Relatório de Desenvolvimento - Hands-On

João Choma Neto NUSP: 10633606 [1]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC)
Programa de Ciências da Computação e Matemática Computacional
Disciplina SSC 5904 - 2018 — Reúso de Software
Profa. Dra. Rosana T. Vaccare Braga

joaochoma@usp.br

1. Introdução

Este relatório tem como objetivo apresentar o processo de implementação de um serviço REST, bem como, a utilização dos recursos disponíveis. Para isso foi utilizada uma plataforma de reúso de software chamada Django [Django 2018] e Django REST Framework [Django REST 2018]. Nas próximas seções serão apresentados os conceitos necessários para compreender o processo de implementação de um serviço REST.

2. Django REST Framework

REST (REpresentational State Transfer) é um estilo arquitetural que fornece padrões entre sistemas na Web com propósito de facilitar a comunicação. Os sistemas compatíveis com REST, geralmente chamados de sistemas RESTful e são caracterizados pela forma como separam as responsabilidades do cliente e do servidor [Codeacademy 2018].

O estilo arquitetural REST permite que a implementação do cliente e a implementação do servidor sejam construídas de forma independente. Desse modo, o código do cliente pode ser alterado a qualquer momento sem afetar a operação do servidor, e o código do servidor pode ser alterado sem afetar a operação do cliente. Esta flexibilidade é possibilitada pelo padrão de mensagens existente entre cliente e servidor. Usando uma interface REST, diferentes clientes executam as mesmas ações e recebem as mesmas respostas [Codeacademy 2018].

A arquitetura REST requer que um cliente faça uma solicitação ao servidor para recuperar ou modificar dados. Uma solicitação geralmente consiste em: (i) um verbo HTTP, que define o tipo de operação a ser executada, (ii) um cabeçalho, que permite ao cliente passar informações sobre o pedido, (iii) um caminho para um recurso, (iv) um corpo de mensagem opcional contendo dados [Codeacademy 2018].

Para interagir com um sistema REST existem 4 verbos HTTP básicos [Codeacademy 2018]:

- GET: recupera um dado específico (por id) ou uma coleção de dados;
- POST: cria um novo dado;
- PUT: atualiza um dado específico (por id);
- DELETE: remove um dado específico por id.

Dentre as plataformas que utilizam a arquitetura REST existe o framework Django REST, composto por um kit de ferramentas poderoso e flexível para criar APIs na Web. O

framework oferece uma forma simplificada de utilização da arquitetura REST disponibilizando diversos serviços de uma API Web. Além disso, viabiliza políticas de autenticação garantindo concistência do ambiente utilizado [Django REST 2018].

Já o Django é um web framework para aplicações Web gratuito e de código aberto, escrito em Python. Um web framework é um conjunto de componentes que ajuda você a desenvolver sites de forma mais rápida e fácil [Django 2018].

O Django REST utiliza a linguagem de comunicação JSON que representa um padrão de mensagens. JSON (JavaScript Object Notation) é um formato de troca de dados naturalmente interpretável por humanos e gerado por máquinas. O formato de texto é baseado em um subconjunto da Linguagem de Programação JavaScript, Padrão ECMA-262 3ª Edição - Dezembro de 1999. A linguagem JSON é completamente independente de linguagem, mas usa convenções que são familiares aos desenvolvedores das linguagens C , C ++, C , Java, JavaScript, Perl, Python e muitos outros [JSON 2018].

O objetivo do trabalho é alcançado utilizando-se do Django REST Framework para implementar uma simples aplicação que utiliza os verbos básicos GET e POST do serviço REST. As tarefas executadas evidenciam a comunicação existente entre cliente e servidor durante a solicitação e recebimento de dados. Na sequência é apresentado o processo de implementação do serviço web.

3. Implementação

Foi utilizado o tutorial e exemplo disponíveis no site oficial do Django REST Framework, disponível em [Django REST 2018]. Inicialmente foram realizadas as configurações do servidor REST com a instalação das dependências necessárias para o funcionamento dos serviços web. O Django REST Framework faz uso dos recursos disponíveis pelas linguagens Python (2.7, 3.4, 3.5, 3.6, 3.7) e Django (1.11, 2.0, 2.1).

Para facilitar o desenvolvimento do projeto foi utilizado Virtualenv [Virtualenv 2018]. Essa ferramenta é uma ambiente virtual de desenvolvimento que tem seus próprios diretórios de instalação não compartilhando bibliotecas com outros ambientes virtuais, o que evita conflitos durante a instalação de diversa bibliotecas.

Uma vez instanciada a Vistualenv pelo comando $.venv/bin/activate$ foram realizados os seguintes comandos para criação e configuração do serviço REST. Os comandos realizam toda a configuração do ambiente Django REST instalando todas as dependências necessárias.

pip install django

pip install djangoRESTframework

pip install markdown

pip install django-filter

Uma vez configurado o servidor REST foi possível criar o projeto da API responsável por oferecer os serviços. Desta forma, foram utilizados os seguintes comandos:

django-admin startproject api - criação do projeto da API;

python manage.py starapp - instanciação do projeto;

python manage.py startup core - criação do backend do projeto;

Com as principais classes criadas se faz necessário realizar o registro da aplicação core e REST_framework nas urls.py. Na sequência, com os comandos a baixo, foi criado o banco de dados responsável por armazenar os dados da aplicação.

python manage.py migrate

python manege.py createsuperuser

Ao termino das configurações anteriores já foi possível executar o serviço REST com o seguinte comando: python manege.py runserver. E, para testar o seu funcionamento, buscou-se a url: localhost:8000, como pode ser visto na Figura 1.

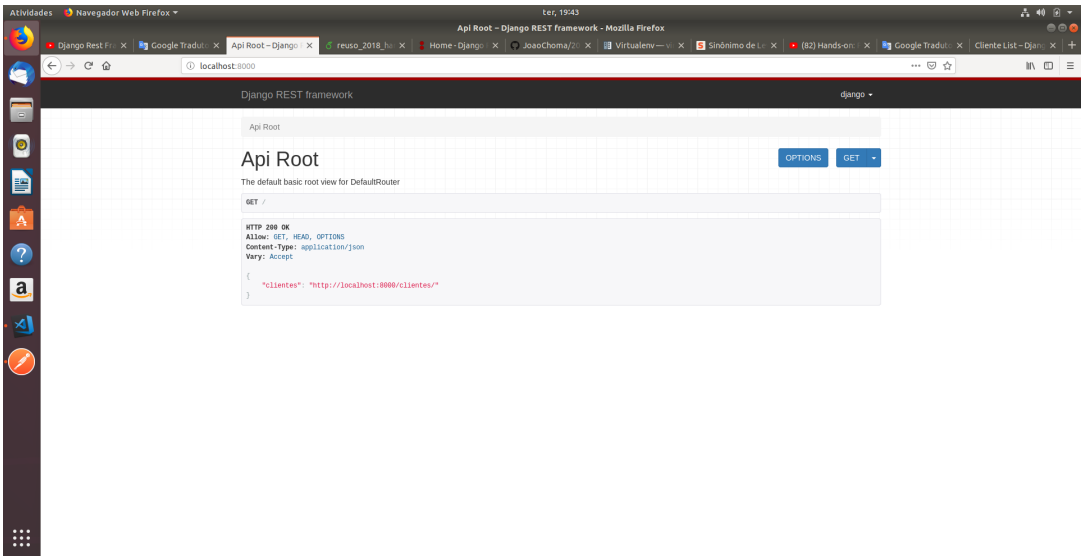

Figura 1. Resultado da URL: localhost:8000.

A ideia do serviço REST é permitir que a partir de uma url e um verbo de requisição, seja possível que o cliente solicite uma tarefa ao servidor e, como resposta, o servidor retorne os dados solicitados por meio de uma interface JSON. Para isso foi codificada uma classe de modelo responsável por implementar um objeto Cliente com duas informações: nome e dados. O código da classe está apresentado no Código 1.

Código 1: Implementação classe de Modelo Cliente.

```
from django.db import models

class Cliente(models.Model):
    nome = models.CharField(max_length=50)
    dados = models.CharField(max_length=50)

def __str__(self):
    return self.nome
```

A partir da implementação da classe de modelo Cliente é possível realizar os serviços oferecidos pelos verbos GET e POST. As requisições e os respectivos resultados estão apresentados no Código 2.

Código 2: Resultados das solicitações GET e POST.

```
GET http://localhost:8000/clientes/

    {
        "id": 1,
        "nome": "Jo o",
        "dados": "Primeiro - 1"
    },
    {
        "id": 2,
        "nome": "Joao2",
        "dados": "Segundo"
    },
    {
        "id": 3,
        "nome": "Joao3",
        "dados": "Terceiro"
    }

POST http://localhost:8000/clientes/?nome=Joao4&dados=Quarto

    {
        "id": 4,
        "nome": "Joao4",
        "dados": "Quarto"
    }
```

Todos os códigos gerados durante o processo de implementação do serviço REST estão disponíveis no link: $https$: $//github.com/JoaoChoma/2018\_reuso\_handson\_services/$ e no arquivo $handson.zip$ que acompanha o presente texto.

3.1. Dificuldades e Aprendizado

Em razão de ser um assunto novo para o autor houveram diversas dificuldades que tiveram de ser sanadas para alcançar o objetivo deste trabalho. Por este motivo, nesta seção serão apresentadas as principais dificuldades encontradas no processo de implementação do servidor REST e, as soluções encontradas.

Inicialmente houve a necessidade de realizar um estudo transversal sobre as principais característica de uma arquitetura REST. Posteriormente foi realizada uma pesquisa de quais frameworks poderiam ser utilizados para auxiliar na implementação do servidor de serviços.

O framework Django REST foi identificado como promissor por ser baseado na linguagem Python que apresenta grande suporte didático na web, bem como, uma sintaxe simplificada e familiar ao autor. Não obstante, o autor desconhecia a existencia do framework Django e por essa razão houve, novamente, a necessidade de realizar um estudo transversal sobre o framework Django.

Como aprendizado adquirido fica o aperfeiçoamento da linguagem Python e o conhecimento de novas tecnologias de reúso (Django, Django REST), a linguagem JSON e ferramenta de virtualização Virtualenv.

Referências

Codeacademy (2018). What is rest? disponível em: https://www.codecademy.com/articles/what-is-rest.

Django (2018). Django disponível em: https://www.djangoproject.com/.

Django REST, F. (2018). Django rest framework. disponível em: https://www.django-rest-framework.org/.

JSON, I. (2018). Json disponível em: https://www.json.org/.

Virtualenv (2018). Virtualenv. disponível em: https://virtualenv.pypa.io/en/latest/.