Educa UTF

Documentação completa

Rafael F. Meneses

Índice

1. Inicio

1.1 Início

1.1.1 A filosofia

O Educa UTF tem como sua filosofia facilitar a criação e acessibilidade de materiais pedagógicos.

Os usuários podem criar materiais em formato de **Blog interativo** utilizando um superconjunto da linguagem de marcação Markdown. Esse superconjunto permite que o editor adicione componentes pré-fabricados à suas páginas, o que permite o aumento da interatividade com o usuário.

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

2. Criar Conteúdo

2.1 Criar conteúdo

Essa seção é destinada a usuários com a intenção de **criar novas páginas**

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3. Desenvolvimento

3.1 Desenvolvimento

Esta coleção é focada principalmente para os interessados em modificar a aplicação.

Serão apresentandas

Para informações sobre o setup do ambiente de desenvolvimento visite setup.

3.1.1 Arquitetura

A aplicação principal foi escrita majoritariamente na linguagem de programação `.tsx` um superconjunto de TypeScript.

Os conteúdos são escritos em uma especie de Markdown customizado, utilizando aspectos da linguagem `.mdx`.

O *front-end* do Educa UTF foi desenvolvido a partir da biblioteca ReactJS. Para mais informações acesse a página front-end

3.1.2 Github

O repositório educa-utf no Github é o responsável em criar novas **Releases** automaticamente da imagem docker.

Continuous Integration Docker-hub

A integração continua com o Docker-hub é realizada no momento que ocorre uma ação de `push` na *branch* `release`.

> **Attention**
>
> A `tag` da release é o **titulo do *Pull Request***. Por isso preste atenção no momento que fizer um PR de outra *branch* para a *branch* `release`. Por convenção o titulo deve obedecer "vx.x.x". Por exemplo "v0.1.2"

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

3.2 Configuração

Esta página definirá o passo a passo para compilar este projeto.

3.2.1 Clonar o repositório

Bash

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

cd educa-utf
```

3.2.2 Instalar dependências

Bash

```
npm run install
```

3.2.3 Criando variáveis de ambiente

Os seguintes arquivos são necessários para o desenvolvimento da aplicação:

Variáveis do NEXTJS

Crie um arquivo de variável de ambiente recomendo `.env.local` e adicione as seguintes variáveis:

Bash

```
# URL of your PocketBase db
PB_URL=https://myproject.pockethost.io

# Should Next analyze your packages during build
ANALYZE=false
```

> **Atenção**
>
> Esses são **SEGREDOS** e não deverão ser enviados a repositórios públicos. Seja GitHub, DockerHub e outros.

Variáveis do TypeGen

Em um arquivo de variável de ambiente chamado `.env` adicione as seguintes variáveis:

Bash

```
# URL of your PocketBase db
PB_TYPEGEN_URL=https://myproject.pockethost.io
```

```
# Email of an admin on PocketBase dashboard
PB_TYPEGEN_EMAIL=admin@myproject.com

# Password of an admin on PocketBase dashboard
PB_TYPEGEN_PASSWORD=secr3tp@ssword!
```

Atenção

Esses são **SEGREDOS** e não deverão ser enviados a repositórios públicos. Seja GitHub, DockerHub e outros.

3.2.4 Documentação

Este projeto foi desenvolvido usando a versão Python `Python 3.10.x`. A etapa a seguir é necessária apenas para aqueles que desejam **editar a documentação**.

Para compilar a documentação você precisará de Python, **PIP** e **GIT**.

Criando um ambiente virtual

A etapa a seguir não é obrigatória, mas é **recomendada**. Se você quiser saber um pouco mais sobre ambientes virtuais de python visite https://docs.python.org/3/library/venv.html

Bash

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

python -m venv venv
```

Um diretório `venv` deve ser criado na pasta raiz do projeto.

Como ativar:

Ativação no Windows

```
venv/Scripts/activate
```

or

Ativação no Linux

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

Com isso o ambiente virtual estará ativado e qualquer biblioteca instalada será aplicada apenas à esse ambiente.

Instalando dependências

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

Esse comando irá instalar todas as dependências contidas no arquivo `requirements.txt`

Build

Para compilar a documentação temos duas opções:

- **Serve**:

  Esta opção é usada para depuração, ela abrirá a página estática em uma das portas localhost.

  ```Bash
  mkdocs serve
  ```

- **Build**: Esta opção cria uma compilação da documentação e a salva no diretório `/site/`.

  ```Bash
  mkdocs build
  ```

> **Note**
>
> Esteja ciente da **Environment Variable** `ENABLE_PDF_EXPORT`, ela só irá gerar o PDF se essa variável estiver definida como `1`.
>
> Você pode alterar o arquivo `mkdocs.yml` e remover esta linha se preferir.

3.2.5 Integração Pocketbase

Para gerar os Bindings do pocket base está sendo usado a biblioteca pocketbase-typegen. Para gerar os tipos execute:

`yarn typegen`

O *output* está localizado em `/src/types/pocketbase-types.ts`.

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

3.3 Posts

Aqui comentarei sobre o sistema de posts que desenvolvi.

3.3.1 Renderizando conteúdo

Para a renderização de arquivos Markdown `.md` estou utilizando o pacote markdown-to-jsx.

3.3.2 Estilizando

Para a estilização modifiquei o pacote github-markdown-css e incorporei os estilos em um arquivo `github-markdown.css`.

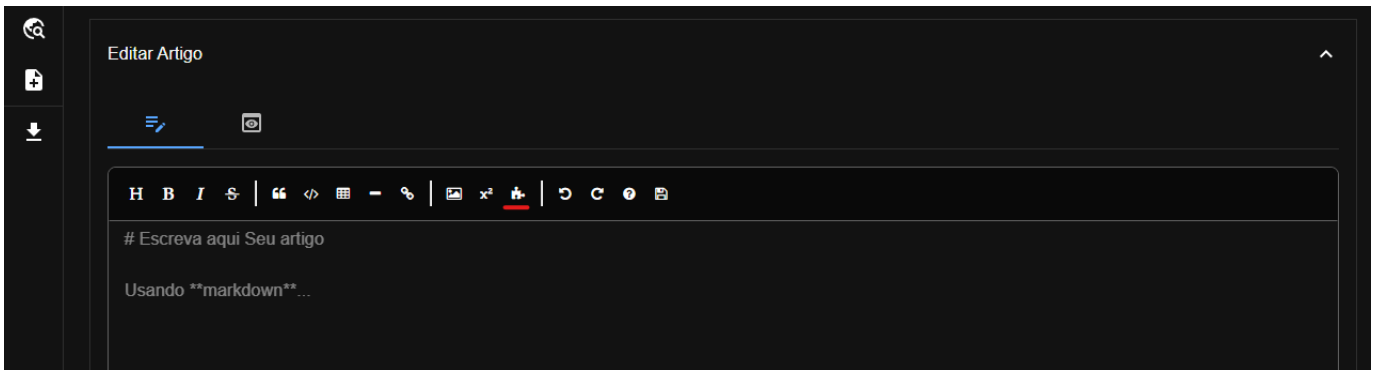
Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3.4 Plugins

Plugins são uma maneira de aprimorar e adicionar interatividade a conteúdos publicados.

Os Plugins podem ser acessados na Barra de ferramentas do editor.



3.4.1 Criando um novo plugin

Você pode contribuir com o EducaUTF criando novos plugins!

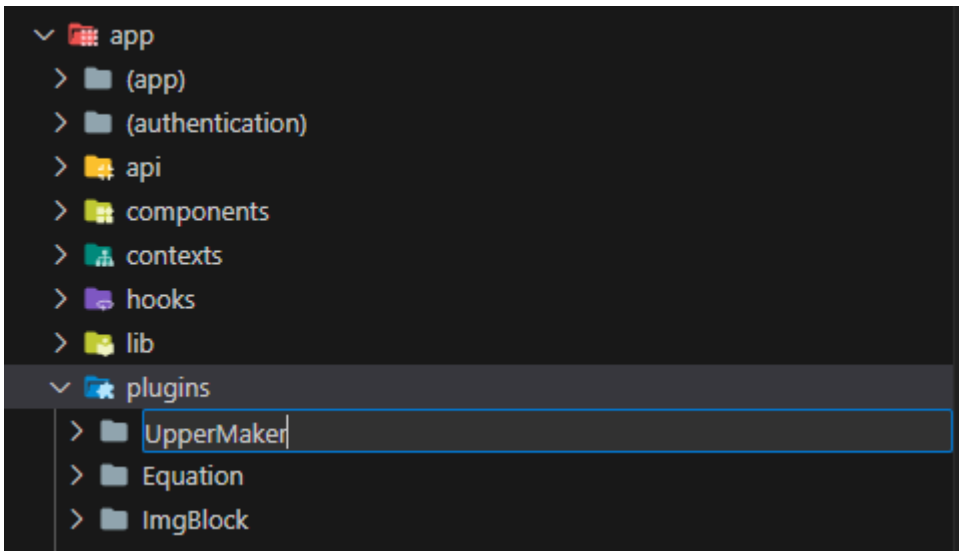
Criar um plugin é fácil, vou te mostrar como fazer.

Vamos criar um plugin que converte o input do usuário para letras maiúsculas.

Preparação

O primeiro passo é criar uma pasta com o nome de seu plugin, todos os plugins devem **Ter sua própria pasta**.

Vá até a pasta `/app/plugins/` e crie a sua pasta, chamarei de `UpperMaker`.

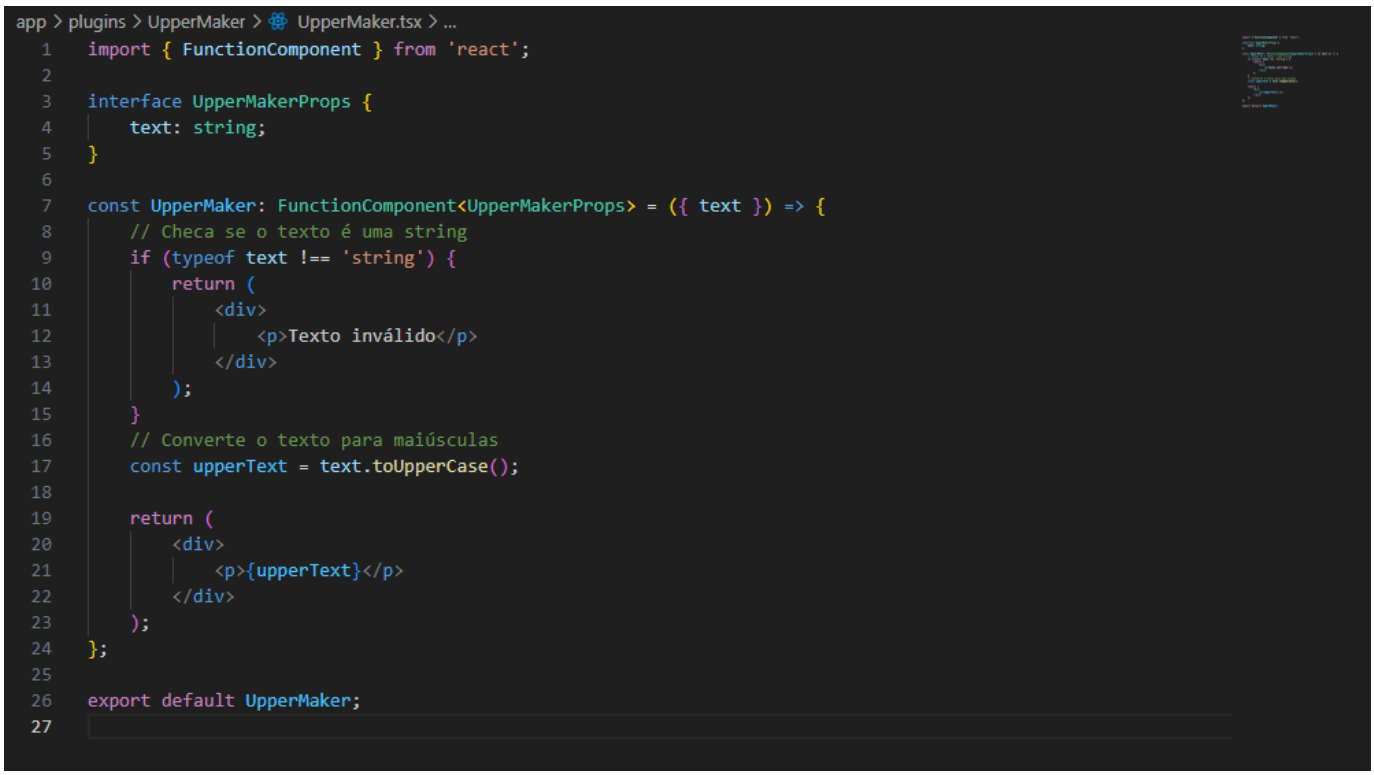


Agora vou criar 2 arquivos:

- `UpperMaker.tsx`: Esse é o componente que será renderizado para os leitores.
- `UpperMakerEditor.tsx`: Esse é o componente que o autor usará para editar o conteúdo.

UpperMaker.tsx

Vamos começar criando o componente que será renderizado.



Esse componente recebe uma string chamada `text` como `prop`, checa se ela é uma string válida e logo em seguida renderiza o texto em letras maiúsculas.

> **dica**
>
> Você pode utilizar outras bibliotecas, css e o que mais desejar para criar esse componente. Se divirta!

UpperMakerEditor.tsx

Agora vamos para o editor, todos os editores serão renderizados dentro de um modal. Você não precisa se preocupar com isso.

Text Only

```
'use client';

import { FunctionComponent, useState } from 'react';
import { PluginEditorProps } from '../PluginsTypes';
```

```
import UpperMaker from './UpperMaker';

const UpperMakerEditor: FunctionComponent<PluginEditorProps> = ({
    returnFunction,
}) => {
    const [userInput, setUserInput] = useState<string>('');

    return (
        <>
            <input
                type="text"
                placeholder="Digite o texto aqui"
                onChange={(e) => {
                    setUserInput(e.target.value);
                }}
            />
            <button
                onClick={() => {
                    returnFunction(<UpperMaker text={userInput} />);
                }}
            >
                Enviar
            </button>
        </>
    );
};

export default UpperMakerEditor;
```

Os editores recebem algumas `props` por padrão, por tanto vamos começar utilizando o tipo `PluginEditorProps` para nos fornecer intellisense.

Todos os plugins devem utilizar a função `returnFunction` essa função recebe uma o componente que deve ser renderizado.

Adicionando à barra de ferramentas

Para adicionar seu plugin à barra de ferramentas você deve atualizar o arquivo `/app/plugins/PluginsArray.tsx`.

Vamos adicionar o novo plugin

```tsx
'use client';

import EquationEditor from './Equation/EquationEditor';
import { PluginType } from './PluginsTypes';
import Equation from './Equation/Equation';
import RadialSelectorEditor from './RadialSelector/RadialSelectorEditor';
import RadialSelector from './RadialSelector/RadialSelector';
import UpperMakerEditor from './UpperMaker/UpperMakerEditor';
import UpperMaker from './UpperMaker/UpperMaker';

const PluginsArray: PluginType[] = [
    {
        key: 'equationPlugin',
        title: 'Criador de equações',
        tooltip: 'Equação matemática',
        editor: EquationEditor,
        render: Equation,
        hidden: true,
        category: '',
    },
    {
        key: 'radialSelectorPlugin',
        title: 'Exercício de seleção radial',
        tooltip: 'Novo exercício de Seleção radial',
        editor: RadialSelectorEditor,
        render: RadialSelector,
        hidden: false,
        category: 'Exercícios',
    },
    {
        key: 'upperMakerPlugin',
        title: 'Utilizar maiúsculas',
        tooltip: 'Converte o texto para maiúsculas',
        editor: UpperMakerEditor,
        render: UpperMaker,
        hidden: false,
        category: 'Texto',
    },
];

export default PluginsArray;

```

Preencha as informações como achar melhor, utilize o intellisense para mais informações sobre os parâmetros.

Adicionando o plugin ao renderizador

Por fim vá ao arquivo `/app/plugins/useOverridePlugins.tsx` para permitir que o renderizador utilize seu plugin.

O primeiro passo é importá-lo dinamicamente:

```
23
24  const UpperMaker = dynamic(() => import('@/plugins/UpperMaker/UpperMaker'), {
25      ssr: true,
26  });
```

E por fim adicioná-lo aos *overrides*

Text Only

```
'use-client';

import ImgBlock from '@/plugins/ImgBlock/ImgBlock';
import PreBlock from '@/plugins/PreBlock/PreBlock';
import { MarkdownToJSX } from 'markdown-to-jsx';
import dynamic from 'next/dynamic';
import { useMemo } from 'react';

const Equation = dynamic(() => import('@/plugins/Equation/Equation'), {
    ssr: true,
});
const RadialSelector = dynamic(
    () => import('@/plugins/RadialSelector/RadialSelector'),
    {
        ssr: true,
    }
);

const UpperMaker = dynamic(() => import('@/plugins/UpperMaker/UpperMaker'), {
    ssr: true,
});

const useOverridePlugins = () => {
    const overrides = useMemo<MarkdownToJSX.Overrides>(() => {
        return {
            pre: PreBlock,
            Equation: Equation,
            img: ImgBlock,
            RadialSelector: RadialSelector,
            UpperMaker: UpperMaker,
        };
    }, []);

    return [overrides] as const;
};

export default useOverridePlugins;
```

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3.5 Docker

3.5.1 Criando um container da aplicação NEXT.js

Dockerfile

Cheque a configuração do arquivo `Dockerfile`, a versão contida no repositório foi gerada apartir do projeto nextjs-with-docker

ATENÇÃO As variáveis de ambiente `ENV` definidas na `Dockerfile`, principalmente `PB_URL` essa variável define a URL do backend

Criando a imagem

A imagem oficial desse projeto pode ser encontrada em https://hub.docker.com/r/zrafaf/educa_utf_nextjs

Rode `docker build -t educa_utf_nextjs .` na root do projeto.

Dando *push* em uma imagem para o docker hub

1. Faça login no docker usando `docker login`
2. Crie uma nova tag para sua imagem com `docker tag educa_utf_nextjs [username]/educa_utf_nextjs:[version]`
3. Faça o push com `docker push [username]/educa_utf_nextjs:[version]`

> **Fazendo `push` de uma imagem como `latest`**
>
> 1. `docker tag educa_utf_nextjs zrafaf/educa_utf_nextjs`
> 2. `docker push zrafaf/educa_utf_nextjs`

A qualquer momento você pode encontrar suas imagens através de `docker images`

3.5.2 Iniciando os Containers

Variáveis de ambiente

O docker compose faz uso das seguintes variáveis de ambiente:

- `UMAMI_APP_SECRET`
- `UTF_AUTH_TOKEN`

Esses **segredos** devem ser setados dentro de um arquivo `.env` no mesmo diretório do `docker-compose.yaml`.

Exemplo de `.env`

Bash

```
UMAMI_APP_SECRET=meusegredoumami
UTF_AUTH_TOKEN=meutokendeautenticacao
```

Docker compose

Para iniciar os containers você pode utilizar o seguinte docker-compose

YAML

```yaml
version: '3.8'

services:
    next:
        depends_on:
            - pocketbase
        image: zrafaf/educa_utf_nextjs:latest
        restart: unless-stopped
        ports:
            - 3000:3000

    pocketbase:
        image: zrafaf/educa_utf_pocketbase:latest
        restart: unless-stopped
        ports:
            - 8090:8090
        volumes:
            - ./pocketbase_data:/pb/pb_data
        environment:
            UTF_AUTH_TOKEN: '${UTF_AUTH_TOKEN}'

    watchtower:
        image: containrrr/watchtower
        restart: unless-stopped
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
        command: --interval 30

    nginx_proxy:
        image: jc21/nginx-proxy-manager:latest
        restart: unless-stopped
        ports:
            - '80:80'
            - '81:81'
            - '443:443'
        volumes:
            - ./nginx_proxy/data:/data
            - ./nginx_proxy/letsencrypt:/etc/letsencrypt

    dozzle:
        container_name: dozzle
        image: amir20/dozzle:latest
        labels:
            - 'com.centurylinklabs.watchtower.enable=false' # Disables watchtower auto update
```

```
        restart: unless-stopped
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
        ports:
            - 9999:8080

    # Runs on port 61208
    glances:
        image: nicolargo/glances:latest-full
        pid: host
        network_mode: host
        restart: unless-stopped
        volumes:
            - /var/run/docker.sock:/var/run/docker.sock
            - /run/user/1000/podman/podman.sock:/run/user/1000/podman/podman.sock
        environment:
            - 'GLANCES_OPT=-w'

    umami:
        image: zrafaf/educa_utf_umami:latest
        ports:
            - '3100:3000'
        environment:
            DATABASE_URL: postgresql://umami:umami@umami_db:5432/umami
            DATABASE_TYPE: postgresql
            APP_SECRET: '${UMAMI_APP_SECRET}'
        depends_on:
            umami_db:
                condition: service_healthy
        restart: unless-stopped

    umami_db:
        image: postgres:15-alpine
        environment:
            POSTGRES_DB: umami
            POSTGRES_USER: umami
            POSTGRES_PASSWORD: umami
        volumes:
            - ./umami-db-data:/var/lib/postgresql/data
        restart: unless-stopped
        healthcheck:
            test:
                [
                    'CMD-SHELL',
                    'pg_isready -U $${POSTGRES_USER} -d $${POSTGRES_DB}',
                ]
            interval: 5s
            timeout: 5s
            retries: 5
```

Você pode iniciar os containers com `docker compose up`.

Info

Se algo der errado você pode tentar criar o volume manualmente com:

Bash

```
mkdir -p ~/pocketbase-data
chown -R $USER:$USER ~/pocketbase-data
```

Você também pode checar se o usuário tem permissão de leitura e escrita com

Bash

```
chmod -R 755 ~/pocketbase-data
```

Atualizando a imagem `educa_utf_nextjs`

Para atualizar para a ultima versão disponivel será necessário **apagar** a imagem defasada.

Para ver as imagens atuais execute `docker images`. Você deve ver algo parecido com:

Bash

```
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
zrafaf/educa_utf_nextjs   latest    ffb4489371c3   15 minutes ago   818MB
```

Para deletar a imagem defasada **FORÇADO** você pode executar: `docker rmi [id] -f`.

No caso da imagem a cima seria necessário executar `docker rmi ffb4489371c3 -f`

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3.6 Front-end

Esta página apresentará todas as informações relacionadas ao front-end da aplicação.

3.6.1 NEXTJS

Referências

- Estrutura do projeto
- Essenciais do React
- Nextjs com docker oficial

3.6.2 Editor de artigos

Para o editor de artigos está sendo utilizado o react-simplemde-editor que se trata de um *wrap* do easy-markdown-editor.

O EasyMDE faz uso do Fonts Awesome V5 para seus diferentes ícones.

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

3.7 Backend

Foi utilizado o Banco de dados / backend de código aberto PocketBase.

Um dos grandes problemas enfrentado para a criação de conteúdo por parte dos usuários é a possibilidade de injetar código malicioso nos arquivos de Markdown, por tanto algumas medidas tiveram de ser tomadas que acabaram tornando a aplicação menos customizável.

3.7.1 Setup

Para desenvolvimento recomendo realizar em uma **máquina LINUX** como seu sistema operacional. Algumas funções do backend assumem que estão rodando em um ambiente UNIX e pode apresentar funcionamento incorreto se executado em outro sistema operacional.

> **Tip**
>
> Caso seu ambiente de desenvolvimento seja o windows dê uma olhada em Windows Subsystem for Linux (WSL).

Variáveis de ambiente

Para iniciar o backend é necessário criar as seguintes **Variáveis de Ambiente**:

Bash

```
UTF_AUTH_TOKEN=meutokendeautenticacao
```

DOCKER COMPOSE

O docker-compose da aplicação carrega os segredos a partir de um arquivo `.env`. Portanto a variável `UTF_AUTH_TOKEN` deve estar lá.

Exemplo de `.env`

Bash

```
UTF_AUTH_TOKEN=meutokendeautenticacao
```

Executando o aplicativo

Você pode executar a seguinte linha na *root* do projeto:

Bash

```
UTF_AUTH_TOKEN=meutokendeautenticacao ./pocketbase serve
```

Ou carregar a chave de autenticação apartir de um arquivo `secrets.txt` com

Bash

```
source secrets.txt && ./pocketbase serve
```

3.7.2 Como a autenticação é feita?

A autenticação de usuários da UTFPR é feito através de um HTTP POST request para o *endpoint* `api/educautf/utfpr-auth`.

O corpo do request deve seguir o formato:

JSON

```
{
    "username": "my-username",
    "password": "my-password",
}
```

Riscos

A autenticação no sistema da UTFPR é feito diretamente no Backend sem acesso ao usuário através de um *hook*, para saber mais visite https://github.com/zRafaF/educa-utf-db/tree/main/pb_hooks. A única vulnerabilidade para ataque direto seria através do spoofing de senhas.

3.7.3 Como a compressão de imagens é feita?

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

3.8 Documentação

Para desenvolvimento da documentação foi utilizado a biblioteca de Python MkDocs, junto com o tema Material for MkDocs.

Também foi utilizado o plugin MkDocs with PDF para a renderização de toda a documentação como `.pdf` automaticamente, esse arquivo pode ser encontrado aqui.

3.8.1 Configurando o ambiente de desenvolvimento

Para informações sobre como configurar o ambiente de desenvolvimento para a documentação visite a página setup.

3.8.2 Editando a documentação

Para informações sobre como editar documentação do MkDocs visite ZRafaF/ReadTheDocksBase.

Para informações mais a fundo verifique:

- MkDocs
- Material for MkDocs.

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3.9 SEO

SEO ou *Search Engine Optimization* é um aspecto crucial para essa aplicação. Esta página apresentará as ferramentas e técnicas utilizadas para melhorar o SEO do EducaUTF.

3.9.1 Indexadores

Google Search Console

O Google Search Console é uma ferramenta fornecida pela Google para analisar a indexação de seu website na plataforma.

Os dados fornecidos por essa ferramenta são cruciais para a melhoria da aplicação.

VERIFICANDO PROPRIEDADE DO WEBSITE

Para verificar a propriedade da página foi adicionado o arquivo `google9365f4fd3245c688.html` a pasta `public` da aplicação.

Bing Webmaster

O Bing Webmaster é a ferramenta de indexação do Bing, ela permite a importação de páginas direto do Google Search Console, facilitando a integração.

3.9.2 Metadados

Metadados são informações extremamente úteis para os *crawlers*, para os artigos estes foram gerados dinamicamente, o exemplo a seguir mostra a geração de metadados para os artigos:

Text Only

```
export async function generateMetadata({
    params,
}: PageProps): Promise<Metadata> {
    const articleId = params.slug[0];

    try {
        const article = await getArticleById(articleId);
        const articleStats = await getArticleStatsById(articleId);
        const ArticleCoverUrl = getArticleCoverURL(article);
        let tag = article.expand?.tag?.name ?? '';
        return {
            title: article.title,
            description: article.description,
            applicationName: 'EducaUTF',
            authors: [{ name: articleStats.author_name }],
            openGraph: {
                title: article.title,
                description: article.description,
                siteName: 'EducaUTF',
```

```
                images: [{ url: ArticleCoverUrl }],
                locale: 'pt_BR',
                type: 'website',
            },
            keywords: ['EducaUTF', 'artigo', article.title, tag],
        };
    } catch (error) {
        return {
            title: 'Artigo privado',
            description: 'Artigo privado',
        };
    }
}
```

3.9.3 Sitemap

Sitemaps fornecem aos indexadores todo o esquema de seu website, eles são uteis para facilitar que o *crawler* alcance páginas que não possuam muitos links a elas.

O arquivo `sitemap.xml` também é gerado dinamicamente, para incluir todo o conteúdo gerado por usuários, você pode encontrar o script responsável em gera-lo em `app/sitemap.ts`.

3.9.4 robots.txt

Esse arquivo serve como instruções para os *crawlers*, esse também é gerado dinamicamente e pode ser encontrado em `app/robots.ts`.

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024

3.10 Proxy Reverso

Aqui será apresentado a arquitetura, setup e configuração do proxy reverso da aplicação.

3.10.1 Nginx Proxy Manager

Para o proxy reverso foi utilizado o Nginx Proxy Manager, o deploy foi feito através do container `Docker` e `Docker Compose`.

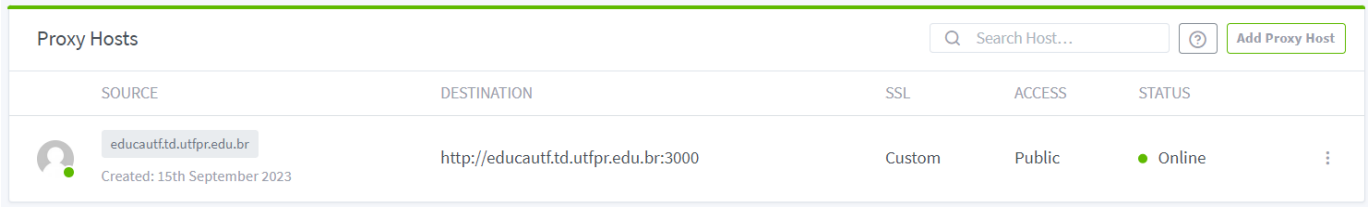
3.10.2 Configurando o proxy

Para acessar o serviço é necessário que seja acessado via a rede da UTFPR-toledo, ou então fazendo uso da VPN.

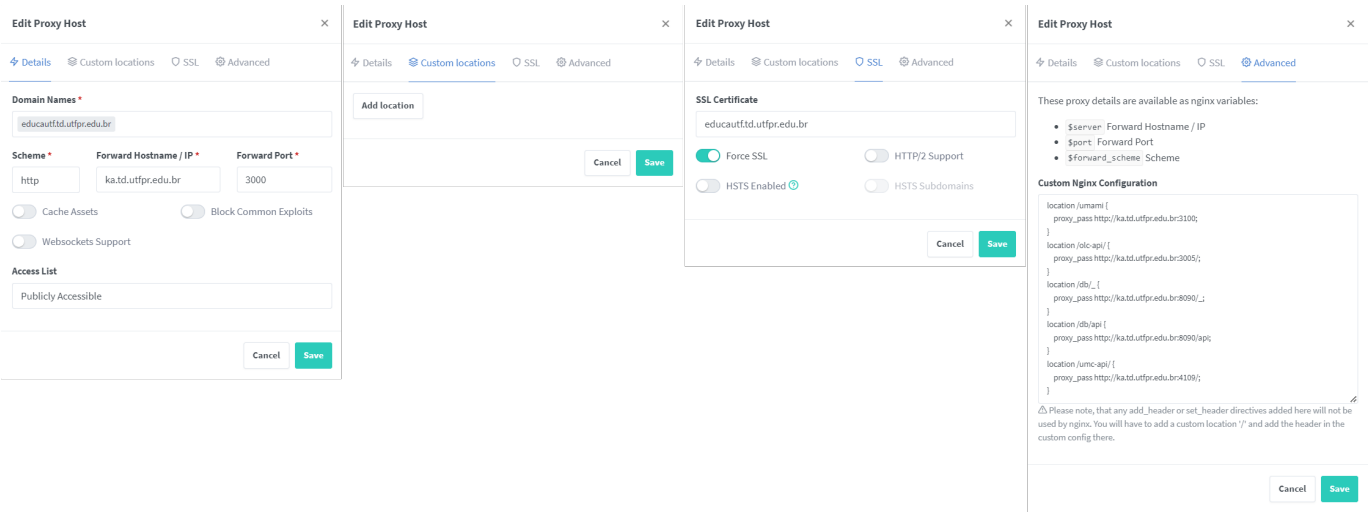
O acesso pode ser feito via `HTTP` no endereço `educautf.td.utfpr.edu.br:81`

Proxy Hosts

O seguinte proxy host foi configurado:



Com as seguintes configurações:



Para as `custom locations` a seguinte configuração está sendo usada:

Text Only

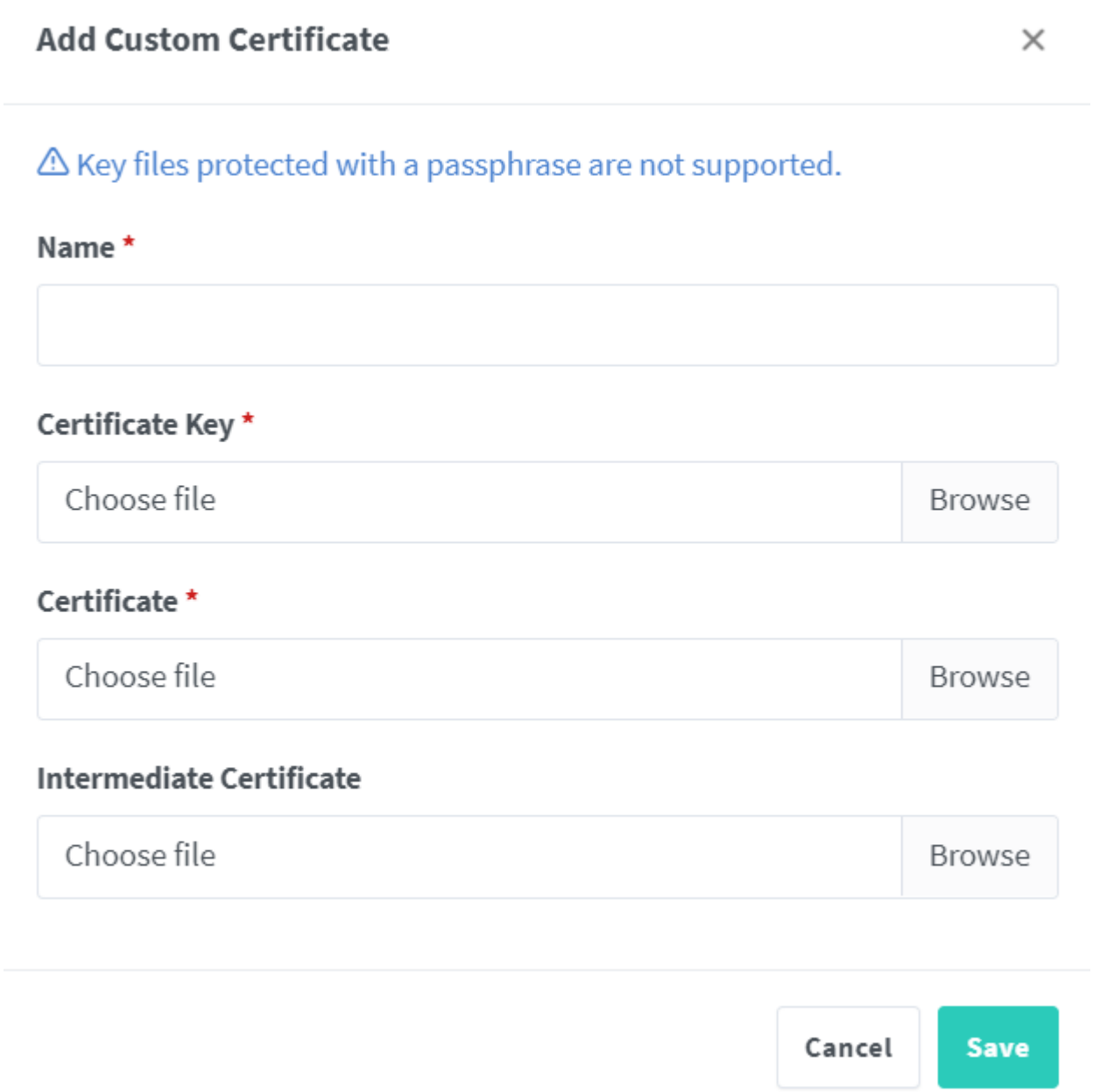
```
location /umami {
    proxy_pass http://ka.td.utfpr.edu.br:3100;
}
location /olc-api/ {
    proxy_pass http://ka.td.utfpr.edu.br:3005/;
}
location /db/_ {
    proxy_pass http://ka.td.utfpr.edu.br:8090/_;
}
location /db/api {
    proxy_pass http://ka.td.utfpr.edu.br:8090/api;
}
location /umc-api/ {
    proxy_pass http://ka.td.utfpr.edu.br:4109/;
}
```

Atenção

As configurações podem estar desatualizadas.

Certificado SSL

Ao selecionar *`Add SSL Certificate`* e *`Custom`* você será apresentado a seguinte tela:



Os arquivos relacionados ao certificado são:

- `*.key` : Deverá ser inserido no campo `Certificate key`, atenção essa é uma chave **PRIVADA**;
- `*.cer` : Deverá ser inserido no campo `Certificate`;
- `*.pem` : Deverá ser inserido no campo `Intermediate Certificate`;

Última atualização: 4 de março de 2024
Criado em: 4 de março de 2024

3.11 Métricas

3.11.1 Umami

Para as métricas do website está sendo utilizado Umami através de uma *build* customizada.

3.11.2 Customização da *build*

Para adequar as variaveis do container foi feito um *fork* do repositório original, esse *fork* pode ser acessado em https://github.com/zRafaF/educa-utf-umami, nele adicionei dois *workflows* um para checar nomenclatura de *releases* e outro para realizar a *build* e publicação automática do *container*, observe a variável `BASE_PATH`.

O repositório no DockerHub pode ser acessado em https://hub.docker.com/r/zrafaf/educa_utf_umami

3.11.3 Dashboard

A aplicação está acessível através do proxy reverso em https://educautf.td.utfpr.edu.br/umami.

Você também pode visualizar os dados públicos em https://educautf.td.utfpr.edu.br/umami/share/NmknLrXmtHfUfTwY/EducaUTF

3.11.4 Segredos

O `docker-compose.yaml` depende da variável de ambiente `UMAMI_APP_SECRET`, essa deve ser setada dentro de um arquivo `.env` no mesmo diretório do `docker-compose.yaml`.

Exemplo de `.env`

Bash

```
UMAMI_APP_SECRET=meusegredoumami
```

Última atualização: 4 de março de 2024

Criado em: 4 de março de 2024