Pesquisa para Landing Page

* O que é Django?

Segundo a documentação oficial, o Django é uma estrutura escrita em Python busca o desenvolvimento rápido, design limpo e pragmático. Com código aberto e gratuito, a principal proposta é ajudar o desenvolvedor ou desenvolvedora a pensar na aplicação. Sua principal convenção é o DRY (Don't Repeat Yourself, em tradução livre: não seja repetitivo), que visa o máximo de proveito do código criado evitando código duplicado. O Django utiliza um padrão similar ao MVC (model-view-controller), porém aqui chamado de MTV (model-template-view), onde dividimos a aplicação em camadas ou partes, tornando o código mais organizado e legível.

Imagina que Python é como uma caixa de ferramentas que você usa para construir coisas, como brinquedos ou máquinas. Com ela, você pode criar várias coisas, como jogos ou sites, de um jeito bem fácil e divertido.

Agora, Django e Flask são como "kits de construção" que ajudam você a montar um site. Eles já vêm com peças prontas, que facilitam o trabalho.

- Django é como um super kit de construção que já vem com muitas peças prontas, então você pode construir seu site mais rápido, sem precisar procurar tantas peças.

- Flask é um kit mais simples, onde você escolhe as peças que quer usar, o que dá mais liberdade,mas você precisa trabalhar um pouco mais para achar as peças certas.

Então, Python é a caixa de ferramentas, e Django e Flask são os kits de construção para montar sites, só que um é mais rápido e o outro mais livre.

* O Que é um FrameWork?

O framework é do uma ferramenta que vai te ajudar a ter como único objetivo focar em desenvolver o projeto, não em detalhes de configurações, sendo assim, o framework trouxe a prática de evitar que tenhamos que fazer tarefas repetitivas, automatizando parte do trabalho.

* Vantagens

É possível agilizar trabalho, já que nossos esforços se voltam para o desenvolvimento, em vez de ter a preocupação com detalhes de configurações e padrões de projeto. Com frameworks, há um código mais limpo, garantindo maior clareza de entendimento em tudo que é implementado pela ferramenta, o que facilita o trabalho de quem lida com o projeto.

* Tipos de FrameWorks (micro, full stack e API)

- Full Stack

O termo “Full Stack Framework” refere-se a um conjunto de ferramentas e bibliotecas que permitem o desenvolvimento de aplicações web completas, abrangendo tanto o front-end quanto o back-end. Esses frameworks são projetados para simplificar o processo de desenvolvimento, oferecendo uma estrutura coesa que integra diferentes tecnologias e linguagens de programação. O uso de um Full Stack Framework pode acelerar o desenvolvimento e melhorar a eficiência, pois fornece soluções prontas para problemas comuns enfrentados por desenvolvedores.

- Micro FrameWork

Um micro framework é uma estrutura de desenvolvimento de software que oferece um conjunto mínimo de ferramentas e funcionalidades essenciais para criar aplicações, sem impor uma arquitetura rígida ou um grande número de dependências. Ele é projetado para ser leve, rápido e flexível

- API FrameWork

Um framework API é uma estrutura de desenvolvimento focada em facilitar a criação, implementação e manutenção de APIs (Interfaces de Programação de Aplicações). APIs são conjuntos de regras e definições que permitem que diferentes softwares se comuniquem entre si, e os frameworks API fornecem ferramentas para tornar esse processo mais rápido e organizado

* Arquitetura de software de um FrameWork

Arquitetura de software pode ser entendida como um conjunto de normas, princípios e técnicas para construção de software. De certa forma, assemelha-se à ideia de arquitetura que conhecemos e que delimita os padrões de edificações para casas em ambientes pequenos, frios, quentes etc.

Os padrões de arquitetura mais populares são Model-View-Controller (MVC) e o Model-View-ViewModel (MVVM) Cada um possuindo vantagens únicas, oferecendo soluções para diferentes projetos e necessidades.

Controlador de visualização de modelo (MVC)

MVC é um dos padrões de arquitetura mais conhecidos e adotados pela indústria de software. O padrão facilita a separação de interesses dividindo o aplicativo em três componentes principais:

Modelo: Representa os dados e a lógica de negócios da aplicação. É responsável por processar, armazenar e gerenciar dados e implementar quaisquer regras de negócios necessárias. O modelo é independente da interface do usuário e não se comunica diretamente com a visualização ou o controlador.

Visualizar: representa a interface do usuário (IU) e a camada de apresentação do aplicativo. A função principal da visualização é exibir os dados obtidos do modelo. Ele não acessa diretamente o modelo, mas recebe atualizações por meio do controlador. As visualizações podem ter múltiplas representações visuais dos mesmos dados, permitindo maior flexibilidade e adaptabilidade.

Controlador: atua como intermediário entre o modelo e a visualização. O controlador recebe a entrada do usuário da visualização, processa-a e atualiza o modelo. Depois que o modelo é atualizado, ele notifica o controlador, que então atualiza a visualização com novos dados. A principal responsabilidade do controlador é gerenciar o fluxo do aplicativo e manter o modelo e a visualização sincronizados. A arquitetura MVC promove componentes fracamente acoplados, melhorando a manutenção e os testes de aplicativos.

Modelo-Visualização-ViewModel (MVVM)

O padrão arquitetônico Model-View-ViewModel (MVVM) tem suas raízes nas pilhas de desenvolvimento da Microsoft e foi introduzido como uma resposta às limitações do padrão MVP, com o objetivo de simplificar o desenvolvimento de UI. MVVM é uma evolução do padrão MVP, com foco na separação de interesses e no aprimoramento da testabilidade. O padrão MVVM consiste em três componentes principais:

Modelo: Representa os dados e a lógica de negócios da aplicação. É responsável por recuperar e armazenar dados e processar quaisquer dados necessários.

Visualizar: Representa a interface do usuário e exibe os dados ao usuário. No MVVM, a visualização normalmente é projetada usando uma linguagem de marcação como XAML, que permite uma separação clara entre o design da UI e o code-behind.

ViewModel: Serve como ponte entre o Modelo e a Visualização, responsável por manter o estado da Visualização e realizar quaisquer operações necessárias para transformar os dados dentro do Modelo em um formato amigável à Visualização. Ele fornece ligação de dados entre o modelo e a visualização usando observáveis, comandos e eventos. Essa comunicação normalmente é obtida através da implementação da interface INotifyPropertyChanged.

- Participantes



Fernanda de Oliveira Nunes – edição e gravação do vídeo



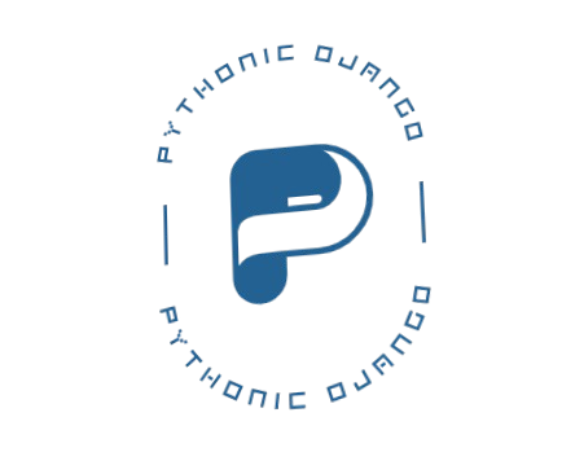
Isabela Costa Jeronymo – roteiro e pesquisa



Lauren Miguel Norberto – apresentação



Lucas Duarte Geraldo – participação e pesquisa



* Roteiro

\*Abertura\*

“Oiii Phytonics, sejam bem-vindos ao nosso vídeo sobre o framework Django

da linguagem Phyton, no final do vídeo vocês poderão ver exemplos práticos

desse framework em funcionamento!”

1. Introdução

Primeiramente, quero explicar pra vocês que a linguagem Python é uma

linguagem de programação que podemos usar para criar jogos, sites e diversas

aplicações. E hoje, vamos explorar um framework muito popular desta

linguagem que é o Django.

2. O que é um Framework?

“E se você caiu aqui de paraquedas, um framework é uma estrutura pré-

definida que nos ajuda a desenvolver aplicações de forma mais eficiente. Ele

automatiza tarefas repetitivas, permitindo que os desenvolvedores se

concentrem no desenvolvimento do projeto. Isso acelera o desenvolvimento,

promove código limpo e facilita a manutenção e colaboração em equipe.”

3. A Importância dos Frameworks Backend é que eles definem a lógica de

negócios, a interação com bancos de dados e a segurança em um projeto.

Assim, os frameworks vão ser responsáveis por simplificar o desenvolvimento e

permitir que os desenvolvedores se concentrem em funcionalidades

específicas.”

4. Tipos de Frameworks

“agora, existem diferentes tipos de frameworks:”

- um framework Full Stack Abrange tanto o front-end quanto o back-end,

oferecendo uma estrutura coesa para aplicações completas.

- já um Micro Framework é Leve e flexível, ideal para projetos menores ou

personalizados.

-e por último, um API Framework vai Focar na criação e manutenção de APIs,

facilitando a comunicação entre diferentes sistemas.

5. Arquitetura de Software

“pra voce entender o django também, e importante voce saber um pouco sobre

Arquitetura de software. Mas o que é arquitetura de Software? Podemos defini-

la como o conjunto de normas, princípios e técnicas para a construção de

software. Os padrões mais populares são

MVC (Model-View-Controller) – é um dos padrões de arquitetura mais

conhecidos e adotados pela indústria de software.

MVP (Model-View-Presenter) – é um padrão arquitetônico que aborda algumas

das desvantagens da abordagem MVC tradicional, com a intenção de melhorar

a separação de interesses entre a visão e o modelo. Esse padrao foi

introduzido pela primeira vez na década de 1990.

MVVM (Model-View-ViewModel) – que também foi uma resposta às limitações

do padrão anterior- o MVP, com o objetivo de simplificar o desenvolvimento de

UI. Seu foco é a separação de interesses e o aprimoramento da testabilidade.”

6. O que é Django?

“com isso, o Django é um framework Python gratuito e de código aberto,

projetado para um desenvolvimento rápido e eficiente de aplicações web. Ele

segue o princípio DRY, que é uma sigla em ingles para Don’t Repeat Yourself,

em tradução livre significa “não seja repetitivo”. E também segue o padrão de

arquitetura de software MTV (Model-Template-View).”

“Com Django, você ganha agilidade no desenvolvimento e uma estrutura

organizada, ideal para projetos complexos.”

7. Comparação entre Django e Flask

além do django, flask também é um framework para linguagem python.

Podemos destacar de diferenças entre os dois que Django é um framework

completo com muitas funcionalidades prontas, ideal para projetos que exigem

rapidez e estruturação. Já o Flask é minimalista e flexível, oferecendo mais

liberdade, mas exigindo mais trabalho para configurar e integrar componentes.

8. Conclusão

Python é uma ferramenta poderosa para desenvolvimento, e Django e Flask

são frameworks que facilitam a criação de aplicações web. A escolha entre

Django e Flask depende das necessidades do projeto: Django para projetos

estruturados e rápidos, Flask para projetos mais personalizados e flexíveis.

Frameworks backend são fundamentais para acelerar o desenvolvimento e

garantir a qualidade do software.

\*Finalização\*

Link Video - <https://drive.google.com/file/d/1bZaVovZyKWJUIx9gq8_j1zm28YZ1VoTN/view?usp=sharing>