Octofiles Documentation

Release 1.0.0

Hackultura

March 14, 2016

Contents

1 Configuração **3**
1.1 Desenvolvimento 3

2 Guia de Estilo **5**
2.1 Links úteis 5

3 Funcionalidades **7**
3.1 Administradores 7
3.2 Gerenciamento de Sistemas 7

4 API **9**
4.1 Autenticação 9
4.2 Documentos 9

5 Indices and tables **13**

HTTP Routing Table **15**

Bem-vindo a documentação do projeto Octofiles. Esse foi projeto foi idealizado para servir outros sistemas da plataforma SISCULT, oferecendo um ambiente de envio de arquivos e documentações em que precisam de um controle e também de centralização para futuras consultas entre as funcionalidades. Ele segue de forma superficial o princípio do Amazon S3 na forma de persistência, controle e permissão dos mesmos, mas com mudanças para o modelo de negócio que a Secretaria de Cultura necessita.

Abaixo segue as informações acerca do seu funcionamento e desenvolvimento:

CHAPTER 1

Configuração

Vamos orientar a configurar o projeto em um ambiente local, de duas formas: localmente para desenvolvimento e implantação. Lembrando que para todos os casos, precisa ter os seguintes requisitos mínimos:

- Python 2.7 ou 3.2+
- virtualenv
- Git
- PostgreSQL
- Redis

1.1 Desenvolvimento

Acesse o repositório do projeto e efetue o checkout na sua máquina:

```
$ git clone https://github.com/gilsondev/octofiles.git
```

Com o projeto no seu computador, vamos criar o ambiente virtual e instalar as dependências:

```
$ cd octofiles
$ virtualenv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
```

Instalado, vamos iniciar o servidor local:

```
$ python manage.py runserver
```

CHAPTER 2

Guia de Estilo

Note: Essa seção está em contínua mudança

Para contribuir com o projeto, é interessante que o desenvolvedor tenha em mente algumas das ferramentas e boas práticas no desenvolvimento:

- Usar ferramenta para verificação do código no padrão PEP8;
- Sempre testar as implementações feitas;
- A endentação do código segue o padrão de 4 espaços;
- A cada nova mudança de feature, **sempre** atualize essa documentação;

2.1 Links úteis

Segue alguns links para enriquecer esse guia e entender melhor os padrões usados no projeto:

- The Hitchhiker's Guide to Python

CHAPTER 3

Funcionalidades

O projeto Octofiles, como definido na introdução, ele oferece um serviço de armazenamento de documentos dos entes artísticos, como também alguns metadados de cada sistema, na necessidade de futuras consultas. Assim foi pensado nas seguintes funcionalidades.

3.1 Administradores

Aqui os administradores do sistema, terão a opção de manter os seus dados e de outros. Além disso ele terá o papel importante no gerenciamento dos sistemas autorizados a usar o serviço.

3.2 Gerenciamento de Sistemas

Sendo um Administrador, o usuário pode cadastrar os sistemas que vão ter acesso ao serviço do Octofiles. Nesse cadastro será gerado um *APP_ID* e um *SECRET_KEY*, em que o sistema consumidor precisará enviar a este, para gerar um token de autenticação e assim ser autorizado para enviar arquivos para upload como consulta e download. Além disso, o usuário poderá revogar o acesse de quem precisar.

CHAPTER 4

API

Todo

Usar a opção de autohttp.flask para atualizar automaticamente essa seção

Nessa seção vamos definir toda a documentação da API do projeto.

4.1 Autenticação

Em breve

4.2 Documentos

Nesse recurso, é aonde fica disponibilizado e persistido os documentos do projeto. Temos as seguintes URIs:

GET **/api/v1/** (**string:** *bucket*) **/documentos/**
Lista os documentos de um determinado sistema a partir do seu *bucket*.

Exemplo de requisição:

```
GET /api/v1/procult/documentos/ HTTP/1.1
Authorization: Token 1af538baa9045a84c0e889f672baf83ff24
Host: octofiles.cultura.df.gov.br
```

Parameters

- **bucket** – Nome do bucket do sistema.

Request Headers

- Authorization – Token para autenticação

Exemplo de resposta:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

[
```

```
        {
            "document_url": "http://octofiles.cultura.df.gov.br/api/v1/procult/propostas/alsd01lkasd
            "self_url": "http://octofiles.cultura.df.gov.br/api/v1/procult/ask12312309aslk1230",
            "metadata": {
              "name": "Fotos do Show",
              "type": "pdf",
              "size": 1200000
            },
            "owner_uid": "asdk1239asdlk12309as",
            "mode": "private"
        }
    ]
```

Response Headers

- Content-Type – Formato JSON

Status Codes

- 200 OK – Requisição aceita com sucesso
- 400 Bad Request – Problema na requisição enviada
- 500 Internal Server Error – Erro na consulta dos documentos

POST /api/v1/ (**string:** *bucket*) **/documentos/**

Envia um novo documento para armazenamento, a partir do seu *bucket*.

Parameters

- **bucket** – Nome do bucket do sistema

Form Parameters

- **name** – Nome do Arquivo
- **path** – Caminho do arquivo a ser salvo
- **file** – Binário do arquivo para upload
- **mode** – Modo de visualização: *public* ou *private*
- **owner** – Se o modo de visualização for *private*, o identificador único do dono do arquivo

Request Headers

- Authorization – Token para autenticação

Response Headers

- Content-Type – Formato JSON
- Location – URI do arquivo recém criado

Status Codes

- 201 Created – Documento criado com sucesso
- 400 Bad Request – Erro na montagem da requisição para upload do arquivo
- 500 Internal Server Error – Erro durante o upload do arquivo

GET /api/v1/ (**string:** *bucket*) **/**

string: *uid* Retorna informações do documento pesquisado a partir do seu *bucket* e o *uid* do arquivo.

Exemplo de requisição:

```
GET /api/v1/procult/ask12312309aslk1230 HTTP/1.1
Authorization: Token 1af538baa9045a84c0e889f672baf83ff24
Host: octofiles.cultura.df.gov.br
```

Parameters

- **bucket** – Nome do bucket do sistema.
- **uid** – Identificador único (UID) do arquivo salvo

Request Headers

- Authorization – Token para autenticação

Exemplo de resposta:

```
HTTP/1.1 200 OK
Vary: Accept
Content-Type: application/json

{
    "document_url": "http://octofiles.cultura.df.gov.br/api/v1/procult/propostas/alsd011kasd9123
    "self_url": "http://octofiles.cultura.df.gov.br/api/v1/procult/ask12312309aslk1230",
    "metadata": {
      "name": "Fotos do Show",
      "type": "pdf",
      "size": 1200000
    },
    "owner_uid": "asdk1239asdlk12309as",
    "mode": "private"
}
```

Response Headers

- Content-Type – Formato JSON

Status Codes

- 200 OK – Requisição aceita com sucesso
- 400 Bad Request – Problema na requisição enviada
- 500 Internal Server Error – Erro na consulta do documento

PUT /api/v1/ (**string:** *bucket*) **/**
string: *uid* Atualiza as informações do documento selecionado a partir do seu *bucket* e o *uid* do arquivo.

Parameters

- **bucket** – Nome do bucket do sistema
- **uid** – Identificador único (UID) do arquivo salvo

Form Parameters

- **name** – Nome do Arquivo
- **mode** – Modo de visualização: *public* ou *private*
- **owner** – Se o modo de visualização for *private*, o identificador único do dono do arquivo

Request Headers

- Authorization – Token para autenticação

Response Headers

- Content-Type – Formato JSON
- Location – URI do arquivo recém criado

Status Codes

- 200 OK – Documento atualizado com sucesso
- 400 Bad Request – Erro na montagem da requisição para atualização do arquivo
- 500 Internal Server Error – Erro durante atualização do arquivo

DELETE /api/v1/ (**string:** *bucket*) **/**
string: *uid* Remove o documento selecionado a partir do seu *bucket* e o *uid* do arquivo.

Parameters

- **bucket** – Nome do bucket do sistema
- **uid** – Identificador único (UID) do arquivo salvo

Request Headers

- Authorization – Token para autenticação

Status Codes

- 204 No Content – Documento excluido com sucesso
- 400 Bad Request – Erro na montagem da requisição para remoção do arquivo
- 500 Internal Server Error – Erro durante remoção do arquivo

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

/api

```
GET /api/v1/(string:bucket)/(string:uid),
       10
GET /api/v1/(string:bucket)/documentos/,
       9
POST /api/v1/(string:bucket)/documentos/,
       10
PUT /api/v1/(string:bucket)/(string:uid),
       11
DELETE /api/v1/(string:bucket)/(string:uid),
       12
```