**CENTRO UNIVERSITÁRIO ESTÁCIO DE JUIZ DE FORA**

**COMPARAÇÃO ENTRE FERRAMENTAS PARA DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTON**

**GUILHERME LOUBACK DE VIRGILIO**

**Juiz de Fora, setembro de 2023**

Sumário

[**1) INTRODUÇÃO** 3](#_Toc145580260)

[**2)FRAMEWORKS EM PYTHON** 3](#_Toc145580261)

[**2.1) Áreas De Aplicação** 3](#_Toc145580262)

[**3) FLASK** 3](#_Toc145580263)

[**3.1) História** 3](#_Toc145580264)

[**3.2) Motivação** 4](#_Toc145580265)

[**3.3) Princípios Básicos** 4](#_Toc145580266)

[**3.4) Exemplo de aplicações** 4](#_Toc145580267)

[**4) DJANGO** 4](#_Toc145580268)

[**4.1) História** 5](#_Toc145580269)

[**4.2) Motivação** 5](#_Toc145580270)

[**4.3) Princípios básicos** 5](#_Toc145580271)

[**4.4) Exemplo de aplicações** 5](#_Toc145580272)

[**5) APLICAÇÃO DE EXEMPLO** 6](#_Toc145580273)

[**5.1) Código em Flask** 6](#_Toc145580274)

[**5.2) Código em Django** 6](#_Toc145580275)

[**6) CONCLUSÕES** 6](#_Toc145580276)

[**7) REFERÊNCIAS** 8](#_Toc145580277)

**1) INTRODUÇÃO**

Frameworks são conjuntos de bibliotecas, padrões e diretrizes que auxiliam no desenvolvimento de aplicações de software de forma mais eficiente e organizada. No contexto da linguagem de programação Python, dois dos frameworks mais populares são Django e Flask. Esta revisão bibliográfica tem como objetivo contextualizar esses frameworks, abordando sua história, motivação, princípios básicos e fornecendo exemplos de aplicações em cada um deles.

**2)FRAMEWORKS EM PYTHON**

Python é uma linguagem de programação conhecida por sua simplicidade, legibilidade e versatilidade. Ela se tornou muito popular para o desenvolvimento web devido à disponibilidade de diversos frameworks que facilitam a criação de aplicativos web. Dois dos principais frameworks em Python são Django e Flask.

**2.1) Áreas de Aplicação**

Ambos Django e Flask são utilizados para desenvolver aplicações web, mas eles atendem a diferentes necessidades e possuem abordagens distintas. Django é um framework de alto nível que favorece a produtividade e a convenção sobre configuração, enquanto Flask é um microframework que oferece maior flexibilidade e é mais adequado para projetos menores e simples.

**3) FLASK**

**3.1) História**

Flask foi criado por Armin Ronacher em 2010 e é frequentemente chamado de "microframework" devido à sua simplicidade e tamanho compacto. Ronacher desenvolveu o Flask como uma alternativa leve e descomplicada a outros frameworks mais pesados, como o Django. Flask adota a filosofia "faça você mesmo", o que significa que oferece as ferramentas básicas necessárias para o desenvolvimento web, mas permite que os desenvolvedores escolham as bibliotecas e ferramentas adicionais que desejam usar.

**3.2) Motivação**

A motivação por trás do Flask era fornecer aos desenvolvedores um ambiente minimalista e flexível para a criação de aplicativos web. Ao manter o núcleo do framework pequeno, o Flask permite que os desenvolvedores escolham as peças que melhor se adequam aos seus projetos, promovendo a simplicidade e evitando abstrações excessivas.

**3.3) Princípios Básicos**

Os princípios básicos do Flask incluem simplicidade, extensibilidade e minimalismo. O framework não impõe uma estrutura de projeto específica, tornando-o flexível para se adaptar às necessidades de diferentes projetos. Ele fornece apenas as funcionalidades essenciais, como roteamento de URLs e manipulação de solicitações HTTP, deixando o restante do desenvolvimento nas mãos dos desenvolvedores.

**3.4) Exemplo de aplicações**

Um exemplo típico de aplicação Flask seria um blog pessoal. O código em Flask para criar uma aplicação de blog seria relativamente simples, definindo rotas para exibir postagens, permitir comentários e gerenciar o conteúdo do blog.

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return "Bem-vindo ao meu blog!"

@app.route('/post/<int:post\_id>')

def show\_post(post\_id):

# Lógica para recuperar e exibir uma postagem específica

return "Postagem {}".format(post\_id)

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**4) DJANGO**

**4.1) História**

Django foi desenvolvido por Adrian Holovaty e Simon Willison em 2003, originalmente como um sistema de gerenciamento de conteúdo (CMS). Ele foi lançado como software de código aberto em 2005 e desde então evoluiu para um framework web de alto nível e extremamente poderoso. Django é conhecido por seguir o princípio "bateria inclusa", fornecendo muitas funcionalidades prontas para uso.

**4.2) Motivação**

A motivação por trás do Django era simplificar o desenvolvimento web, tornando-o mais rápido e acessível. Ele foi projetado para automatizar tarefas comuns, como autenticação de usuários, administração de bancos de dados e geração de formulários, permitindo que os desenvolvedores se concentrem na lógica de negócios de suas aplicações.

**4.3) Princípios básicos**

Os princípios básicos do Django incluem produtividade, consistência e convenção sobre configuração. O framework segue uma estrutura MVC (Model-View-Controller) chamada de MTV (Model-Template-View), que ajuda a organizar o código de forma clara. Além disso, o Django incentiva o uso de ORM (Object-Relational Mapping) para interagir com bancos de dados, tornando o acesso aos dados mais simples e seguro.

**4.4) Exemplo de aplicações**

Um exemplo de aplicação Django seria um sistema de comércio eletrônico. O Django oferece um conjunto de ferramentas robustas para gerenciar produtos, pedidos, autenticação de usuários e muito mais. Abaixo, um trecho de código simplificado para criar um modelo de produto:

from django.db import models

class Product(models.Model):

name = models.CharField(max\_length=100)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

def \_\_str\_\_(self):

return self.name

**5) APLICAÇÃO DE EXEMPLO**

**5.1) Código em Flask**

from flask import Flask, render\_template, request, redirect, url\_for

app = Flask(\_\_name\_\_)

tasks = []

@app.route('/')

def index():

return render\_template('index.html', tasks=tasks)

@app.route('/add', methods=['POST'])

def add\_task():

task = request.form.get('task')

tasks.append(task)

return redirect(url\_for('index'))

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**5.2) Código em Django**

from django.shortcuts import render, redirect

from .models import Task

def index(request):

tasks = Task.objects.all()

return render(request, 'index.html', {'tasks': tasks})

def add\_task(request):

if request.method == 'POST':

task = request.POST['task']

Task.objects.create(name=task)

return redirect('index')

**6) CONCLUSÕES**

Em resumo, Django e Flask são dois frameworks Python populares para o desenvolvimento web, cada um com suas próprias características e filosof

**7) REFERÊNCIAS**

[The web framework for perfectionists with deadlines | Django (djangoproject.com)](https://www.djangoproject.com/)  
[Welcome to Flask — Flask Documentation (2.3.x) (palletsprojects.com)](https://flask.palletsprojects.com/en/2.3.x/)  
[W3Schools Online Web Tutorials](https://www.w3schools.com/)