EducaUTF OLC

Documentação completa

Rafael F. M. & Reinaldo

Índice

1. Inicio

1.1 Início

O EducaUTF OLC é um microsserviço dedicado à classificação de dados e ciência de dados dentro do EducaUTF.

Escrito em Python, este serviço aceita solicitações de acesso ao seu conteúdo por meio de uma REST API. Em sua fase inicial, o OLC concentra-se na avaliação de **artigos** e **capítulos** em ascensão.

1.1.1 O que é o EducaUTF?

O EducaUTF tem como missão simplificar a criação e acessibilidade de materiais pedagógicos.

Os usuários podem desenvolver materiais no formato de um **Blog interativo** utilizando um superconjunto da linguagem de marcação Markdown. Esse superconjunto permite que os editores incorporem componentes pré-fabricados em suas páginas, proporcionando uma experiência mais interativa para os usuários.

1.1.2 Objetivos Futuros do EducaUTF OLC

O EducaUTF OLC continuará a evoluir, com planos futuros incluindo a execução de diversos algoritmos. Isso abrangerá desde **algoritmos de recomendação**, que ajudarão os usuários a descobrir conteúdos mais relevantes, até outras funcionalidades inovadoras. Estamos comprometidos em aprimorar constantemente a experiência de aprendizado no EducaUTF.

1.1.3 Sobre essa documentação

A documentação deste microsserviço está organizada em diferentes tópicos, sendo acessível de diversas maneiras:

- **Web**: Explore online aqui.
- **PDF**: Faça o download aqui.

Se você está interessado em contribuir, um excelente ponto de partida é o tópico Desenvolvimento

1.1.4 Deploy

A implantação desta aplicação é realizada por meio de Docker Containers. Todo o repositório de imagens está disponível em zrafaf/educa_utf_olc.

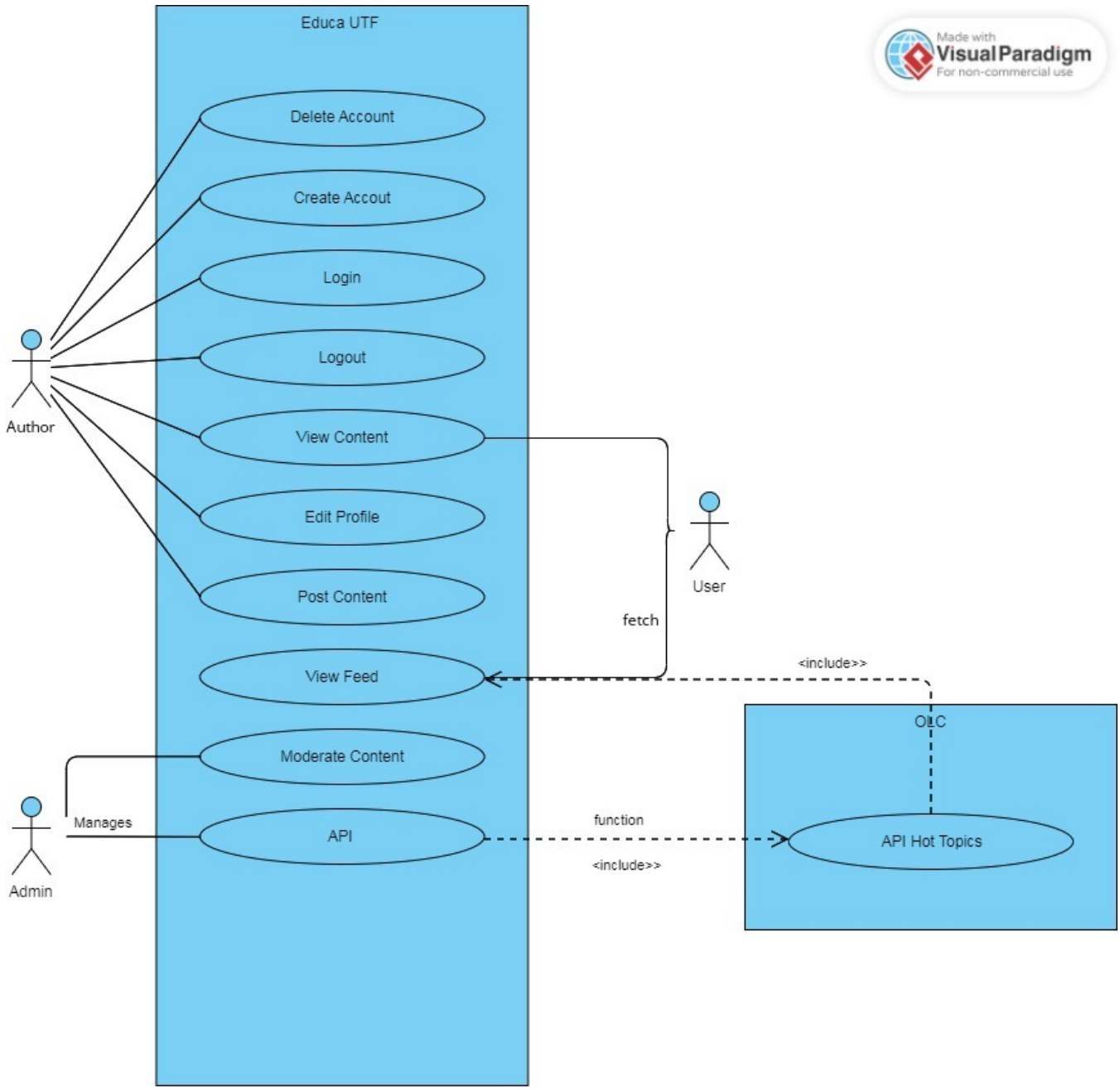
O processo de *deploy* é executado através de um ***pull request*** na *branch* `release`. Isso inicia uma automática build e release usando o GitHub Actions. Estamos sempre buscando aprimorar e facilitar o processo de implantação.

1.1.5 Executando / contribuindo

Para informações sobre como executar ou contribuir para a aplicação visite https://zrafaf.github.io/educa-utf-olc/desenvolvimento/.

1.2 Casos de uso

1.2.1 Diagrama de casos de uso



2. API

2.1 API

Bem-vindo à API do EducaUTF OLC! Aqui, você encontrará informações cruciais sobre como a aplicação utiliza tecnologias específicas para oferecer uma experiência eficiente e dinâmica.

2.1.1 Tecnologias Utilizadas

FastAPI

O EducaUTF OLC utiliza o FastAPI como framework web principal. FastAPI é conhecido por sua rapidez, facilidade de uso e suporte nativo para a especificação OpenAPI, o que facilita a geração automática de documentação.

Uvicorn

O servidor web Uvicorn é a escolha para executar a aplicação FastAPI de forma eficiente e rápida. Ele suporta o protocolo ASGI (Asynchronous Server Gateway Interface), essencial para a execução assíncrona das solicitações.

2.1.2 Explorando a API

Visite os sub-tópicos da API para descobrir mais sobre o seu funcionamento.

2.2 pocketbase_api

2.2.1 Classes

`PocketBase_API`

Classe de abstração do Backend utilizando PocketBase.

Oferece uma implementação assíncrona do SDK fornecido (não oficial) em https://github.com/vaphes/pocketbase

A lista de tipos e relações pode ser encontrada em https://github.com/zRafaF/educa-utf/blob/main/app/types/pocketbase-types.ts

FUNCTIONS

`__init__(pb_url='http://127.0.0.1:8090', timeout=5.0)`

Construtor da classe PocketBase_API.

PARAMETER	DESCRIPTION
`pb_url`	URL base da API do PocketBase. Defaults to "http://127.0.0.1:8090". **TYPE:** `str` **DEFAULT:** `'http://127.0.0.1:8090'`
`timeout`	Tempo máximo de espera para uma resposta da API do PocketBase. Defaults to 5.0. **TYPE:** `float` **DEFAULT:** `5.0`

`build_url(path)`

Função auxiliar para construir a URL completa para a API do PocketBase.

Concatena o `path` ao url base da API do PocketBase.

PARAMETER	DESCRIPTION
`path`	Caminho novo. **TYPE:** `str`

RETURNS	DESCRIPTION
`srt`	url construída. **TYPE:** `str`

`get_list_of_articles_stats_records(page=1, num_of_records=10, filter='', sort='')` `async`

Retorna uma lista de artigos do PocketBase

PARAMETER	DESCRIPTION
`page`	Página a ser retornada. Defaults to 1. **TYPE:** `int` **DEFAULT:** `1`
`num_of_records`	Número de registros a serem retornados. Defaults to 10. **TYPE:** `int` **DEFAULT:** `10`
`filter`	Filtro a ser aplicado. Defaults to "". **TYPE:** `str` **DEFAULT:** `''`
`sort`	Ordenação a ser aplicada. Defaults to "". **TYPE:** `str` **DEFAULT:** `''`

RETURNS	DESCRIPTION
`Response | HTTPError`	httpx.Response: Resposta da requisição HTTP

Exemplo

Python

```python
import asyncio
from pocketbase_api.core import pb_api
from pocketbase_api import helpers as pb_helpers

PB_URL = "https://educautf.td.utfpr.edu.br/db/api"

async def main():
    # Inicializando a comunicação com o banco
    pb_api.set_base_url(PB_URL)
    h = await pb_api.get_list_of_articles_records(5)
    article_list = pb_helpers.get_record_list_from_response(h)
    for i in article_list:
        print(i.get("title"))

if __name__ == "__main__":
    asyncio.run(main())
```

`get_list_of_articles_stats_records_by_age(num_of_records=10, age_in_days=7)` `async`

Retorna uma lista de artigos do PocketBase com idade menor ou igual a `age_in_days` dias. Ordenado por `created` em ordem decrescente.

PARAMETER	DESCRIPTION
`num_of_records`	Número de registros a serem retornados. **TYPE:** `int` **DEFAULT:** `10`
`age_in_days`	Idade máxima dos registros em dias. **TYPE:** `int` **DEFAULT:** `7`

Returns: httpx.Response: Resposta da requisição HTTP

`health()` `async`

Retorna o status de saúde da API do PocketBase

RETURNS	DESCRIPTION
`Response | HTTPError`	httpx.Response: Resposta da requisição HTTP

`set_base_url(url)`

Altera a URL base da API do PocketBase

PARAMETER	DESCRIPTION
`url`	Nova URL base para a API do PocketBase. **TYPE:** `str`

2.2.2 Functions

`calculate_date_since_today(start_date=datetime.today(), number_of_days=30)`

Calcula uma data no formato YYYY-MM-DD de acordo com o número de dias passados.

PARAMETER	DESCRIPTION
`start_date`	Data inicial. Defaults to datetime.today(). **TYPE:** `datetime` **DEFAULT:** `today()`
`number_of_days`	Número de dias a serem subtraídos. Defaults to 30. **TYPE:** `int` **DEFAULT:** `30`

RETURNS	DESCRIPTION
`str`	Data no formato YYYY-MM-DD **TYPE:** `str`

`get_record_list_from_response(response)`

Retorna a lista de Artigos da resposta como uma Lista de dicionários.

PARAMETER	DESCRIPTION
`response`	Resposta da requisição HTTP. **TYPE:** `Response`

RETURNS	DESCRIPTION
`List[Dict]`	List[Dict]: Lista de dicionários representando os Artigos.

2.3 olc_server

2.3.1 Classes

`OLCServer`

Bases: `Server`

Servidor uvicorn customizado

O servidor Uvicorn sobrescreve a função de desligamento para incluir o Rocketry.

2.3.2 Classes

`AlgorithmsOLC`

Classe que contém os algoritmos do OLC

FUNCTIONS

`calculate_score(articles_stats_list, date_for_calculation=datetime.today())`

Calcula o score de cada artigo e atualiza a lista de artigos em alta, ordenando-os pelo score

PARAMETER	DESCRIPTION
`articles_stats_list`	Lista de artigos **TYPE:** `Response | HTTPError`
`date_for_calculation`	Data para calcular o score. Defaults to datetime.today(). **TYPE:** `datetime` **DEFAULT:** `today()`

`fetch_article_list(min_articles=10, age_in_days=7)` async

Fetches a list of articles from the database to calculate the score, if the number of articles is less than `min_articles` then it returns a fallback list of articles.

PARAMETER	DESCRIPTION
`min_articles`	Minimum number of articles in the response. Defaults to 10. **TYPE:** `int` **DEFAULT:** `10`
`age_in_days`	Maximum age of the articles in the response. Defaults to 7. **TYPE:** `int` **DEFAULT:** `7`

RETURNS	DESCRIPTION
`Response | HTTPError`	httpx.Response | httpx.HTTPError: Response from the API

`update_trending_articles()` async

Atualiza a lista de artigos em alta

2.3.3 Functions

`get_olc()` async

Easter egg do OLC

RETURNS	DESCRIPTION
	responses.HTMLResponse: HTML

`get_tasks()` async

Endpoint para obter as tarefas em execução do Rocketry

`get_trending_articles()`

Endpoint para obter os artigos em alta

RETURNS	DESCRIPTION
	``json
	{ "page": 1, "perPage": 50, "totalPages": 1, "totalItems": 50, "items": [{ "id": 1, "title": "Aprendendo a usar o PocketBase", "description": "Aprenda a usar o PocketBase", "content": "Aprenda a usar o PocketBase", "created": "2023-09-15 16:50:07.282000", "updated": "2023-09-15 16:50:07.282000", "author": 1, "likes": 0, "latest_views": 0, "tags": [1, 2] }, ...]
	}
	``

`read_root(request)` async

Endpoint de saúde da API

2.3.4 Functions

`check_schedule_health()`

Tarefa para verificar a saúde do scheduler

`update_trending_articles()` async

Tarefa para atualizar os artigos em ascensão

3. Desenvolvimento

3.1 Desenvolvimento

Bem-vindo à página de desenvolvimento do EducaUTF OLC! Se você está interessado em contribuir para a aplicação, este é o ponto de partida perfeito. Aqui estão alguns recursos importantes para começar:

3.1.1 Formatação

Para padronização da formatação foi utilizado o black.

3.1.2 Executando

Antes de executar a aplicação garanta que o **ambiente de desenvolvimento** foi configurado corretamente, visite desenvolvimento-setup para saber mais.

Para iniciar a aplicação, execute o script `main.py` localizado no diretório `src` usando o seguinte comando:

Bash

```
python src/python.py
```

Este comando iniciará a conexão com o backend do **EducaUTF** e criará um endpoint no endereço `127.0.0.1:8000`.

Além disso, você pode passar alguns **argumentos** opcionais para personalizar o comportamento da aplicação:

- `--pb_url`: Define a URL do backend da aplicação.
- `--host`: Define o endereço no qual a aplicação será servida.
- `--port`: Define a porta na qual a aplicação será servida.

Bash

```
python src/main.py --host=0.0.0.0 --port=3000 --pb_url=https://my_pb_backend/api
```

Certifique-se de ajustar os valores conforme necessário para atender às suas configurações específicas. Agora você está pronto para explorar a aplicação EducaUTF OLC localmente!

3.2 Setup

Esta página fornece um guia detalhado para configurar este projeto. Certifique-se de seguir cada passo cuidadosamente.

Para executar este projeto, você precisará de Python, **PIP** e **GIT**.

Este projeto foi desenvolvido utilizando a versão `Python 3.11`.

3.2.1 Clonar o repositório

Abra o terminal e execute os seguintes comandos:

Bash

```
git clone https://github.com/ZRafaF/educa-utf-olc

cd educa-utf-olc
```

3.2.2 Criando um ambiente virtual

A criação de um ambiente virtual não é obrigatória, mas é altamente **recomendada**. Para saber mais sobre ambientes virtuais em Python, visite https://docs.python.org/3/library/venv.html.

Execute os seguintes comandos para criar e ativar um ambiente virtual:

Bash

```
python -m pip install --user virtualenv

python -m venv venv
```

Um diretório `venv` será criado na pasta raiz do projeto.

Ativação do Ambiente Virtual

Ativação no Windows

```
venv/Scripts/activate
```

ou

Ativação no Linux

```
source venv/bin/activate
```

Com o ambiente virtual ativado, qualquer biblioteca instalada será aplicada apenas a este ambiente.

3.2.3 Instalando dependências

Execute o seguinte comando para instalar todas as dependências listadas no arquivo `requirements.txt`.

Bash

```
pip install -r requirements.txt
```

Agora seu ambiente está pronto para executar a aplicação.

3.2.4 Testes

Para obter informações sobre os testes, consulte Desenvolvimento-Testes

3.2.5 Documentação

Para executar ou contribuir para a documentação, consulte Desenvolvimento-Documentação

3.3 Documentação

A documentação desse projeto foi criada utilizando MkDocs juntamente com Material for MkDocs e alguns outros plugins.

3.3.1 Auto documentação

Para gerar documentação automática do código está sendo usado o mkdocstring, com o handler mkdocstrings-python

O padrão utilizado nesse projeto é o Google-style, também são suportados alguns outros veja griffe para saber mais.

3.3.2 Setup

Para poder editar a documentação será preciso instalar suas **dependências** com:

Bash

```
pip install -r docs/requirements.txt
```

Após isso utilize o comando a seguir para servir a documentação no `localhost`:

Bash

```
mkdocs serve
```

Ou o comando a seguir para criar uma build completa da documentação

Bash

```
mkdocs build
```

Info

A documentação desse projeto também pode ser acessada em formato de `.pdf`, entretanto esse so será criado caso seja a variável de ambiente `ENABLE_PDF_EXPORT` seja igual a `1`.

Bash

```
ENABLE_PDF_EXPORT: 1
```

3.4 Testes

Os testes da aplicação foram realizados através do pytest e os seguintes módulos auxiliares:

- pytest-httpserver: Um módulo para simulação de um *HTTP endpoint*. Consulte a documentação para mais detalhes.
- pytest-asyncio: Um módulo que oferece suporte a funções `assíncronas`.

3.4.1 Instalando dependências

Para instalar as dependências necessárias, execute o seguinte comando no terminal:

Bash

```
pip install -r tests/requirements.txt
```

3.4.2 Executando os testes

Siga os passos abaixo para executar a bateria de testes:

1. Certifique-se de que todas as requisitos foram instalados.
2. Execute o seguinte comando:

Bash

```
pytest .\tests\
```

Isso garantirá que todos os aspectos críticos da aplicação foram testados de maneira abrangente.