Usage Manual of Book Explorer

Contents

1 Generalities

In this section are found different procedures and basic concepts that are shared throughout the different profiles and stages.

1.1 Initial Installation

To start developing a feature, test a functionality or module, or deploy the service, it is required to clone the repository, for this purpose it is used:

```
git clone https://github.com/alexisuaguilaru/BookExplorer.git
```

Each of the modules that are written in Python, have a `requirements.txt` file that contains the libraries required for the operation of the module or, alternatively, to install to develop or implement a change in a feature of the respective module.

1.2 Docker Compose

The service is divided into three files `docker-compose.yml`, where each one contains the different microservices according to the functionality it performs within the service. The files contain the following microservices:

- **Backend**: MongoDB database [BooksDatabase], Data extraction from the API [DataExtraction] and API of the main microservice [SystemRecommender].
- **Frontend**: CSS styles builder [NodeJS] and web interface [WebInterface].
- **Proxy**: Reverse proxy based on nginx [Proxy] and automation of SSL certificate acquisition [Certbot].

1.3 Configuration File *.env*

The `.evn_example` file serves as an example of the environment variables that must be changed to adjust the behavior of the three `docker-compose.yml` files. The configurable environment variables are:

- *UID*: ID of the non root user that executes the different services in the `docker-compose`
- *GID*: ID of the group the non root user belongs to
- *MONGO_INITDB_ROOT_USERNAME*: Username of the database administrator in MongoDB.

- *MONGO_INITDB_ROOT_PASSWORD*: Password for the MongoDB administrator.
- *DB_NAME*: Name of the database to be created in MongoDB.
- *MONGO_WRITE_USERNAME*: Username of non root user who can only write data to the database.
- *MONGO_WRITE_PASSWORD*: Password for user with write role.
- *MONGO_READ_USERNAME*: Username of non root user who only reads data in the database.
- *MONGO_READ_PASSWORD*: Password for the user with read role.
- *AMOUNT_BOOKS*: Number of books that are extracted from the API. It does not represent the final number found in the associated collection in MongoDB.
- *USER_AGENT*: Identification credentials for the OpenLibrary API (project name, email). It is not a token or key for the API.
- *FLASK_SECRET_KEY*: Secret key used by Flask for encryption and session management.
- *DEBUG_MODE*: Mode in which Flask applications run, whether in debug mode or production mode.
- *DOMAIN_NAME*: Name of the domain where the service will be served.
- *EMAIL_ID*: Email used to generate certificates generated from CertBot.

1.4 Execution File *Makefile*

The `Makefile` file contains commands that execute the different services contained in the `docker-compose.yml` files depending on whether it is the first time it is executed or not. It contains the following variables to configure:

- *DOMAIN_NAME*: Name of the domain where the service will be served and from where the self-signed certificates are generated.
- *ENV*: Environment under which the service is being executed, depending on whether it is for development, testing or deployment.

Since it deals with both passwords and secret keys, the following command is available to generate them using alphanumeric characters:

```
make secret_key LENGTH=integer
```

Where *LENGTH* controls the length of the key, preferably greater than 12 characters.

2 Development Stage

3 Testing Stage

4 Deployment Stage

In order to carry out the deployment of the service, you must first configure the environment variables in a `.evn_production` file, for this purpose it is suggested to use:

```
cp .env_example .env_production
```

In which, it is necessary to modify and change each one of the values of the environment variables following the suggestions and references in the sections 1.3 and 1.4.
With this, it makes use of the commands in `Makefile`, where the first one is the one related to the generation of SSL certificates, it makes use of:

```
make DOMAIN_NAME=your_domain ssl_certificate
```

Where the *your_domain* must match the value of DOMAIN_NAME in the environment variables file. Finally, the web service is deployed by means of:

```
make ENV=production deploy
```

In case any component has been updated, changed or stopped working, the following is used:

```
make ENV=production deploy_restart
```