LUIS FERNANDO ALVES DA SILVA
PEDRO RAMIRES DA SILVA AMALFI COSTA
REGINADO DONIZETI CÂNDIDO

**ARTIGO Arquitetura de um Software Web em Python com o Padrão
MVC Utilizando o Micro-Framework Web Flask e o Mapeamento
Objeto-Relacional Peewee**

DOI

**MOGI GUAÇU - SP
2019**

RESUMO

O artigo tem como objetivo uma breve explicação de como é possível se estruturar um aplicativo Web utilizando o padrão de projeto MVC utilizando o Micro-Framework Web Flask e o Mapeamento Objeto-Relacional Peewee ambas ferramentas da linguagem de programação Python. O objetivo não é demonstrar como isso é feito de forma pratica, mas sim apresentar os conceitos teóricos e princípios de funcionamentos destas tecnologias.

PALAVRAS-CHAVE: Micro-Framework Web, Mapeamento Objeto-Relacional, Flask, Peewee, Arquitetura de Software.

LISTA DE FIGURAS

Figura 1 - Fluxograma do padrão MVC	8
Figura 2 - Camadas de um ORM	11
Figura 3 - Como o ORM trabalha	12
Figura 4 - Site em Flask utilizando os mecanismos de objeto de requisição e rota do Werkzeug	15
Figura 5 - Mecanismo de Template do Jinja2.....	16
Figura 6 - Criando um modelo no Peewee	17
Figura 7 - Sincronizando o arquivo Python com o Banco de Dados	18

LISTA DE ABREVIATURAS

API (Application Programming Interface – Interface de Programação de Aplicativos)

CGI (Common Gateway Interface)

HTML (Hypertext Markup Language – Linguagem de Marcação de Hipertexto)

HTTP (Hypertext Transfer Protocol – Protocolo de Transferência de Hipertexto)

MVC (Model View Controller – Modelo Visão Controlador)

ORM (Object-relational mapping – Mapeamento objeto-relacional)

SGBD (Sistemas de Gestão de Base de Dados)

SQL (Structured Query Language – Linguagem de Consulta Estruturada)

URL (Uniform Resource Locator – Localizador Padrão de Recursos)

WSGI (Web Server Gateway Interface)

SUMÁRIO

INTRODUÇÃO AO DESENVOLVIMENTO APLICATIVOS WEB.....	6
1 ARQUITETURA DE UM SOFTWARE WEB.....	7
1.1 FRAMEWORK WEB.....	7
1.2 O PADRÃO DE PROJETO MVC.....	7
1.3 MICRO-FRAMEWORK WEB.....	8
1.4 SISTEMAS GERENCIADORES DE BANCO DE DADOS (SGBD) RELACIONAIS.....	9
1.5 LINGUAGEM DE CONSULTA ESTRUTURADA (SQL).....	10
1.6 MAPEAMENTO OBJETO-RELACIONAL (ORM)	10
2 UTILIZANDO O FASK E O PEEWEE PARA CRIAR UM FRAMEWORK WEB MVC	13
2.1 O MICRO-FRAMEWORK WEB FLASK.....	13
2.1.1 O TOOLKIT WSGI WERKZEUG	14
2.1.2 O MECANISMO DE TEMPLATE JINJA2	15
2.2 O ORM PEEWEE	16
2.2.1 O MECANISMO DE TEMPLATE JINJA2	17
CONCLUSÃO.....	19
REFERÊNCIAS.....	20

INTRODUÇÃO AO DESENVOLVIMENTO APLICATIVOS WEB

Atualmente existem diversas ferramentas no mercado para a programação de aplicativos Web, essas ferramentas se encontram em diversos estados de evolução, das mais primitivas as mais avançadas. Dentre elas os frameworks Web que utilizam o padrão de projeto MVC se destacam como ferramentas modernas e eficazes.

Entretanto para a utilização de um framework Web com o padrão MVC é necessário um nível considerável de conhecimento, o que dificulta a sua utilização pela maioria dos programadores iniciantes.

Para tornar a criação de uma aplicação Web com essas características mais fácil, nós iremos utilizar as ferramentas Flask e Peewee. Ambas são desenvolvidas na linguagem de programação Python que é conhecida por seu poder e facilidade de uso além de serem gratuitas e exigirem pouco poder computacional.

Iremos demonstrar nesse artigo como através do uso destas duas tecnologias nós podemos obter a mesma arquitetura dos grandes frameworks MVC com o uso de softwares minimalista com uma menor curva de aprendizado.

1 ARQUITETURA DE UM SOFTWARE WEB

A maneira como os componentes que formam um software é denominada arquitetura. Existem diversas arquiteturas possíveis para a criação de um software Web, porém os que utilizam os frameworks MVC se destacam como ferramentas modernas e eficazes.

1.1 FRAMEWORK WEB

De acordo com (JAQUES, 2016) “Framework é um conjunto de códigos abstratos e/ou genéricos, geralmente classes, desenvolvidos em alguma linguagem de programação, que relacionam-se entre si para disponibilizar funcionalidades específicas ao desenvolvedor de software. Em outras palavras, é como uma caixa de ferramentas, um kit que possui diversas funcionalidades devidamente implementadas, testadas e prontas para serem utilizadas na construção de softwares, poupando ao desenvolvedor tempo e trabalho na elaboração de operações básicas como acesso a banco de dados, sistema de templates, mapeamento de rotas e validação de dados”.

1.2 O PADRÃO DE PROJETO MVC

O padrão MVC, Modelo-Visão-Controlador (em inglês: Model-View-Controller) separa as responsabilidades do software em três camadas.

O Modelo (Model) consiste nos dados da aplicação, regras de negócios, lógica, funções, entidades e camada de acesso a dados.

A Visão (View) renderiza a resposta, pode ser qualquer saída de representação dos dados em aplicações web são (templates, html, css, javascript), em aplicações desktop (formulários, tabela, diagrama) enfim tudo relativo a visualizações.

Controlador (Controller) controla o fluxo de informações, ele faz a mediação da entrada, convertendo-a em comandos para o modelo ou visão, também coordena o fluxo de dados, recebe informações do usuário, e as requisições.

Figura 1 - Fluxograma do padrão MVC

Fonte: RAMOS ALLAN (2015).

1.3 MICRO-FRAMEWORK WEB

Frameworks Web comuns também chamados de Frameworks Full-Stack são estruturas que buscam fornecer soluções para as diversas tarefas presentes em uma aplicação Web, conforme foi ocorrendo a evolução dessas aplicações alguns destes Frameworks foram incorporando cada vez mais soluções, crescendo assim em complexidade.

Por definição um Framework é um conjunto de funcionalidades para uma tarefa específica e para funcionar como tal este deve possuir um alto nível de integração, sendo assim ao mesmo tempo que um Framework prove soluções para problemas de desenvolvimento comuns ele também exige para funcionar em contra partida um nível considerável de conhecimento do programador, além de reduzir a flexibilidade das escolhas do programador pois esse deve seguir

algumas convenções adotadas pelo Framework em questão.

Buscando manter os benefícios dos Frameworks Full-Stack mas fornecendo uma maior flexibilidade e simplicidade surgiram os Micro-Frameworks Web.

Conforme (ANDRADE, 2019), “Os Micro-Frameworks são Frameworks modularizados que possuem uma estrutura inicial muito mais simples quando comparado a um Framework convencional. Pense em um Micro-Framework como uma peça de LEGO. Inicialmente, um projeto criado com o micro-framework possui apenas o básico para funcionar, (normalmente, sistema de rotas), porém, ao decorrer do projeto, podem haver necessidades para utilização de outros recursos como conexão de banco de dados, sistemas de templates, envio de email, etc. A partir desta necessidade, novas bibliotecas são “encaixadas” no projeto, como uma estrutura de um LEGO”.

1.4 SISTEMAS GERENCIADORES DE BANCO DE DADOS (SGBD) RELACIONAIS

Os Sistemas Gerenciadores de Banco De Dados surgiram durante os anos 60, a principal motivação foi a busca por maneiras mais rápidas e baratas de se gerenciar grandes quantidades de informação.

Como nos conta (PROFISSAODBA, 2019), “Igualmente a muitas tecnologias na computação industrial, os fundamentos de bancos de dados relacionais surgiram na empresa IBM, nas décadas de 1960 e 1970, através de pesquisas de funções de automação de escritório. Foi durante um período da história na qual empresas descobriram que estava muito custoso empregar um número grande de pessoas para fazer trabalhos como armazenar e indexar (organizar) arquivos. Por este motivo, valia a pena os esforços e investimentos em pesquisar um meio mais barato e ter uma solução mecânica eficiente”.

Diversos modelos de SGBDs foram desenvolvidos, entretanto um modelo se tornou o mais popular nas décadas seguinte, o modelo relacional. Nesse sistema os dados são organizados em tabelas denominadas entidades, cada tabela possui uma estrutura composta de campos com tipos específicos de dados, alguns desses campos podem conter chaves que permitem relacionar os dados de diversas tabelas.

Como visto em (PROFISSAODBA, 2019) “Em 1970 um pesquisador da IBM – Ted Codd – publicou o primeiro artigo sobre bancos de dados relacionais. Este artigo tratava sobre o uso de cálculo e álgebra relacional para permitir que usuários não técnicos armazenassem e recuperassem grande quantidade de informações. Codd visionava um sistema onde o usuário seria capaz de acessar as informações através de comandos em inglês, onde as informações estariam armazenadas em tabelas”.

Para que o Banco de Dados Relacional seja eficaz, devemos seguir algumas regras de modelagem que garantem a integridade do sistema, chamamos essas regras de normalização.

“Normalização de dados é o processo formal e passo a passo que examina os atributos de uma entidade, com o objetivo de evitar anomalias observadas na inclusão, exclusão e alteração de registros”, (PROFESSOR DIGITAL, 2019).

1.5 LINGUAGEM DE CONSULTA ESTRUTURADA (SQL)

A manipulação dos SGBDs relacionais é feita através da Linguagem de Consulta Estruturada (SQL). Ela fornece uma interface através do qual utilizando de uma linguagem de pesquisa declarativa podemos estruturar um esquema de banco de dados. Nele é possível executar pesquisas, inserir, alterar e apagar dados além de operações mais complexas como transações.

Segundo (ALVES, 2013), “A ‘Linguagem Estruturada de Consultas’ (SQL, traduzida para o português) é utilizada para interagir com o SGBD e executar várias tarefas como inserir e alterar registros, criar objetos no banco de dados, gerenciar usuário, consultar informações, controlar transações, etc. Todas as operações realizadas no banco de dados podem ser solicitadas ao SGBD utilizando esta linguagem”.

1.6 MAPEAMENTO OBJETO-RELACIONAL (ORM)

Para ao integrar um SGBD relacional a um sistema o programador deve ter domínio em modelagem de dados além de possuir conhecimento da linguagem SQL. Entretanto como dito anteriormente esses conhecimentos fogem do escopo das linguagens de programação que se focam em sua maioria

no uso do paradigma da programação orientada a objetos.

“Mapeamento objeto-relacional (ou ORM, do inglês: Object-relational mapping) é uma técnica de desenvolvimento utilizada para reduzir a impedância da programação orientada aos objetos utilizando bancos de dados relacionais. As tabelas do banco de dados são representadas através de classes e os registros de cada tabela são representados como instâncias das classes correspondentes. Com esta técnica, o programador não precisa se preocupar com os comandos em linguagem SQL; ele irá usar uma interface de programação simples que faz todo o trabalho de persistência. Não é necessária uma correspondência direta entre as tabelas de dados e as classes do programa. A relação entre as tabelas onde originam os dados e o objeto que os disponibiliza é configurada pelo programador, isolando o código do programa das alterações à organização dos dados nas tabelas do banco de dados”. (WIKIPEDIA, 2019)

Figura 2 - Camadas de um ORM

Fonte: DEVMEDIA (2019)

Nesse contexto surgem os softwares Mapeamento Objeto-Relacional, com ele os programadores fazem a modelagem através da definição de classes e o ORM mapeia essas classes em comandos SQL que são enviados ao Banco de Dados.

Como nos conta (DEV MEDIA, 2019) “No mundo relacional prevalecem princípios matemáticos com a finalidade de armazenar e gerenciar corretamente os dados, de forma segura e se trabalha com a linguagem SQL que é utilizada

para dizer o banco de dados ‘O QUE?’ fazer e não como fazer. Já no mundo orientado a objetos trabalhamos com classes e métodos, ou seja, trabalhamos fundamentados na engenharia de software e seus princípios que nos dizem “COMO” fazer. O ORM é justamente, a ponte entre estes dois mundos, ou seja, é ele quem vai permitir que você armazene os seus objetos no banco de dados”.

Além disso o ORM gera métodos responsáveis pela interação entre o software e o Banco de Dados, esses métodos são os mesmos independente SGBD utilizado, o que torna o procedimento de migração simples.

Como mostrado em (DEV MEDIA, 2019) “... nos traz uma ideia de como o ORM trabalha. Ele faz o mapeamento da sua classe para o banco de dados e cada ORM tem suas particularidades para gerar o SQL referente a inserção do objeto que corresponde a uma tabela no banco de dados e realizar a operação. Utilizando um ORM, também se ganha produtividade, pois deixa-se de escrever os comandos SQL para deixar que o próprio ORM, faça isto por você”.

Figura 3 - Como o ORM trabalha

Fonte: DEV MEDIA (2019)

2 UTILIZANDO O FASK E O PEEWEE PARA CRIAR UM FRAMEWORK WEB MVC

Neste capítulo será explicado qual é o princípio do Flask e do Peewee, através destes princípios é possível criar uma arquitetura MVC utilizando os referidos softwares.

2.1 O MICRO-FRAMEWORK WEB FLASK

O Flask é um Micro-Framework Web para a linguagem de programação Python, ele foi construído com o objetivo de ser simples e flexível, sendo de menor complexidade do que outros Frameworks Full-Stack em Python como o Django.

Segundo (DEVFURIA, 2019), “Flask é um micro-framework para Python, segundo a documentação... “Micro” não significa que a sua aplicação web inteira tenha que se encaixar em um único arquivo Python, embora isso seja totalmente possível. Também não quer dizer que falte funcionalidade ao Flask. “Micro” em microframework significa que o Flask procura manter o núcleo simples, mas extensível. O Flask não toma muitas decisões por você, tipo, que banco de dados usar. Por outro lado, é fácil mudar as coisas que já decidiu, tipo, o mecanismo de templates. Todo o resto fica por sua conta decidir, de modo que você pode obter do Flask tudo o que precisa e dispensar tudo o que não quer”.

O Flask é baseado no Werkzeug uma biblioteca para WSGI (a especificação padrão do Python para a comunicação entre aplicações web e servidores HTTP) e no Jinja2 (um mecanismo de renderização de templates), mas diferente do Django ele não conta com um ORM padrão.

Como mostra (DEVFURIA, 2019), “Por padrão, o Flask não inclui uma camada de abstração de banco de dados, validação de formulário, ou qualquer outra coisa para a qual já existam diferentes bibliotecas. Em vez disso, suporta extensões para adicionar uma determinada funcionalidade à sua aplicação como se ela tivesse sido implementada no Flask em si. Muitas extensões fornecem integração de banco de dados, validação de formulários, manipulação de upload, diversas tecnologias de autenticação abertas, entre outras coisas. O Flask pode ser “micro”, mas está pronto para uso em produção, preenchendo uma gama variada de necessidades”.

2.1.1 O TOOLKIT WSGI WERKZEUG

O Werkzeug é um toolkit para a especificação do Python WSGI Web Server Gateway Interface (em português Interface de Porta de Entrada do Servidor Web), trata-se de uma forma para servidores conversarem com frameworks, e vice-versa.

De acordo com (WIKIPEDIA, 2019), “Frameworks de aplicação web em Python têm sido um problema para novos usuários de Python pois a escolha do framework web pode limitar a escolha de servidores web úteis, e vice-versa. Aplicações em Python eram frequentemente projetadas para apenas um dos servidores CGI, FastCGI, mod_python ou alguma outra API personalizada de um servidor web específico. O WSGI foi criado como uma interface de baixo-nível entre servidores web e aplicações web ou frameworks para promover uma base comum para o desenvolvimento de aplicações web portáteis”.

As funcionalidades do Werkzeug incluem:

- Um depurador interativo
- Um objeto para lidar com requisições incluindo:
 - cabeçalhos
 - query strings
 - dados de formulário
 - arquivos
 - cookies
- Um objeto para lidar com respostas e manipular dados de streaming
- Um sistema de roteamento para lidar com recursos e endpoints de URLs além capturar variáveis de URLs
- Um utilitário HTTP para:
 - controle de cache
 - agentes de usuário
 - cookies
 - arquivos
- Um servidor de desenvolvimento WSGI local
- Um cliente para simular solicitações HTTP

Figura 4 - Site em Flask utilizando os mecanismos de objeto de requisição e rota do Werkzeug

```
from flask import Flask, escape, request

app = Flask(__name__)

@app.route('/')
def hello():
    name = request.args.get("name", "World")
    return f'Hello, {escape(name)}!'
```

Fonte: THE PALLETS PROJECTS (2019)

2.1.2 O MECANISMO DE TEMPLATE JINJA2

O Jinja2 é um mecanismo de template (modelo) em Python, ele permite separar a lógica do modelo do site de uma forma semelhante a abordagem de modelo citada anteriormente.

Segundo (OLIVEIRA, 2019), “Jinja é um mecanismo de templates para Python. Ele facilita a tarefa de usar HTML dentro de nossas aplicações Python, tem completo suporte unicode, e para aplicativos que necessitam de maior segurança, adiciona execução em área restrita e escape automático”.

Ele torna a sintaxe da aplicação Web mais clara pois assim como na abordagem de modelo a maior parte do código é composta de HTML e nos locais onde é necessário que conteúdo seja inserido dinamicamente nos utilizamos tags especiais que podem ter diversas funcionalidades, de passagem de variáveis, filtros personalizados até estruturas de decisão e repetição.

De acordo com (SILVA, 2014), “O HTML embebido no código gera uma grande confusão e torna difícil fazer manutenção do mesmo, o melhor é usar um sistema de templates, onde o HTML é mantido em arquivos separados com uma sintaxe especial para a introdução de alguma lógica e manipulação dos dados vindos das aplicações”.

A diferença em relação a abordagem de modelo é que essa tem um poder programático limitado poucas instruções, já com o Jinja2 essa limitação não existe, pois, os dados são processados pelo Flask e após isso as variáveis

são passadas para o Jinja2 que age como a camada de visão do modelo MVC renderizando a resposta no formato desejado. Além disso o Jinja2 possui mecanismos de herança e composição que tornam mais simples a criação de sites.

Como afirma (SILVA, 2014), “O Jinja2 permite a herança de templates, isto é, podemos ter um template base onde fazemos o design geral da página, podemos incluir um cabeçalho e um rodapé, um menu horizontal ou vertical que depois poderá ser reutilizado por todos os templates que herdem do template base.html”.

Figura 5 - Mecanismo de Template do Jinja2

```
{% extends "layout.html" %}
{% block body %}
<ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
</ul>
{% endblock %}
```

Fonte: THE PALLETS PROJECTS (2019)

2.2 O ORM PEEWEE

O Peewee é um ORM para a linguagem de programação Python, ele foi construído com o objetivo de ser simples e pequeno, sendo de menor complexidade do que outros ORMs em Python como o SQLAlchemy.

Segundo (STUTTGART, 2017), “O Peewee é destinado a projetos de pequeno/médio porte, se destacando pela simplicidade quando comparado a outros ORM mais conhecidos, como o SQLAlchemy. Uma analogia utilizada pelo autor da API e que acho muito interessante é que Peewee está para o SQLAlchemy assim como SQLite está para o PostgreSQL”.

Mesmo sendo simples ele ainda possui suporte aos Bancos de Dados SQLite, MySQL e PostgreSQL cobrindo assim os principais SGBDs relacionais open source do mercado.

(STUTTGART, 2017), Nos diz que, “Em relação aos recursos por ele oferecidos, podemos citar que ele possui suporte nativo a SQLite, PostgreSQL e MySQL, embora seja necessário a instalação de drivers para utilizá-lo com PostgreSQL e MySQL e suporta tanto Python 2.6+ quanto Python 3.4+”.

O Peewee pode ser facilmente instalado através do gerenciador de pacotes do Python o pip com o comando `pip install peewee`.

2.2.1 O MECANISMO DE TEMPLATE JINJA2

Como na maioria dos ORM's o procedimento para se criar um modelo para ser utilizado no Peewee começa com a definição de uma classe em Python como pode ser visto na imagem a seguir.

Figura 6 - Criando um modelo no Peewee

```
from peewee import *

db = SqliteDatabase('pessoa.db')

class Pessoa(Model):
    nome = CharField()
    sobrenome = CharField()
    data_nascimento = DateField()

    class Meta:
        database = db
```

Fonte: Própria

Iniciamos o nosso projeto importando toda biblioteca Peewee.

Em seguida definimos qual o Banco de Dados iremos utilizar, neste caso o SQLite, uma observação importante e que para mudar o SGBD que a aplicação utiliza, basta alterar o método de conexão na variável `db`. Deve se salientar que cada Banco de Dados possui suas próprias peculiaridades e a definição do

objeto db varia conforme o SGBD utilizado. Essa configuração equivale a criação de um Banco de Dados em um SGBD (CREATE DATABASE).

Após essas configurações iniciais nos partimos para a criação das entidades (tabelas) modelando as suas classes, esse procedimento é o equivalente ao comando do SQL CREATE TABLE. Para isso nos utilizamos a herança da orientação a objeto através da classe do Peewee Model.

Os atributos são objetos herdados das classes do Peewee, eles equivalem aos tipos dos campos do SQL, e podem ser customizados através de seus atributos.

Após isso temos a meta classe (Meta) que ao ser herdada define o método de conexão através do objeto db, isso vincula a tabela ao seu Banco de Dados, equivalente ao USE DATABASE do SQL.

Após as classes terem sido modeladas devemos executar o método create_tables(), ele sincroniza o modelo em Python com o Banco de Dados escolhido, podemos fazer uma analogia com o procedimento de restaurar um backup de um arquivo SQL.

Figura 7 - Sincronizando o arquivo Python com o Banco de Dados

```
if __name__ == '__main__':  
    # Quando esse arquivo for executado como main  
    # será criada as tabelas de banco de dados  
    create_tables()
```

Fonte: Própria

Uma maneira comumente adotada para isso é definir o método create_tables() dentro de um if __name__ == '__main__', o que garante que ele será chamado somente quando o arquivo que o contém seja diretamente executado.

CONCLUSÃO

O objetivo deste artigo não é ser um tutorial explicando a funcionalidade de cada ferramenta apresentada, e sim, explicar os princípios arquiteturais através do qual é possível montar um aplicativo Web MVC utilizando o Flask e o Peewee.

Levando em consideração tal objetivo, é possível constatar que foi completamente alcançado nesse artigo inicial. Outros artigos que apresentarão como esse processo de integração são executados de forma pratica não apresentados nesse serão escritos futuramente, complementando o conhecimento e criando um caminho inicial no desenvolvimento do conhecimento necessário para a criação aplicativos Web utilizando o padrão de projeto MVC com essas tecnologias.

REFERÊNCIAS

ALVES GUSTAVO FURTADO DE OLIVEIRA – Você precisa saber o que é SQL - URL (<https://dicasdeprogramacao.com.br/o-que-e-sql/>) visto em 18/09/2019 publicado em 23/04/2013.

ANDRADE ANA PAULA DE – O que é um Micro-Framework - URL (<https://tableless.com.br/mvc-afinal-e-o-que/>) visto em 18/09/2019 publicado em 06/05/2019.

DEVFURIA – Python Flask Introdução - URL (<http://www.devfuria.com.br/python/flask/>) acessado em 18/09/2019.

DEVMEDIA - ORM: Object Relational Mapper – URL (<https://www.devmedia.com.br/orm-object-relational-mapper/19056>) acessado em 18/09/2019.

JAIQUES RAFAEL – O que é um Framework? Para que serve? - URL (<http://www.phpit.com.br/artigos/o-que-e-um-framework.php>) visto em 18/09/2019 publicado em 07/01/2016.

OLIVEIRA KAMILA DE FÁTIMA SANTOS – O que é Flask e porque ele é considerado um microframework - URL (<https://imasters.com.br/desenvolvimento/conhecendo-o-jinja2-um-mecanismo-para-templates-no-flask>) visto em 18/09/2019 publicado em 15/07/2019.

PROFESSOR DIGITAL - Normalização de dados e as formas normais – URL (<https://www.luis.blog.br/normalizacao-de-dados-e-as-formas-normais.html>) acessado em 18/09/2019.

PROFISSAODBA – Evolução dos Bancos de Dados e SGBDs - URL (<https://profissaodba.wordpress.com/historico-dos-bancos-de-dados/>) acessado em 18/09/2019.

RAMOS ALLAN – O que é MVC - URL (<https://tableless.com.br/mvc-afinal-e-o-que/>) visto em 18/09/2019 publicado em 26/02/2015.

SILVA L.F.A.D., COSTA P.R.D.S.A, CÂNDIDO R.D. Estudo Sobre o Software de Mapeamento Objeto-Relacional Peewee. Mogi Guaçu, 2019.

SILVA L.F.A.D., COSTA P.R.D.S.A, CÂNDIDO R.D. Estudo sobre o Software de Micro-Framework Web Flask. Mogi Guaçu, 2019.

SILVA SÉRGIO – Tutorial, Blog em Python - Parte 4: Templates - URL
(<http://www.sergiofasilva.com/post/tutorial-blog-em-python-parte-4-templates>)
visto em 18/09/2019 publicado em 23/01/2014.

STUTTGART MICHELL – Peewee: Um ORM Python minimalista - URL
(<http://pythonclub.com.br/peewee-um-orm-python-minimalista.html>) visto em
18/09/2019 publicado em 20/07/2017.

THE PALLETS PROJECTS – Flask - URL
(<https://palletsprojects.com/p/flask/>) acessado em 18/09/2019.

THE PALLETS PROJECTS – Jinja - URL
(<https://palletsprojects.com/p/jinja/>) acessado em 18/09/2019.

WIKIPEDIA – Mapeamento objeto-relacional - URL
(https://pt.wikipedia.org/wiki/Mapeamento_objeto-relacional) acessado em
18/09/2019.

WIKIPEDIA – Web Server Gateway Interface - URL
(https://pt.wikipedia.org/wiki/Web_Server_Gateway_Interface) acessado em
18/09/2019.