@fpftech.educacional

FPFtech
Escola Tecnológica

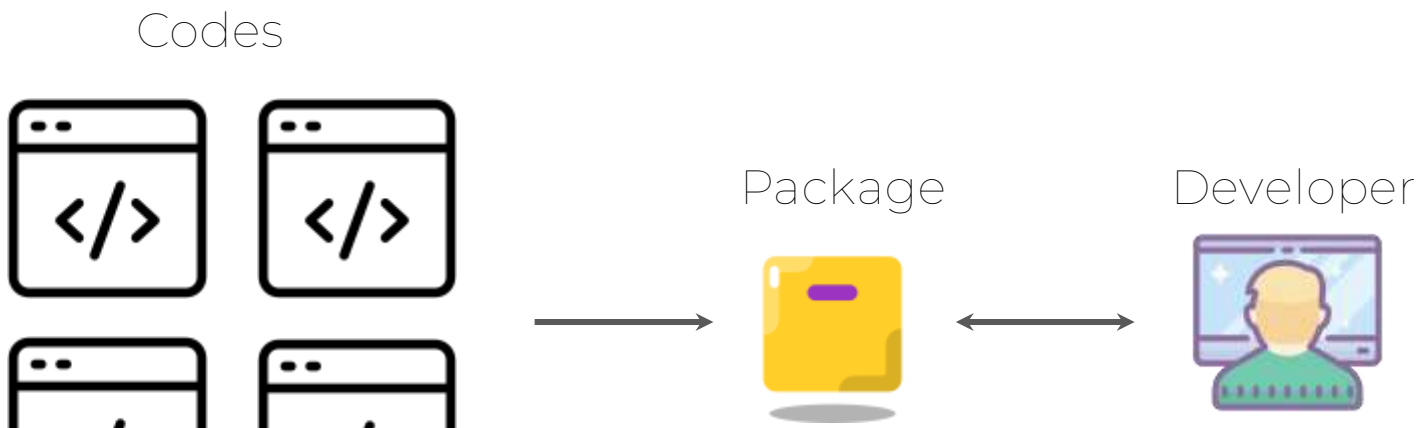
FPFtech
Escola Tecnológica

O que é Framework?

Define a estrutura do seu futuro projeto e proporciona as ferramentas necessárias que você pode usar como blocos de construção.

Framework na área de desenvolvimento de software é uma abstração que une vários códigos genéricos prontos e disponibiliza o pacote para pessoas reutilizar.

Ilustração:



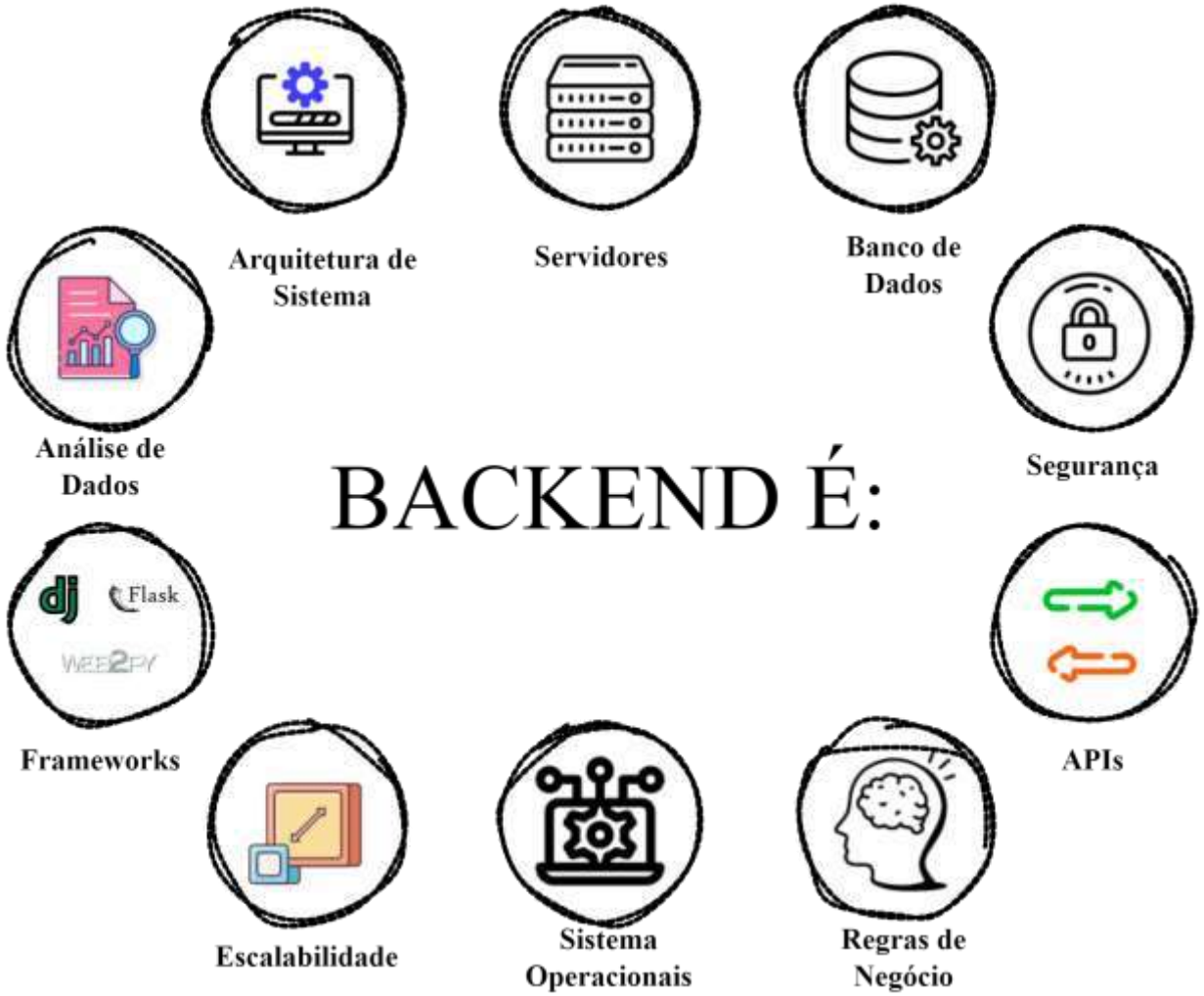
O que é um Backend?

É parte de um sistema ou aplicação que lida com lógica, processamento e armazenamento de dados. Em desenvolvimento de Software se refere à “ parte de trás” de um sistema, longe dos olhos do usuário. Ele é responsável por fazer as funções e operações que permitem que o Frontend (Interface que o usuário realiza interação) funcione corretamente.

A principal função é gerenciar as requisições realizadas por uma interface (Frontend), processar e responder de forma adequada.

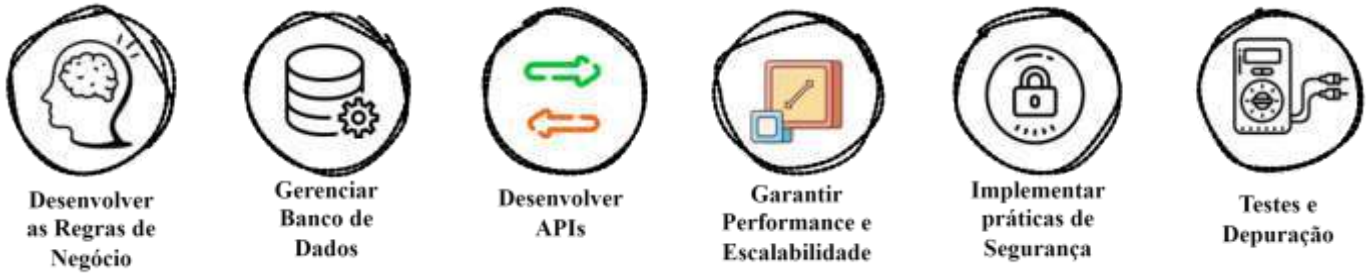
O que é um Backend?

Segue uma ilustração:



O que é um Desenvolvedor Backend?

É o profissional responsável por construir e manter a parte "invisível" de um sistema ou aplicação, ou seja, tudo o que acontece nos bastidores. O trabalho desse desenvolvedor está focado na lógica de negócios, no processamento de dados e na comunicação entre o frontend (o que o usuário vê) e o banco de dados (onde as informações são armazenadas).



O que é um Desenvolvedor Backend?

O desenvolvimento de backend requer conhecimento de vários aspectos. Aqui está um mapa mental do que um desenvolvedor deve aprender:

1 – Fundamentos: Inclui tópicos como backend vs frontend, cliente-servidor, DNS, etc.

2 - Linguagens de Programação de Backend: Escolha entre uma ou mais linguagens de programação, como Java, Python, JS, Go, Rust e C#.

3 - Bancos de Dados: Inclui tópicos como tipos de bancos de dados, como SQL (Postgres, MySQL, SQLite), NoSQL (MongoDB, Firebase, DynamoDB), NewSQL (CockroachDB, Spanner). Outros tópicos incluem trabalhar com ORMs e cache de banco de dados.

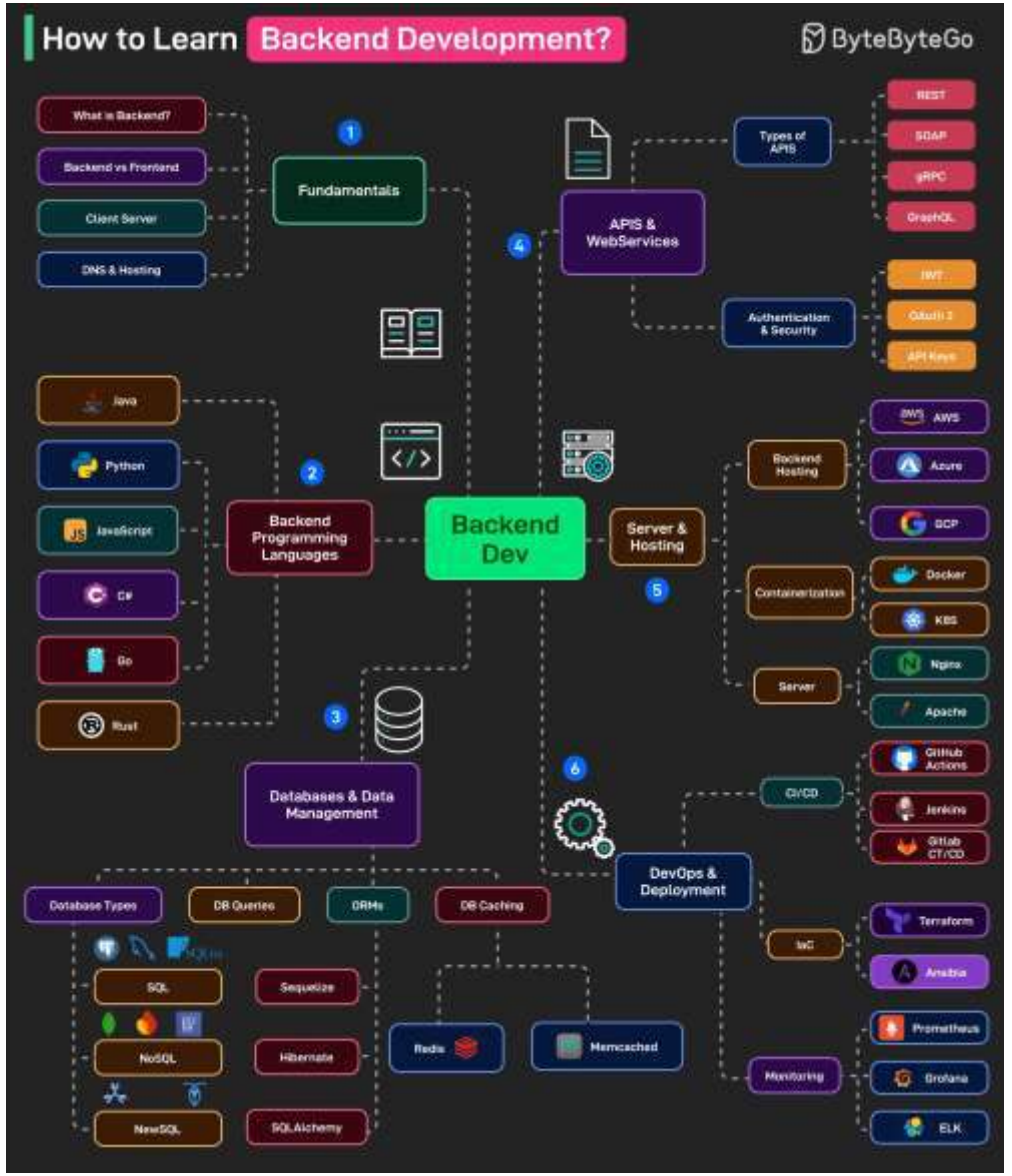
4 - APIs e Serviços Web: Aprenda sobre os tipos de API (REST, GraphQL, gRPC, SOAP) e técnicas de autenticação (como JWT, OAuth 2, chaves de API).

5 - Servidor e Hospedagem: Isso envolve tópicos como serviços de hospedagem de backend (AWS, Azure, GCP), conteinerização usando Docker e Kubernetes e configuração de servidores para Nginx, Apache, etc.

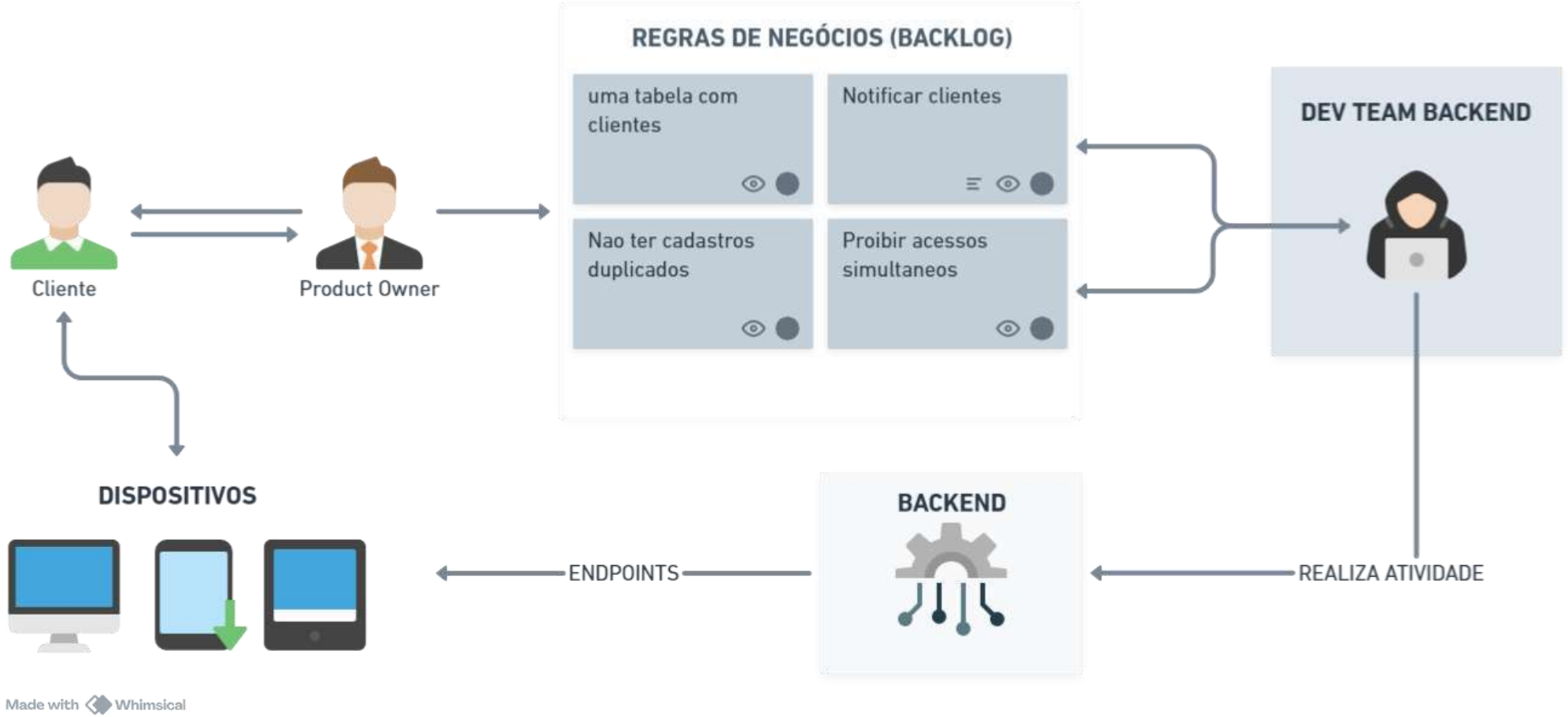
6 – DevOps: Aprenda sobre pipelines de CI/CD usando GitHub Actions e Jenkins, IaC (Terraform, Ansible) e monitoramento com ferramentas como Prometheus, Grafana e ELK.

O que é um Desenvolvedor Backend?

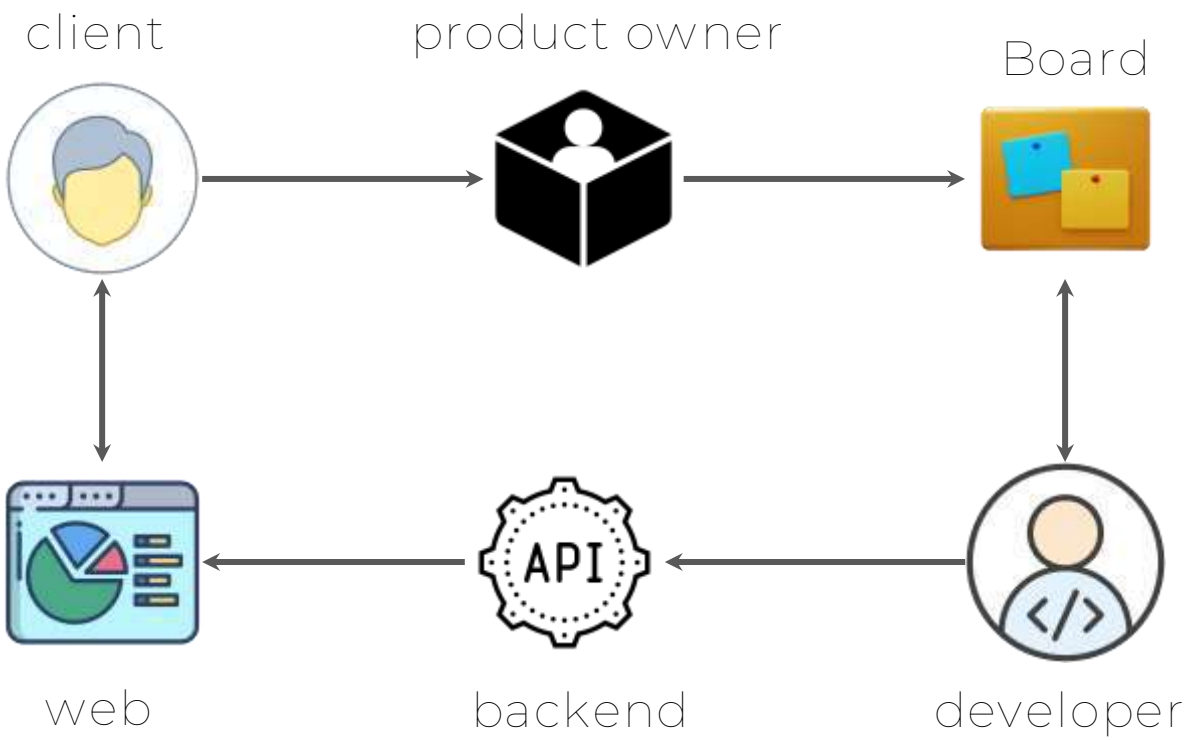
Segue uma ilustração:



Exemplo no Mundo Real



Exemplo no Mundo Real



Framework Backend

Um framework backend é uma ferramenta que busca reutilizar códigos por uma arquitetura que normalmente é MVC (Model View Controller) ou REST (Representational State Transfer) para acelerar no desenvolvimento.

Vantagens:

- Escalabilidade;
- Manutenção e legibilidade;
- Integração com outras tecnologias (Banco de dados, APIs).

Desvantagens:

- Curva de aprendizado (Precisa de um tempo para entender arquitetura);
- Sobrecarga de recursos (Framework pode ter recursos que sua aplicação não necessita);
- Dependência da tecnologia (Framework pode descontinuar o suporte).

Exemplos de Frameworks backend

FPFtech
Escola Tecnológica

python™ | django | Flask | WEB2PY

JS TS | ex | koa | nest

spring boot | HIBERNATE

Framework Django

Um framework Web que utiliza Linguagem Python que incentiva o desenvolvimento rápido com um design limpo. Dentro deste framework tem os seguintes componentes:

- Padrão MVT (Model View Template): Semelhante ao padrão MVC.
- ORM (Object-Relational-Mapping): Utilizar funções que já fazem consultas sem precisar escrever SQL diretamente.
- Admin Panel: Interface que permite o gerenciamento dos dados.
- URLs: usa expressões regulares para roteamento para funções de view.

Vantagens de usar Django

- ✓ Escalabilidade: Django foi projetado para softwares que tendem a escalar conforme o crescimento do projeto.
- ✓ Desenvolvimento rápido: Django já vem com autenticação, roteamento de URLs, sessões desenvolvidas pelo framework, fazendo com que o desenvolvedor se preocupe somente com a regra de negócio e alguns conceitos mais simples.
- ✓ Modularidade e Reutilização de código: facilidade de integrar e reutilizar seus “apps” de forma fácil em diferentes projeto.
- ✓ Segurança: Django já vem com implementação de segurança de injeção de SQL, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) e Clickjacking.

Desvantagens de usar Django

- Alto consumo de recursos: Aplicações Django necessita de mais de recursos de CPU em comparação aos frameworks mais leves como Flask;
- Monolítico: Segue uma estrutura rígida, pode ser gargalo para projetos que requer mais flexibilidade para substituir componentes.
- Não recomendado para projetos pequenos ou simples: Por possuir vários padrões visando escalabilidade, pode não ser útil para projetos com escopo pequeno e fechado.

Empresa que utilizam Django

FPFtech
Escola Tecnológica

Recursos para rodar Django

IDE: Integrated Development Environment (Ambiente de Desenvolvimento Integrado) é um software que permite o desenvolvimento de scripts.

Exemplos:

VENV: Ambiente virtual para isolar dependências do projeto e evitar conflitos de pacotes entre diferentes projetos.

Exemplos:

Recursos para rodar Django

Banco de Dados: Armazenador dos dados, Django já vem com configuração do Sqlite por padrão, porém recomenda-se trocar, visto que o Sqlite só guarda uma capacidade total de 4kb.

Exemplos Relacional:



Exemplos NOSQL (Não relacional):


Antes de tudo, vamos revisar!

FPFtech
Escola Tecnológica

Revisão 1: Herança Simples
Vamos criar uma classe Animal com um método falar(), e uma classe Cachorro que herda de Animal. A classe Cachorro deve implementar o método falar() para imprimir "Au au!".

Revisão 2: Herança com Construtores __init__
Vamos criar uma classe Pessoa com um construtor que recebe um nome. Crie uma classe Estudante que herda de Pessoa, adicionando um atributo matricula.

Revisão 3: Injeção de Dependência Simples
Vamos criar uma classe Mensagem que tenha um método enviar. Crie uma classe Usuário que receba uma instância de Mensagem e use o método enviar para mostrar uma mensagem.

Revisão 4: Herança e Injeção de Dependência
Vamos criar uma classe base Veiculo com um método mover. Crie uma classe Carro que herda de Veículo. Além disso, crie uma classe Motor e injete uma instância de Motor na classe Carro.

Respostas

Exercício 1: Herança Simples
Vamos criar uma classe Animal com um método falar(), e uma classe Cachorro que herda de Animal. A classe Cachorro deve implementar o método falar() para imprimir "Au au!".

exercicio1.py

```python
class Animal:
    def falar(self):
        pass

class Cachorro(Animal):
    def falar(self):
        print("Au au!")


dog = Cachorro()
dog.falar()
```

Respostas

Exercício 2: Herança com Construtores
Vamos criar uma classe Pessoa com um construtor que recebe um nome. Crie uma classe Estudante que herda de Pessoa, adicionando um atributo matricula.

exercicio2.py

```python
class Pessoa:
    def __init__(self, nome):
        self.nome = nome

class Estudante(Pessoa):
    def __init__(self, nome, matricula):
        super().__init__(nome)
        self.matricula = matricula


estudante = Estudante("Carlos", "123456")
print(estudante.nome)
print(estudante.matricula)
```

Respostas

Exercício 3: Injeção de Dependência Simples
Vamos criar uma classe Mensagem que tenha um método enviar. Crie uma classe Usuário que receba uma instância de Mensagem e use o método enviar para mostrar uma mensagem.

exercicio3.py

```python
class Mensagem:
    def enviar(self, texto):
        print(f"Mensagem enviada: {texto}")

class Usuario:
    def __init__(self, mensagem):
        self.mensagem = mensagem

    def notificar(self, texto):
        self.mensagem.enviar(texto)

mensagem = Mensagem()
usuario = Usuario(mensagem)
usuario.notificar("Olá, usuário!")
```

Respostas

Exercício 4: Herança e Injeção de Dependência
Vamos criar uma classe base Veiculo com um método mover. Crie uma classe Carro que herda de Veículo. Além disso, crie uma classe Motor e injete uma instância de Motor na classe Carro.

exercicio4.py

```python
class Veiculo:
    def mover(self):
        pass

class Motor:
    def ligar(self):
        print("Motor ligado")

class Carro(Veiculo):
    def __init__(self, motor):
        self.motor = motor

    def mover(self):
        self.motor.ligar()
        print("O carro está se movendo")

motor = Motor()
carro = Carro(motor)
carro.mover()
```

FPFtech
Escola Tecnológica

Obrigado!

FPFtech
Escola Tecnológica

@fpftech.educacional