantidade total  
usada.')
```

```
tp4_2 = consultar_tabelas("""  
    SELECT descricaoRecurso AS Recurso,  
    SUM(qtdRecurso)          AS Qtd  
    FROM tb_Recurso  
    GROUP BY descricaoRecurso  
    ORDER BY qtd DESC LIMIT 3;  
    """)  
print(tp4_2)
```

Printscreen do resultado da consulta

2. Identificar os três recursos materiais mais usados nos projetos, listando a descrição do recurso e a quantidade total usada.

Recurso	Qtd
Desenvolvedor de Software	16
Assinatura Power BI Premium	4
Servidor de Hospedagem	3



- TP4.3

Este trecho representa uma **consulta SQL** para saber o **custo total** de **projetos concluídos** por **departamento**. Para realizar a consulta foi necessário selecionar o dados:

Tabela	Dados
tb_funcionario	intermediário para junção dos dados
tb_Projeto	status, custo
tb_departamento	nomeDepartamento

Calcular o custo total dos projetos por departamento, considerando apenas os projetos "Concluídos".

```
print(f'{amarelo}3. Calcular o custo total dos projetos por
      departamento, considerando apenas os projetos "Concluídos".')
tp4_3 = consultar_tabelas("""
    SELECT D.nomeDepartamento AS Departamento,
           SUM(P.custo)          AS CustoTotal
    FROM   tb_Funcionario  F
    JOIN   tb_Departamento D ON D.idDepartamento = F.idDepartamento
    JOIN   tb_Projeto      P ON P.idResponsavel = F.idFuncionario
    WHERE  P.status = 'Concluído'
    GROUP BY D.nomeDepartamento;
    """, 'tp4.3')
print(tp4_3)
```

Printscreen do resultado da consulta

3. Calcular o custo total dos projetos por departamento, considerando apenas os projetos "Concluídos".

Departamento	CustoTotal
Financeiro	600.0
Marketing	450.0
TI	3600.0

● TP4.4

Este trecho representa uma **consulta SQL** para saber os **projetos** que estão em **execução**. Para realizar a consulta foi necessário selecionar o dados:

Tabela	Dados
tb_funcionario	nomeFuncionario
tb_Projeto	nomeProjeto, custo, inicio, conclusao, status

Listar todos os projetos com seus respectivos nomes, custo, data de início, data de conclusão e o nome do funcionário responsável, que estejam "Em Execução".

```
print(f'{amarelo}4. Listar todos os projetos com seus respectivos
      nomes, custo, data de início, data de conclusão e o nome do
      funcionário responsável, que estejam "Em Execução".')
tp4_4 = consultar_tabelas("""
    SELECT P.nomeProjeto AS Projeto,
           P.custo,
           P.inicio,
           P.conclusao,
           F.nomeFuncionario AS Funcionario,
           P.status
    FROM tb_Funcionario F
    JOIN tb_Projeto P ON P.idResponsavel = F.idFuncionario
    WHERE P.status = 'Em Execução';
    """,'tp4.4')
print(tp4_4)
```

Printscreen do resultado da consulta

4. Listar todos os projetos com seus respectivos nomes, custo, data de início, data de conclusão e o nome do funcionário responsável, que estejam "Em Execução".

Projeto	custo	inicio	conclusao	Funcionario	status
E-commerce de Eletrônicos [Site Institucional]	8000.0	2025-02-10	2025-04-10	André Xavier	Em Execução
E-commerce de Roupas [Site Institucional]	8000.0	2025-02-15	2025-04-15	Jéssica Borges	Em Execução
Publicidade Paga II	500.0	2025-02-13	2025-02-23	Laís Gomes	Em Execução
Portfolio Garçon [Single page]	3000.0	2025-02-22	2025-03-01	André Xavier	Em Execução



- TP4.5

Este trecho representa uma **consulta SQL** para saber qual foi o **projeto** que teve o **maior número** de **dependentes**. Para realizar a consulta foi necessário selecionar o dados:

Tabela	Dados
tb_funcionario	nomeFuncionario
tb_Projeto	nomeProjeto
tb_Dependente	idDependente utilizado para contagem

Listar qual estagiário possui filho

```
print(f'{amarelo}5. Identificar o projeto com o maior número de
dependentes envolvidos, considerando que os dependentes são
associados aos funcionários que estão gerenciando os projetos.')
tp4_5 = consultar_tabelas("""
SELECT P.nomeProjeto AS Projeto,
COUNT(D.idDependente) AS Qtd
FROM tb_Funcionario F
JOIN tb_Projeto P ON P.idResponsavel = F.idFuncionario
JOIN tb_Dependente D ON D.idFuncionario = F.idFuncionario
GROUP BY P.nomeProjeto
ORDER BY Qtd DESC LIMIT 1;
""")
print(tp4_5)
```

Printscreen do resultado da consulta

5. Identificar o projeto com o maior número de dependentes envolvidos, considerando que os dependentes são associados aos funcionários que estão gerenciando os projetos.

Projeto	Qtd
Restaurante [Single page]	3

- Desconectar Base de Dados

Após todo o **código** ser **executado** o banco tem sua **conexão encerrada**.

Encerramento de conexão com MySQL

```
desconectar(conn)
```



◆ requirements.txt

- dependências

Este arquivo representa as dependências a serem instaladas.

requirements.txt

```
fastapi
pandas
uvicorn
```

◆ render.yaml

- services

Este arquivo representa as **configurações** para fazer o **deploy** no render.com, com o tipo para **web**, o nome para ser **TP4-PB**, o ambiente como **python** do plano gratuito, sem deploy automático, com o arquivo **requirements.txt** para instalar as **dependências** listadas e o comando inicial para rodar a **API**,

render.yaml

```
services:
  - type: web
    name: TP4-PB
    runtime: python
    plan: free
    autoDeploy: false
    buildCommand: pip install -r requirements.txt
    startCommand: uvicorn api.comandos:app --host 0.0.0.0 --port
    $PORT
```

Deploy

◆ Github

Após criar um **repositório no GitHub**, foram feitos **commits da API** para o **repositório**.

```
Comandos GitHub

• PS C:\Users\Letícia\Downloads\TP4- PB> git init
Reinitialized existing Git repository in C:/Users/Letícia/Downloads/TP4- PB/.git/
• PS C:\Users\Letícia\Downloads\TP4- PB> git add .
• PS C:\Users\Letícia\Downloads\TP4- PB> git commit -m "Projeto de Bloco - TP4"
[main 87cf026] Projeto de Bloco - TP4
 1 file changed, 1 insertion(+), 1 deletion(-)
• PS C:\Users\Letícia\Downloads\TP4- PB> git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 394 bytes | 394.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/LeticiaFAAGomes/PB_TP4.git
 92e1d75..87cf026  main -> main
```


Após os **commits serem executados**, o repositório ficou **completo**.



- **Render**

Com o [render](#) foi criada a **configuração** para o **deploy**, adicionando o nome do **projeto** e o **comando inicial** que executa o projeto

Deploy no Render

 Leticia's workspace

New Web Service

Search K

+ New

?

⌵

You are deploying a Web Service

You seem to be using **Python**, so we've autofilled some fields accordingly. Make sure the values look right to you!

Source Code

LeticiaFAAGomes / PB_TP4 • 5m ago

Edit


Name

A unique name for your web service.

PB_TP4

Project Optional

Add this web service to a project once it's created.



Create a new project to add this to?
You don't have any projects in this workspace. Projects allow you to group resources into environments so you can better manage related resources.

+ Create a project

Language

Python 3


Branch

The Git branch to build and deploy.

main

Region

Your services in the same region can communicate over a private network. You currently have services running in **Oregon**.

 Oregon (US West)

1 existing service

Deploy in a new region +

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.

e.g. src

Build Command

Render runs this command to build your app before each deploy.

\$ pip install -r requirements.txt

Start Command

Render runs this command to start your app with each deploy.

\$ uvicorn api.comandos:app --host 0.0.0.0 --port \$PORT



Após as **configurações** serem feitas, o **deploy** foi **concluído** com **sucesso**

Deploy feito com Sucesso

Letícia's workspace ▾ PB_TP4 ▾ 🔍 Search 🔍 + New ⓘ ⌚

← Dashboard

PB_TP4

Python3 Free Upgrade your instance →

LetíciaFAAGomes / PB_TP4 ↗ main
https://pb-tp4.onrender.com 📄

ⓘ Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

February 22, 2025 at 6:39 PM Live
92e1d75 Update README.md

All logs ▾ 🔍 Search ⚙ Live tail ▾ GMT-3 ⬆ ⬇ ⬆

Feb 22 06:41:01 PM | André Xavier | Em Execução | 3000.0 0.000 22-20-0702 2025-02-22

Feb 22 06:41:01 PM | 5. Identificar o projeto com o maior número de dependentes envolvidos, considerando que os dependentes são associados aos funcionários que estão gerenciando os projetos.

Feb 22 06:41:01 PM | Projeto | Qtd |

Feb 22 06:41:01 PM | Restaurante [Single page] | 3 |

Feb 22 06:41:01 PM | Banco Desconectado.

Feb 22 06:41:01 PM | INFO: 127.0.0.1:57498 - "HEAD / HTTP/1.1" 404 Not Found

Feb 22 06:41:05 PM | ➡ Your service is live 🎉

Feb 22 06:41:06 PM | INFO: 34.82.88.145:0 - "GET / HTTP/1.1" 404 Not Found

Changelog ^
Invite a friend
Contact support

Os **JSONs** estão foram lançados.

TP4.1

← ↻ 🔍 https://pb-tp4.onrender.com/tp4.1

Estilos de formatação

```
{
  "Departamento": "Financeiro",
  "Media": 12000
},
{
  "Departamento": "Marketing",
  "Media": 5000
},
{
  "Departamento": "TI",
  "Media": 21500
}
```

TP4.3

← ↻ 🏠 <https://pb-tp4.onrender.com/tp4.3>

Estilos de formatação ☒

```
[
  {
    "Departamento": "Financeiro",
    "CustoTotal": 600
  },
  {
    "Departamento": "Marketing",
    "CustoTotal": 450
  },
  {
    "Departamento": "TI",
    "CustoTotal": 3600
  }
]
```

TP4.4

← ↻ 🏠 <https://pb-tp4.onrender.com/tp4.4>

Estilos de formatação ☒


```
[
  {
    "Projeto": "E-commerce de Eletrônicos [Site Institucional]",
    "custo": 8000,
    "inicio": "2025-02-10",
    "conclusao": "2025-04-10",
    "Funcionario": "André Xavier",
    "status": "Em Execução"
  },
  {
    "Projeto": "E-commerce de Roupas [Site Institucional]",
    "custo": 8000,
    "inicio": "2025-02-15",
    "conclusao": "2025-04-15",
    "Funcionario": "Jéssica Borges",
    "status": "Em Execução"
  },
  {
    "Projeto": "Publicidade Paga II",
    "custo": 500,
    "inicio": "2025-02-13",
    "conclusao": "2025-02-23",
    "Funcionario": "Lais Gomes",
    "status": "Em Execução"
  },
  {
    "Projeto": "Portfolio Garçon [Single page]",
    "custo": 3000,
    "inicio": "2025-02-22",
    "conclusao": "2025-03-01",
    "Funcionario": "André Xavier",
    "status": "Em Execução"
  }
]
```

Looker Studio

◆ Conectando TPs

A **conexão** dos **dados** foi feita com o [link do render](#).

Conectando TP4.1 ao Looker Studio


 **JSON**
Por Windsor.ai

This JSON connector helps you move data from any source to Looker Studio. It works with data from our attribution platform, a custom connector URL or any other data in JSON format. Windsor.ai powers all your reporting and enables you to track your complete funnel across PPC, analytics, SEO, social media, web analytics CRM and ecommerce. IMPORTANT INFORMATION: If you experience issues while installing this connector please make sure that you are logged in to only one Google account and try again.


Você tem a responsabilidade de ler e obedecer a todos os Termos de Serviço de terceiros aplicáveis.

[SAIBA MAIS](#) [INFORMAR UM PROBLEMA](#)

Fill out the form to connect to a JSON data source.


Enter the URL of a JSON data source 

<https://pb-tp4.onrender.com/tp4.1>

Root Element 

Clique em CONECTAR para continuar.

Conectando TP4.3 ao Looker Studio


 **JSON**
Por Windsor.ai

This JSON connector helps you move data from any source to Looker Studio. It works with data from our attribution platform, a custom connector URL or any other data in JSON format. Windsor.ai powers all your reporting and enables you to track your complete funnel across PPC, analytics, SEO, social media, web analytics CRM and ecommerce. IMPORTANT INFORMATION: If you experience issues while installing this connector please make sure that you are logged in to only one Google account and try again.

Você tem a responsabilidade de ler e obedecer a todos os Termos de Serviço de terceiros aplicáveis.

[SAIBA MAIS](#) [INFORMAR UM PROBLEMA](#)

Fill out the form to connect to a JSON data source.

Enter the URL of a JSON data source 

<https://pb-tp4.onrender.com/tp4.3>

Root Element 

Clique em CONECTAR para continuar.

Conectando TP4.4 ao Looker Studio



JSON

Por Windsor.ai

This JSON connector helps you move data from any source to Looker Studio. It works with data from our attribution platform, a custom connector URL or any other data in JSON format. Windsor.ai powers all your reporting and enables you to track your complete funnel across PPC, analytics, SEO, social media, web analytics CRM and ecommerce. IMPORTANT INFORMATION: If you experience issues while installing this connector please make sure that you are logged in to only one Google account and try again.

Você tem a responsabilidade de ler e obedecer a todos os Termos de Serviço de terceiros aplicáveis.

[SAIBA MAIS](#)

[INFORMAR UM PROBLEMA](#)

Fill out the form to connect to a JSON data source.

Enter the URL of a JSON data source ?

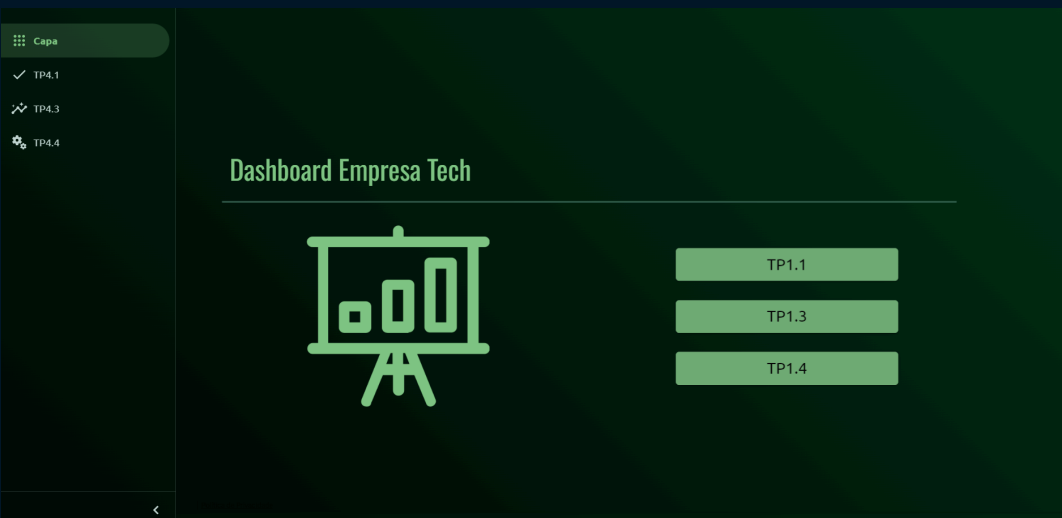
Root Element ?

Clique em CONECTAR para continuar.

- **Capa do Dashboard no Looker Studio**

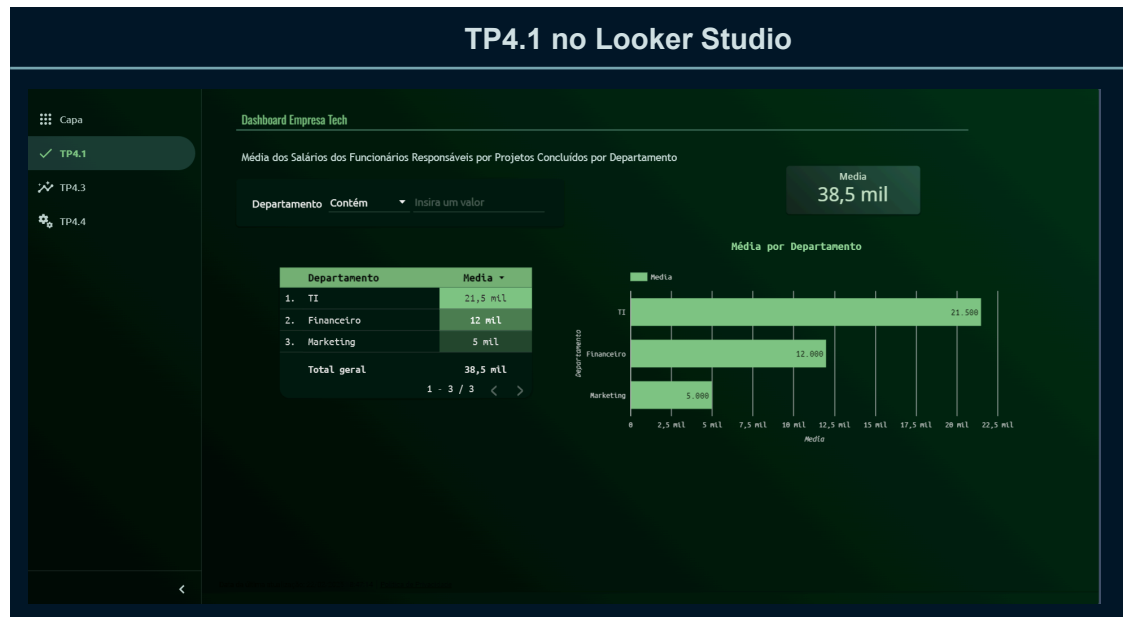
Capa do **dashboard** no **Looker Studio** como **navegação** para os **3 TPs** escolhidos.

Capa do Looker Studio



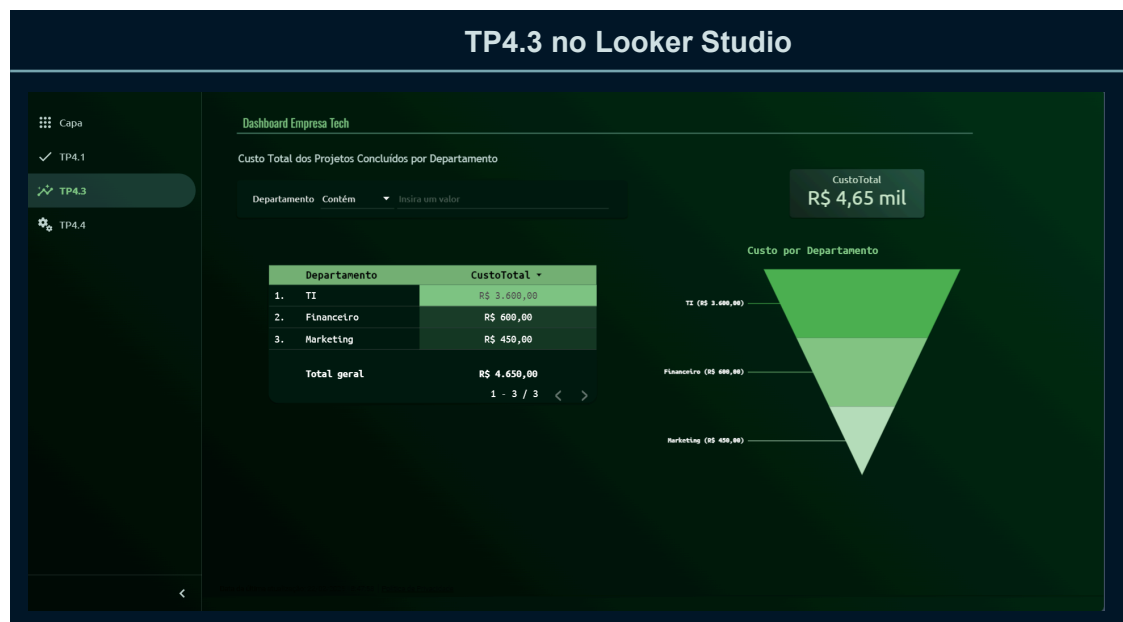
- **TP4.1 no Looker Studio**

TP4.1 no Looker Studio, com **filtro** por departamento, **tabela** e **gráficos**.



- **TP4.3 no Looker Studio**

TP4.3 no Looker Studio, com **filtro** por departamento, **tabela** e **gráficos**.



● TP4.4 no Looker Studio

TP4.4 no **Looker Studio**, com **filtro** por projeto e funcionário, **tabela** e **gráficos**.

