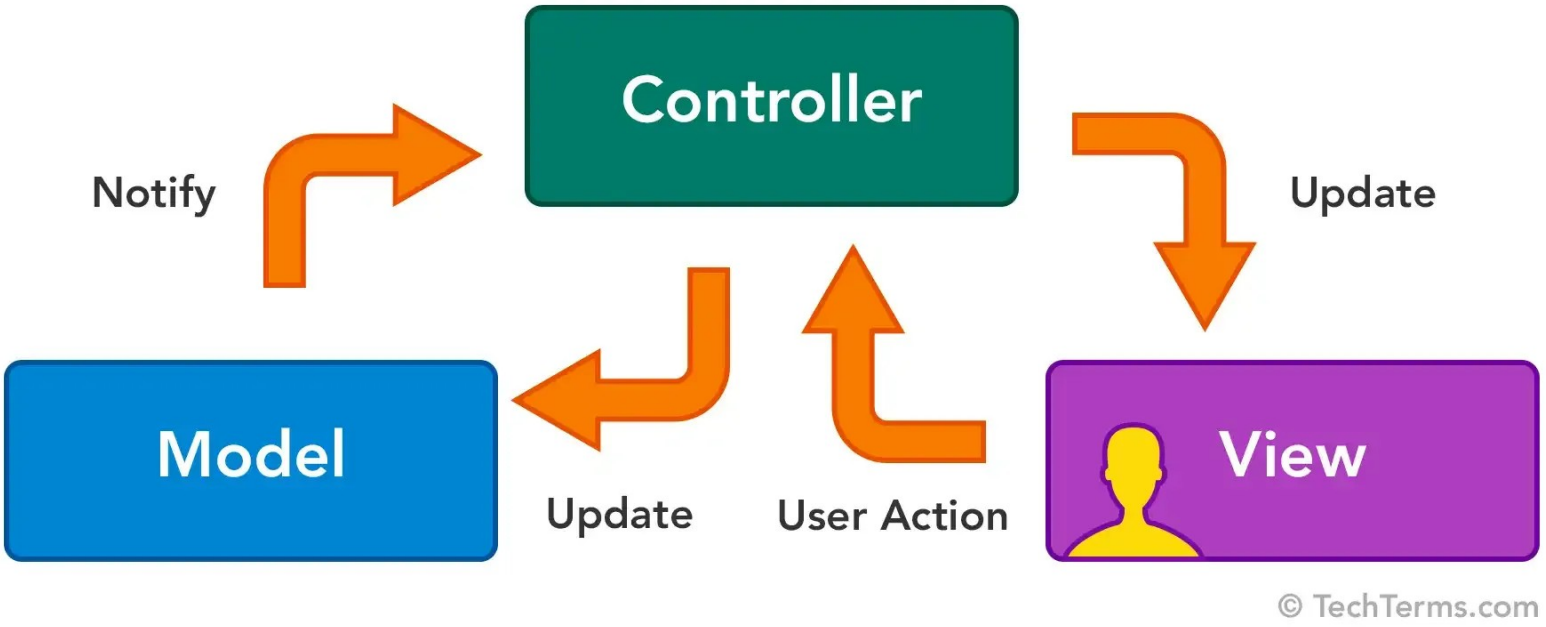
# Arquitetura MVC (Model-View-Controller)



# Arquitetura MVC (Model-View-Controller)

- O MVC é um padrão de arquitetura de software que divide uma aplicação em três camadas principais, cada uma com responsabilidades bem definidas:

| Camada | Responsabilidade principal | Interação |
|---|---|---|
| **Model (Modelo)** | Gerencia os **dados** e as **regras de negócio** | Fala com o banco de dados e o Controller |
| **View (Visão)** | Cuida da **interface com o usuário** | Recebe dados do Controller e os exibe |
| **Controller (Controlador)** | Atua como **intermediário** entre View e Model | Recebe requisições da View e usa o Model |

# Arquitetura MVC (Model-View-Controller)

- O **Model** representa o núcleo lógico da aplicação: onde estão as regras de negócio, a validação de dados, e a integração com o banco de dados (PostgreSQL, MySQL, SQLite, etc.);
- Funções principais do Model:
  - Definir as entidades (ex.: Usuário, Produto, Pedido);
  - Realizar operações CRUD (Create, Read, Update, Delete);
  - Aplicar regras de negócio (ex.: validar senha, calcular imposto).

# Arquitetura MVC (Model-View-Controller)

```
# model.py
import psycopg2

class UsuarioModel:
    def __init__(self):
        self.conexao = psycopg2.connect(
            host="localhost",
            database="sistema",
            user="postgres",
            password="senha"
        )

    def listar_usuarios(self):
        cursor = self.conexao.cursor()
        cursor.execute("SELECT id, nome,
email FROM usuarios")
        usuarios = cursor.fetchall()
        cursor.close()
        return usuarios

    def inserir_usuario(self, nome,
email):
        cursor = self.conexao.cursor()
        cursor.execute("INSERT INTO
usuarios (nome, email) VALUES
(%s, %s)", (nome, email))
        self.conexao.commit()
        cursor.close()
```

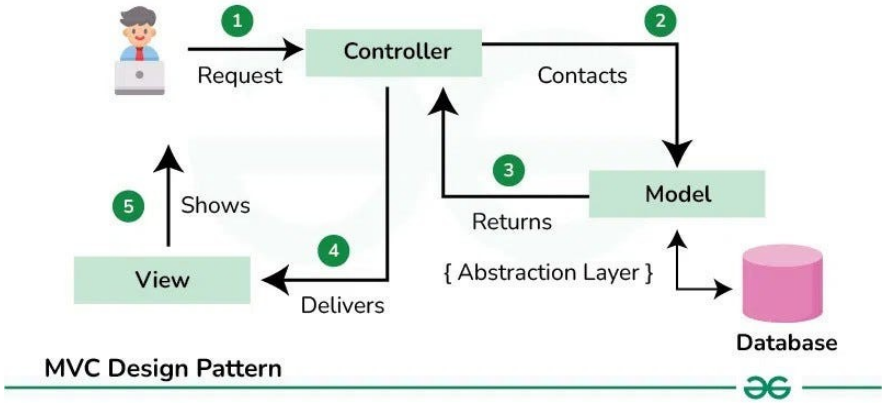
# Arquitetura MVC (Model-View-Controller)

- A **View** é responsável por exibir as informações para o usuário, seja:
  - na tela do terminal;
  - em uma página web (HTML);
  - ou em uma API (JSON).
- Ela não contém lógica de negócio, apenas formata os dados que recebe do Controller.

```
# view.py
def mostrar_usuarios(lista):
    print("=== Lista de Usuários ===")
    for usuario in lista:
        print(f"ID: {usuario[0]} | Nome: {usuario[1]} | Email: {usuario[2]}")
```

# Arquitetura MVC (Model-View-Controller)

- O **Controller** é o “cérebro da aplicação”, responsável por:
  - Receber comandos do usuário (ex.: cliques, requisições HTTP, entradas);
  - Coordenar as ações entre Model e View;
  - Decidir qual dado deve ser exibido e como.



# Arquitetura MVC (Model-View-Controller)

```
# controller.py
from model import UsuarioModel
from view import mostrar_usuarios

class UsuarioController:
    def __init__(self):
        self.model = UsuarioModel()

    def listar(self):
        usuarios = self.model.listar_usuarios()
        mostrar_usuarios(usuarios)

    def cadastrar(self, nome, email):
        self.model.inserir_usuario(nome, email)
        print("Usuário cadastrado com sucesso!")
```

# Arquitetura MVC (Model-View-Controller)

- Fluxo típico de execução:
  1) O usuário solicita uma ação (por exemplo, “listar usuários”);
  2) O Controller recebe o pedido e chama o método do Model;
  3) O Model busca os dados no banco de dados;
  4) O Controller envia os dados para a View;
  5) A View apresenta o resultado ao usuário.

```
# main.py
from controller import UsuarioController

if __name__ == "__main__":
    app = UsuarioController()
    app.listar()
```

# Arquitetura MVC (Model-View-Controller)

- Vantagens do MVC:
  - **Organização:** separa as responsabilidades, facilitando manutenção;
  - **Reutilização:** as Views podem mudar sem alterar o Model;
  - **Escalabilidade:** ideal para projetos grandes e em equipe;
  - **Testabilidade:** facilita a criação de testes unitários e de integração.

# Arquitetura MVC (Model-View-Controller)

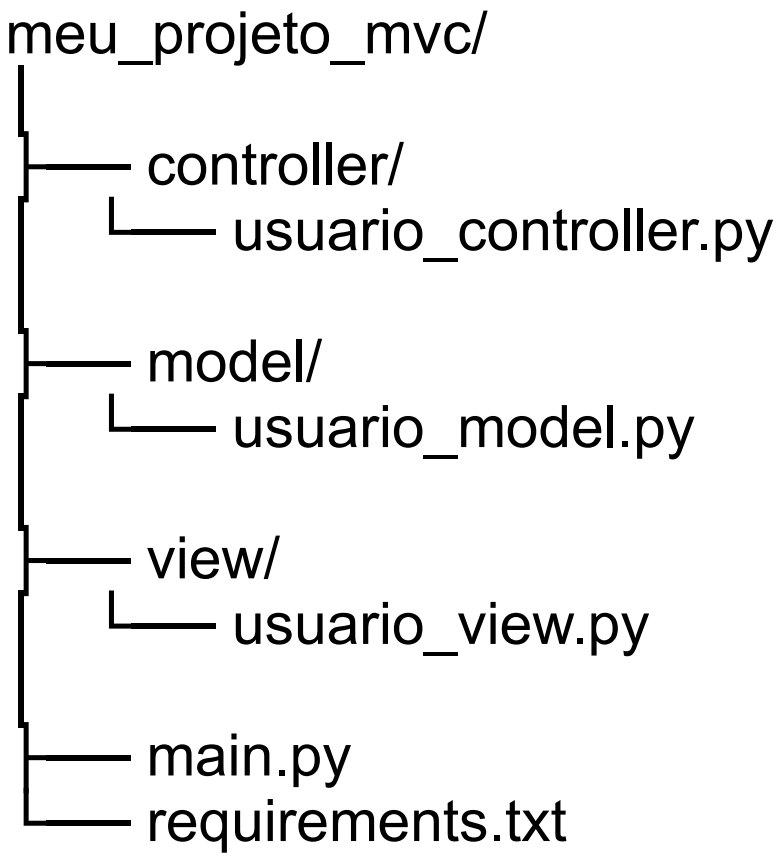
- Cuidados e boas práticas:
  - Não coloque lógica de negócio na View;
  - Evite que o Controller acesse diretamente o banco — use o Model;
  - Mantenha cada camada em arquivos separados (model.py, view.py, controller.py).

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Vamos montar um projeto completo em Python com arquitetura MVC, totalmente funcional, usando um menu no terminal e conexão ao PostgreSQL;
- Esse exemplo vai simular um pequeno cadastro de usuários com as operações básicas (CRUD):
  - Criar usuário;
  - Listar usuários;
  - Atualizar usuário;
  - Excluir usuário.

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Estrutura de pastas do projeto:



# Arquitetura MVC (Model-View-Controller) - Exemplo

- Arquivo requirements.txt:
  - Lista de dependências para instalar:

psycopg2-binary

  - Instale executando:

pip install -r requirements.txt

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Implementação do banco de dados no PostgreSQL:

```
CREATE DATABASE meu_banco;

CREATE TABLE usuarios (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100),
    email VARCHAR(100)
);
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Arquivo model/usuario_model.py: Essa camada faz a comunicação com o banco de dados PostgreSQL.

```
import psycopg2

class UsuarioModel:
    def __init__(self):
        try:
            self.conexao = psycopg2.connect(
                host="localhost",
                database="meu_banco",
                user="postgres",
                password="sua_senha"
            )
        except Exception as e:
            print("Erro ao conectar ao banco:", e)
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def listar_usuarios(self):
        try:
            cursor = self.conexao.cursor()
            cursor.execute("SELECT id, nome, email FROM usuarios ORDER
BY id;")
            usuarios = cursor.fetchall()
            cursor.close()
            return usuarios
        except Exception as e:
            print("Erro ao listar usuários:", e)
            return []
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
    def inserir_usuario(self, nome, email):
        try:
            cursor = self.conexao.cursor()
            cursor.execute("INSERT INTO usuarios (nome, email) VALUES
(%s, %s);", (nome, email))
            self.conexao.commit()
            cursor.close()
        except Exception as e:
            print("Erro ao inserir usuário:", e)
            self.conexao.rollback()
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
    def atualizar_usuario(self, id_usuario, nome, email):
        try:
            cursor = self.conexao.cursor()
            cursor.execute("UPDATE usuarios SET nome = %s, email = %s
WHERE id = %s;", (nome, email, id_usuario))
            self.conexao.commit()
            cursor.close()
        except Exception as e:
            print("Erro ao atualizar usuário:", e)
            self.conexao.rollback()
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
  def excluir_usuario(self, id_usuario):
      try:
          cursor = self.conexao.cursor()
          cursor.execute("DELETE FROM usuarios WHERE id = %s;",
(id_usuario,))
          self.conexao.commit()
          cursor.close()
      except Exception as e:
          print("Erro ao excluir usuário:", e)
          self.conexao.rollback()
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Arquivo view/usuario_view.py: Responsável por interagir com o usuário via terminal.

```
def menu_principal():
    print("\n===== MENU PRINCIPAL =====")
    print("1 - Listar usuários")
    print("2 - Cadastrar usuário")
    print("3 - Atualizar usuário")
    print("4 - Excluir usuário")
    print("0 - Sair")
    return input("Escolha uma opção: ")
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def mostrar_usuarios(lista):
    print("\n=== Lista de Usuários ===")
    if not lista:
        print("Nenhum usuário encontrado.")
    else:
        for usuario in lista:
            print(f"ID: {usuario[0]} | Nome: {usuario[1]} | Email: {usuario[2]}")

def solicitar_dados_usuario():
    nome = input("Nome: ")
    email = input("E-mail: ")
    return nome, email
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def solicitar_id():
    try:
            return int(input("Informe o ID do usuário: "))
    except ValueError:
            print("ID inválido! Deve ser um número inteiro.")
            return None

def mensagem(texto):
    print(texto)
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Arquivo controller/usuario_controller.py: Faz a ponte entre o Model e a View.

```
from model.usuario_model import UsuarioModel
from view.usuario_view import *

class UsuarioController:
    def __init__(self):
        self.model = UsuarioModel()

    def listar(self):
        try:
            usuarios = self.model.listar_usuarios()
            mostrar_usuarios(usuarios)
        except Exception as e:
            mensagem(f"Erro ao listar usuários: {e}")
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def cadastrar(self):
    try:
        nome, email = solicitar_dados_usuario()
        if not nome or not email:
            mensagem("Nome e e-mail são obrigatórios!")
            return
        self.model.inserir_usuario(nome, email)
        mensagem("Usuário cadastrado com sucesso!")
    except Exception as e:
        mensagem(f"Erro ao cadastrar usuário: {e}")
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def atualizar(self):
    try:
        id_usuario = solicitar_id()
        if id_usuario is None:
            return
        nome, email = solicitar_dados_usuario()
        self.model.atualizar_usuario(id_usuario, nome, email)
        mensagem("✏ Usuário atualizado com sucesso!")
    except Exception as e:
        mensagem(f"Erro ao atualizar usuário: {e}")
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

```
def excluir(self):
    try:
        id_usuario = solicitar_id()
        if id_usuario is None:
            return
        self.model.excluir_usuario(id_usuario)
        mensagem("Usuário excluído com sucesso!")
    except Exception as e:
        mensagem(f"Erro ao excluir usuário: {e}")
```

# Arquitetura MVC (Model-View-Controller) - Exemplo

- Arquivo main.py:

```
from controller.usuario_controller import
UsuarioController
from view.usuario_view import menu_principal

def main():
    controller = UsuarioController()

    while True:
        try:
            opcao = menu_principal()
```

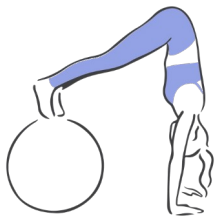
# Arquitetura MVC (Model-View-Controller) - Exemplo

```
            if opcao == "1":
                controller.listar()
            elif opcao == "2":
                controller.cadastrar()
            elif opcao == "3":
                controller.atualizar()
            elif opcao == "4":
                controller.excluir()
            elif opcao == "0":
                print("Saindo do sistema...")
                break
            else:
                print("Opção inválida. Tente novamente.")
        except Exception as e:
            print(f"Ocorreu um erro inesperado: {e}")

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

# Arquitetura MVC (Model-View-Controller) - Exercício

- Desenvolva um programa em Python utilizando o padrão de arquitetura MVC (Model–View–Controller) para gerenciar o cadastro de produtos em um banco de dados PostgreSQL.
- O sistema deve permitir que o usuário interaja por meio de um menu no terminal, realizando as seguintes operações:
    1. Listar produtos cadastrados.
    2. Cadastrar um novo produto (nome e preço).
    3. Atualizar os dados de um produto existente.
    4. Excluir um produto pelo ID.
    5. Sair do sistema.



# FIM