

Python - Características e Aplicabilidade na Web

Matheus Rambo da Roza¹, Yuri Andreas May Henrique¹

¹Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brasil

matheusrambo97@gmail.com, yuri.henrique@edu.udesc.br

Abstract. *The industrial environment requires increasing agility in the creation of applications and services. Promoting the use of tools that enable rapid code development and testing of prototyping environments is increasingly needed. Combining the use of different programming languages and paradigms has the potential to achieve this goal. In this work, the use of the interpreted language Python for the agile development of web applications will be addressed.*

Resumo. *O meio industrial requer cada vez maior agilidade na criação de aplicações e serviços. A promoção do uso de ferramentas que habilitem o rápido desenvolvimento de código e prototipagem de ambientes é cada vez mais necessária. Aliar o uso de diferentes linguagens e paradigmas de programação possui potencial para alcançar este objetivo. Neste trabalho, será abordado o uso da linguagem interpretada Python, para o desenvolvimento ágil de aplicações web.*

1. Objetivo

O objetivo deste artigo é apresentar a tecnologia Python, suas características principais, sua importância de mercado, e sua usabilidade no desenvolvimento de aplicações web. Estas aplicações são possíveis geralmente através de bibliotecas e *frameworks*, os quais também serão abordados durante este trabalho, com destaque em dois *frameworks*: Django e Flask.

2. Justificativa

A necessidade de alternativas de desenvolvimento web que possam ser prototipadas e publicadas em intervalos de tempo cada vez menores torna necessária a usabilidade de ferramentas que possam fugir do contexto usual de desenvolvimento web, dada através da criação de código diretamente por parte do programador. Alternativas eficazes são identificadas através do uso de *templates* ou *geradores de código genérico*, que possam ser configuráveis de acordo com a necessidade ou modelo de negócio.

Utilizando-se de padrões semelhantes de desenvolvimento, é possível estruturar modelos de desenvolvimento web que envolvam linguagens não-naturais à navegadores. Estas linguagens são utilizadas como meios de desenvolvimento com paradigmas de programação e padrões de projeto predefinidos e inerentes, o que possibilita rápida implementação de projetos. Uma das linguagens que vem ganhando maior notoriedade para este tipo de atividade é Python. Os motivos pelos quais a linguagem possui tamanho destaque serão discutidos no decorrer do trabalho.

3. Python - Linguagem de Programação

Lançada em 1991, Python é uma linguagem de programação multiparadigmática, de fácil aprendizado, com ênfase na legibilidade, voltada para desenvolvimento ágil. Pode ser utilizada para as mais diversas tarefas, contando com uma grande variedade de bibliotecas e frameworks, e excelente integração a muitas ferramentas [Menezes 2010]. A extensibilidade e facilidade de uso da linguagem a tornam atrativa ao mercado, e consequentemente, indicam grande importância no acervo de ferramentas do desenvolvedor. Este artigo visa explorar a linguagem no contexto de desenvolvimento web, e apresentar algumas das ferramentas disponíveis.

Suas características incluem:

- **Legibilidade de código**, voltada para a rápida compreensão do desenvolvedor;
- **Rápida implementação**, com poucas linhas de código e regras de implementação bem definidas;
- **Tipagem dinâmica**, com alocação inteligente da definição dos recursos de cada variável;
- **Multiparadigmática**, podendo ser utilizada em blocos ou também orientada a objeto;
- **Ambientes virtualizados**, para controle e melhor administração dos componentes de cada projeto;
- **Enorme coleção de bibliotecas e frameworks**, tornando a linguagem extremamente popular nos mais diversos contextos, incluindo e não restrito a *machine learning*, ciência de dados, *internet of things* e reconhecimento de padrões.

O uso dentro de diversos contextos não vem sem algumas desvantagens, afinal quanto maior a abstração de uma linguagem, maiores as chances que ela necessite de mais recursos e maior refinamento de sua execução e tratamento de exceções:

- **Lenta velocidade de execução** no sistema operacional, comparada a outras linguagens de programação, quando realizando as mesmas tarefas;
- **Alto consumo de memória**, devido a abrangência de seus componentes principais como a própria tipagem dinâmica e virtualização do ambiente de execução;
- **Difícil captura de erros em tempo de execução**, que podem acontecer a qualquer momento e não são detectáveis de forma prévia por não ser uma linguagem compilada;
- **Restrições de design**, que não permite o acesso e teste do programa implementado de forma que o interpretador não reconheça nativamente;
- **Implementação mobile abaixo da média**, por mais que existam opções viáveis para desenvolvimento de aplicações móveis, não é suficiente para ponderar em um nível maior do que linguagens como javascript ou swift.

3.1. Relevância no Mercado

O índice TIOBE [TIOBE 2021] é responsável por categorizar a popularidade das linguagens de programação atualmente utilizadas, mediante estatísticas sobre a frequência na qual as linguagens são pesquisadas via motores de busca como *Google*, *Bing*, *Yahoo*, entre diversos outros. Na Figura 3.1 são apresentadas as dez linguagens mais procuradas na internet, com Python ocupando a terceira colocação geral. Esta colocação mostra











Jul 2021	Jul 2020	Change	Programming Language	Ratings	Change
1	1		 C	11.62%	-4.83%
2	2		 Java	11.17%	-3.93%
3	3		 Python	10.95%	+1.86%
4	4		 C++	8.01%	+1.80%
5	5		 C#	4.83%	-0.42%
6	6		 Visual Basic	4.50%	-0.73%
7	7		 JavaScript	2.71%	+0.23%
8	9	^	 PHP	2.58%	+0.68%
9	13	^^	 Assembly language	2.40%	+1.46%
10	11	^	 SQL	1.53%	+0.13%

Figura 1. Classificação das dez mais populares linguagens de programação, em julho de 2021

grande interesse por parte da comunidade de desenvolvedores em materiais envolvendo seus recursos.

Além disso, diversas instituições e organizações multinacionais fazem uso deste recurso para o desenvolvimento de suas aplicações sistemas, como a Uber [Lozinski 2016], as APIs do Spotify [Engineering 2011], as plataformas do Facebook e Instagram [Labs 2013] e inclusive a NASA [Shafer 2008], no controle de missões espaciais.

3.2. Frameworks

Dos vários *frameworks* disponíveis para uso em desenvolvimento web, foram escolhidos dois para maior abordagem neste trabalho. Django é um *framework* completo, integrado com o padrão de projeto *Model View Template*, possibilitando a implementação de um sistema web de forma fácil e rápida. Flask é um *microframework*, caracterizado para uso principalmente de ambientes voltados a *Internet of Things*, possuindo como características principais a rápida implementação e alta extensibilidade.

3.3. Django

Django (identificado na Figura 2) nasceu em 2003 em uma agência de notícias de Lawrence, Kansas. É uma estrutura da web que usa Python para criar sites. Seu objetivo é escrever sites dinâmicos muito rápidos. Em 2005, a agência decidiu publicar o código-fonte do Django na licença BSD. Em 2008, a Django Software Foundation foi criada para apoiar e promover o Django. A versão 1.00 do framework foi lançada alguns meses depois. O Slogan do Django já diz tudo sobre ele, "O framework web para perfeccionistas com prazos". Este framework foi criado para acelerar a fase de desenvolvimento dos sites. Na verdade, esse framework usa o padrão *Model-View-Controller* (MVC), o que nos permite ter uma arquitetura coerente. Até 2013, o Django era compatível apenas



Figura 2. Logotipo do Framework Django

com o Python versão 2.x, mas o Django 1.5 lançado em 26 de fevereiro de 2013, aponta para o início da compatibilidade com Python 3. Hoje, grandes organizações como o site móvel Instagram, Mozilla.org e Openstack.org estão usando Django. Existem diversas vantagens em se usar o Django [Dauzon et al. 2016], como por exemplo:

- Django é totalmente personalizável. Os desenvolvedores podem se adaptar a ele facilmente criando módulos ou métodos de estrutura substituídos.
- Usar Python neste framework permite que você tenha os benefícios de todas as bibliotecas Python e garante uma boa legibilidade.
- Django é um framework cujo objetivo principal é a perfeição. Ele foi feito especificamente para pessoas que desejam um código claro e uma boa arquitetura para seus aplicativos. Respeita totalmente a filosofia Don't Repeat Yourself (DRY), que significa manter o código simples sem ter que copiar / colar as mesmas partes em vários lugares.
- Django é apoiado por uma boa comunidade. Este é um ativo muito importante porque permite que você resolva problemas e conserte bugs muito rapidamente. Graças à comunidade, também podemos encontrar exemplos de código que mostram as melhores práticas.

Nas seções seguintes iremos estudar mais a fundo esse mundo do framework Django e apresentar diversos conceitos que irão provar o quão facilitador esse framework é na vida de um programador web.

3.3.1. Estrutura MVC

Antes da existência da estrutura MVC, a programação da web misturava o código de acesso ao banco de dados e o código principal da página. Mesmo que estejamos armazenando arquivos CSS e JavaScript em arquivos externos, os códigos de linguagem do lado do servidor são armazenados em um arquivo compartilhado entre pelo menos três linguagens: Python, SQL e HTML. O padrão MVC foi criado para separar a lógica da representação e ter uma arquitetura interna mais tangível e real. O *Model-View-Controller* (MVC) representa as três camadas de aplicação que o paradigma recomenda:

- Modelos: representam a organização de dados em um banco de dados. Em palavras simples, podemos dizer que cada modelo define uma tabela no banco de dados e as relações entre os outros modelos. É graças a eles que cada bit de dados é armazenado no banco de dados.

- Visualizações: contém todas as informações que serão enviadas ao cliente. Eles criam visualizações que o documento HTML final irá gerar. Podemos associar o código HTML às visualizações.
- Controladores: contém todas as ações realizadas pelo servidor e não são visíveis ao cliente. O controlador verifica se o usuário está autenticado ou pode gerar o código HTML a partir de um modelo.

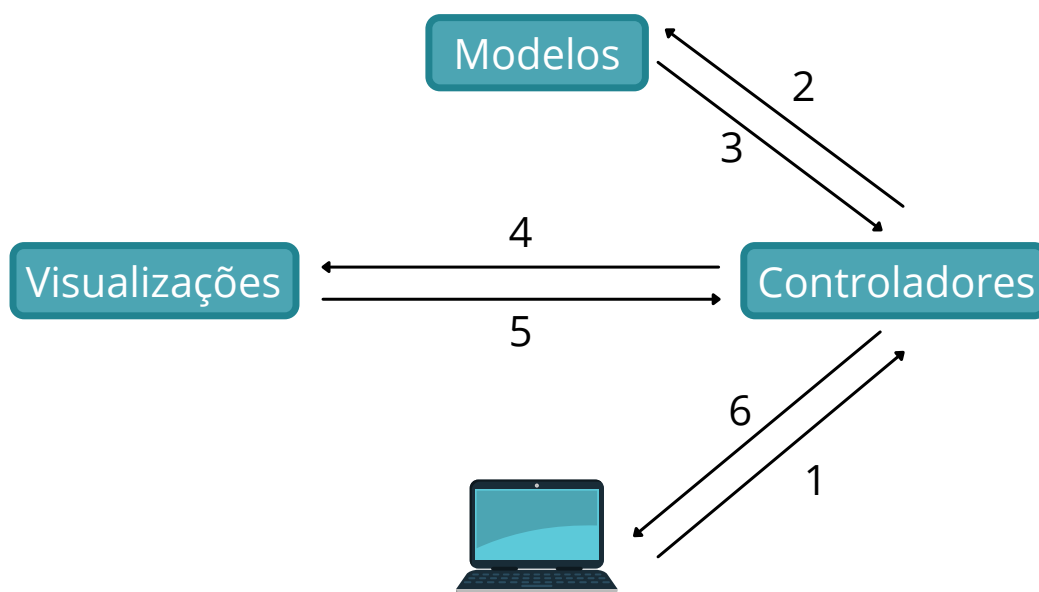


Figura 3. Imagem que ilustra o modelo MVC [Dauzon et al. 2016]

A Figura 3 representa as etapas que são seguidas em um aplicativo com o padrão MVC. (1) O cliente envia uma solicitação ao servidor solicitando a exibição de uma página, após isso (2) o controlador usa um banco de dados por meio de modelos. Ele pode criar, ler, atualizar ou excluir qualquer registro ou aplicar qualquer lógica aos dados recuperados. (3) O modelo envia dados do banco de dados; por exemplo, ele envia uma lista de produtos se tivermos uma loja online. (4) O controlador injeta dados em uma visualização para gerá-los. (5) A visualização retorna seu conteúdo dependendo dos dados fornecidos pelo controlador e aí então (6) o controlador retorna o conteúdo HTML ao cliente.

O padrão MVC nos permite obter coerência para cada trabalhador do projeto. Em uma empresa de criação de sites onde há um web designer e há desenvolvedores, o web designer é o chefe das visualizações. Dado que as visualizações contêm apenas o código HTML, o web designer não será perturbado pelo código do desenvolvedor. Os desenvolvedores editam seus modelos e controladores. Django, em particular, usa um padrão MVT. Neste padrão, as visualizações são substituídas por modelos e os controladores são substituídos por visualizações. Consequentemente, nosso código HTML serão modelos e nosso código Python serão visualizações e modelos.

3.3.2. Instalação e Utilização do Framework

Nessa seção, mostraremos como realizar a instalação e utilização do Framework Django. Todas as informações que aqui serão descritas podem ser encontradas em <https://www.djangoproject.com/>. As aplicações web Django podem rodar em quase todas as máquinas que suportam a linguagem de programação Python 3 (Windows, macOS, Linux/Unix são alguns desses SO's). A maioria dos computadores atuais devem ter o desempenho necessário para operar Django. Para realizar o download do Django, pode ser feito de três maneiras:

- Python Package Repository (PyPi), usando o comando pip. Esta é a melhor forma de conseguir a última versão estável do Django.
- A partir de uma versão do gerenciador de pacotes de seu computador. Distribuições de Django que são empacotadas com o sistema operacional oferecem uma "instalação familiar". Contudo, note que a versão disponível pode ser um pouco antiga, e que pode ser instalada apenas dentro do sistema ambiente do Python (podendo ser algo que você não queira).
- Instalar pelo código-fonte. Você pode pegar a última versão acessível do código do Django e instalar no seu computador. Não é recomendado aos iniciantes, mas é necessário quando você se sentir pronto para contribuir com o Django.

Tratando então de banco de dados, o Django suporta a princípio quatro bancos de dados (PostgreSQL, MySQL, Oracle, e SQLite), embora, seja possível utilizar bibliotecas community que fornecem níveis variados de suporte para outros populares bancos de dados SQL e NoSQL.

Iniciando a instalação do Django, iremos rodar o comando abaixo.

```
pip3 install django
```

Para garantir que a instalação foi bem sucedida, rode o abaixo, você deverá ter como saída a versão mais recente do Django.

```
python3 -m django --version
```

Presumindo que apareceu a versão mais recente, vamos realizar um teste mais interessante. Criaremos um esqueleto de um projeto e vamos executá-lo para vê-lo funcionando. Para fazer isso, para isso navegue em seu prompt de comando/terminal até o diretório que quer armazenar seus aplicativos Django. Crie uma pasta para seu site e navegue nela.

```
mkdir django_test  
cd django_test
```

Agora iremos criar um site de teste usando a ferramenta *django-admin*, nosso site vai se chamar *meusite*.

```
django-admin startproject meusite
```

Após criar o site você pode navegar dentro da pasta onde encontrará o script principal para gerenciar projetos, nomeado *manage.py*.

```
cd meusite
```

Nós podemos rodar o web server de desenvolvimento dentro dessa pasta usando o `manage.py` e o comando `runserver`, como mostrado. *manage.py*.

```
python3 manage.py runserver
```

Agora que com o servidor rodando, você pode acessar o site colocando a seguinte URL no seu navegador local: `http://127.0.0.1:8000/`. Como resultado, você irá ver uma página como a que está na Figura 4.

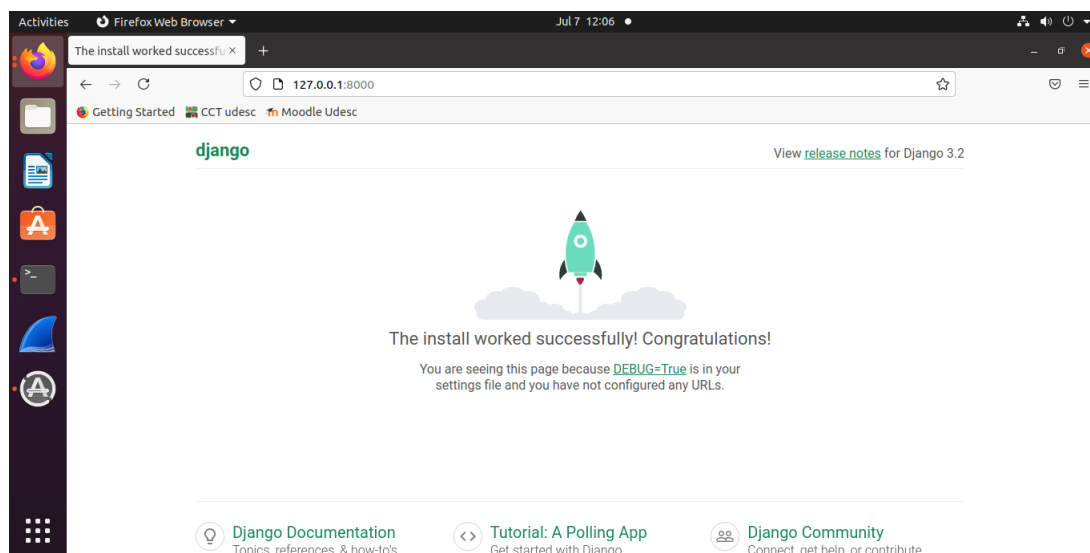


Figura 4. Imagem que exemplifica o uso do Django (acervo do autor)

Aplicativos web feitos com Django geralmente agrupam o código que manipula cada uma dessas etapas em arquivos separados, como URLs, view, models e templates.

- **URLs:** Mesmo que seja possível processar solicitações de cada URL através de uma única função, é mais viável e mais simples realizar a manutenção do código escrevendo uma função view separada para manipular cada recurso. Um mapeador de URLs é usado para redirecionar solicitações HTTP para view apropriada com base na URL da solicitação.
- **View (Vista):** View é uma função manipuladora de solicitações, que recebe solicitações HTTP e retorna as respostas HTTP. As views acessam os dados necessários para satisfazer solicitações por meio dos models (modelos) e encarregam a formatação da resposta aos templates.
- **Models (Modelos):** Modelos são objetos em Python que definem a estrutura dos dados de um aplicativo, e fornecem mecanismos para gerenciar (adicionar, modificar e excluir) e consultar registros no banco de dados.
- **Templates:** Um template é um arquivo de texto que define a estrutura ou o layout de um arquivo (como uma página HTML), com espaços reservados usados para representar o conteúdo real. Uma view pode criar dinamicamente uma página HTML usando um template HTML, preenchendo-a com dados de um model (modelo). Um template pode ser usado para definir a estrutura de qualquer tipo de arquivo, não necessitando ser HTML.

Não iremos apresentar códigos de uma aplicação em Django por razões de deixar o artigo muito extenso e casativo. Mas para quem se interessar, basta acessar

<https://www.devmedia.com.br/como-criar-um-blog-com-django-e-python/33710>, neste link você vai encontrar a criação passo a passo de um blog.

3.4. Flask



Figura 5. Logotipo do Framework Flask

Flask (identificado na Figura 5) é um microframework, baseado em *Web Server Gateway Interface* (WSGI), utilizado para o desenvolvimento de aplicações web. O "mi-cro" é uma palavra referenciando a leveza da ferramenta, e não suas capacidades como um todo, pois conta com grande extensibilidade, além de integração com a maioria das bibliotecas de desenvolvimento da linguagem Python [Aslam et al. 2015]. Seu lançamento inicial foi dado no dia 1 de abril de 2010, e desde então o uso da ferramenta vem ganhando cada vez mais notoriedade.

Algumas das características notáveis do framework são:

- Fácil integração com diversas extensões;
- Escalável conforme a necessidade;
- Ambiente ágil para prototipagem de serviços REST;
- Uso de interfaces para comunicação e engine para renderização;
- Grande acervo de documentação;
- Popular com o uso de *Internet of Things*.

A curva de aprendizado da ferramenta também é menor comparada a outros frameworks que executam as mesmas funções. Isto também é devido ao uso simplificado de sua interface, operando usualmente com chamadas a requisições e serviços de forma direta e discricionária.

3.4.1. Werkzeug

Werkzeug é a WSGI presente no Flask, responsável por conectar as formas na qual o framework irá manipular requisições. Assim como o framework, ela não traz consigo quaisquer dependências, o que faz com que o desenvolvedor tenha que arquitetar a forma que a comunicação web será realizada inicialmente. Dentro do framework, entretanto, isto já vem pré-configurado através da disposição de ambientes de desenvolvimento básicos.

3.4.2. Jinja2

O microframework apresenta como recurso para renderização de páginas web o Jinja2, uma *template engine* que possibilita a passagem de linguagem de programação para

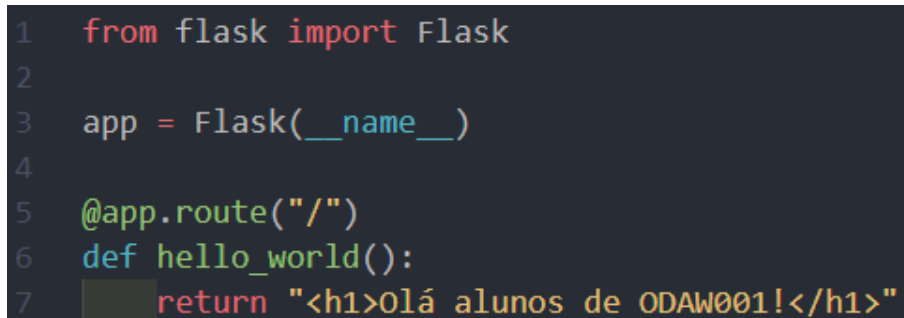
incorporação em páginas no formato HTML. Os recursos providos pela engine possibilitam também o desenvolvimento ágil de código HTML através de estruturas de dados, laços, heranças e outras características próprias do Python.

3.4.3. Exemplo de Aplicação Básica

A criação de uma aplicação simples pelo Flask pode ser feita em poucos passos. Inicialmente, é necessária a aquisição do framework, através do pip:

```
pip install flask
```

Escolha um diretório para criar um arquivo *hello.py*, com as seguintes instruções da Figura 6:

A screenshot of a code editor with a dark background. It shows seven lines of Python code for a Flask application. Line 1: 'from flask import Flask'. Line 2: an empty line. Line 3: 'app = Flask(__name__)'. Line 4: an empty line. Line 5: '@app.route("/")'. Line 6: 'def hello_world()'. Line 7: 'return "<h1>Olá alunos de ODAW001!</h1>"'.

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5  @app.route("/")
6  def hello_world():
7      return "<h1>Olá alunos de ODAW001!</h1>"
```

Figura 6. Código de Aplicação Simples com Flask (acervo do autor)

Após, é necessário indicar a variável de controle para a aplicação no sistema. No Linux, isto pode ser feito através do comando:

```
$ export FLASK_APP=hello
```

E no Windows, quando feito pelo prompt de comando, pode ser feito por:

```
set FLASK_APP=hello.py
```

Então é só executar a aplicação, com:

```
python3 -m flask run
```

O servidor é inicializado por padrão no endereço localhost:5000. O resultado é visto conforme a figura 7:

No terminal, é possível verificar os status das requisições em tempo real, como observa-se na figura 8

3.4.4. Desvantagens Principais

Como o Flask foi feito para ser uma interface de desenvolvimento simples e escalável, a versão base (obtida através do gerenciador de pacotes do Python - pip) não conta com abstrações para bancos de dados, modelos de gerenciamento autenticação do serviço, independente de stack. Ainda, não é recomendado para o uso em grandes aplicações, por



Figura 7. Demonstração da Aplicação de Página em Flask (acervo do autor)

```
D:\flasktest>python -m flask run
* Serving Flask app 'min.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [26/Jul/2021 18:19:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Jul/2021 18:20:32] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Jul/2021 18:20:32] "GET /favicon.ico HTTP/1.1" 404 -
```

Figura 8. Demonstração da Aplicação de Página em Flask (acervo do autor)

maior que seja seu potencial, pois é uma plataforma gulosa para extensões, o que pode causar problemas de gerenciamento conforme o crescimento do projeto. Nestes casos, sugere-se o uso de Django, de forma similar como a abordada anteriormente, por prestar os mesmos tipos de serviços de forma mais abstraída.

4. Conclusões

Python é uma poderosa linguagem para o acervo do desenvolvedor atualmente. Seu uso é completamente adaptável, e pode ser voltado às mais diferentes tarefas. A extensa gama de bibliotecas e frameworks disponíveis é muito atraente para o mercado, oferecendo a implementação ágil e eficiente para serviços distintos.

Referências

- Aslam, F. A., Mohammed, H. N., Mohd, J. M., Gulamgaus, M. A., and Lok, P. (2015). Efficient way of web development using python and flask. *International Journal of Advanced Research in Computer Science*, 6(2):54–57.
- Dauzon, S., Bendoraitis, A., and Ravindran, A. (2016). *Django: web development with Python*. Packt Publishing, 1th edition.
- Engineering, I. (2011). What powers instagram: Hundreds of instances, dozens of technologies. <https://instagram-engineering.com/what-powers-instagram-hundreds-of-instances-dozens-of-technologies-adf2e22da2ad/>.
- Labs, S. (2013). How we use python at spotify. <https://engineering.atspotify.com/2013/03/20/how-we-use-python-at-spotify/>.
- Lozinski, L. (2016). The uber engineering tech stack, part i: The foundation. *uber.com*.

Menezes, N. N. C. (2010). Introdução à programação com python. *Algoritmos e lógica de programação para iniciantes*. Novatec.

Shafer, D. G. (2008). Python streamlines space shuttle mission design. *Python Software Foundation*.

TIOBE (2021). Tiobe - the software quality company. *TIOBE Index — TIOBE—The Software Quality Company*.