

Tweeti Manual

Developer Edition

Laurens Müter

August 29, 2024

Copyright

MIT License

Copyright © 2024 Laurens Müter

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

Copyright	2
1 Background	4
1.1 Tags	4
2 Environment Variables	5
2.1 .env.dev	5
2.2 .env.prod	5
2.3 .env.prod.db	6
3 Docker-compose	7
3.1 Development Server	7
3.2 Production Server	7
4 Citation	8
Bibliography	9

1 Background

Tweeti is a web-based tool designed to facilitate the labeling of short messages, such as posts (formerly known as tweets). It includes features like shortcuts to speed up the labeling process and can be used on multiple devices, including laptops, phones, and tablets.

There is a Code Ocean capsule created for this project, to see the development code in action, see <https://codeocean.com/capsule/8199255/tree/v1>.

1.1 Tags

- labeling
- data labeling
- Twitter

2 Environment Variables

There are two files containing environment variables: `.env.dev` for development and `.env.prod` for production. In a production environment, it's recommended to run additional Docker containers, such as one for a database (Postgres in this case). The database container for production has its own set of environment variables, which are included in the `.env.prod.db` file.

2.1 `.env.dev`

```
DEBUG=1 # Show debug information
SECRET_KEY=foo # Dummy value for secret key, used for
DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1] # Allowed hosts set to local hosts
DJANGO_SUPERUSER_PASSWORD=Welcome123 # Set a superuser password to automate
```

Optional: Set production environment variables for testing (e.g. postgres database)

```
SQL_ENGINE=django.db.backends.postgresql
SQL_DATABASE=postgres_db
SQL_USER=hello_django
SQL_PASSWORD=hello_django
SQL_HOST=db
SQL_PORT=5432
DATABASE=postgres
```

Optional: Set production environment variables for testing (e.g. sendinblue email backend)

```
EMAIL_BACKEND=anymail.backends.sendinblue.EmailBackend
DEFAULT_FROM_EMAIL=info@example.com
EMAIL_API_URL=https://api.sendinblue.com/v3/
EMAIL_API_KEY='VeRySeCrEtKeY'
```

2.2 `.env.prod`

This file contains the same variables as `.env.dev`, with values tailored to the production environment.

```
DEBUG=0 # Do not show debug
SECRET_KEY=<long string of random characters> # Use a strong secret key
DJANGO_ALLOWED_HOSTS=localhost 127.0.0.1 [::1] <additional hosts> # Include the hosts
SQL_ENGINE=django.db.backends.postgresql # Using PostgreSQL
SQL_DATABASE=<DATABASE-NAME> # Database name, slugified
SQL_USER=<DATABASE-USER> # Database username
SQL_PASSWORD=<DATABASE-PASSWORD> # Database password
```

```
SQL_HOST=db
SQL_PORT=5432
DATABASE=postgres
EMAIL_BACKEND=<email backend>
DEFAULT_FROM_EMAIL=<from address>
EMAIL_API_URL=<API URL>
EMAIL_API_KEY=<API key>
```

```
# Default database
# Default database
# Database type, sl
# See https://pypi
# This address wil
# API URL from your
# API key from your
```

2.3 .env.prod.db

```
POSTGRES_USER=<DATABASE-USER>
POSTGRES_PASSWORD=<DATABASE-PASSWORD>
POSTGRES_DB=<DATABASE-NAME>
```

3 Docker-compose

To build the Docker containers and run the project, use Docker Compose. There are separate configurations for production and development.

3.1 Development Server

Run the following commands to start Docker Compose in a development environment:

```
$ docker-compose -f docker-compose.yml up -d --build
$ docker-compose -f docker-compose.yml exec web python manage.py migrate --noinput
```

3.2 Production Server

Run the following commands to start Docker Compose in a production environment:

```
$ docker-compose -f docker-compose.prod.yml up -d --build
$ docker-compose -f docker-compose.prod.yml exec web python manage.py migrate --noinput
```

NOTE: In some setups, collecting static files might be necessary. However, this project uses a separate front end (see `tweeti_front_end`).

```
$ docker-compose -f docker-compose.prod.yml exec web python manage.py collectstatic --noinput
```

To create a superuser during the initial setup (you will be prompted for a username, email, and password), run:

```
$ docker-compose -f docker-compose.prod.yml exec web python manage.py createsuperuser
```

3.2.1 View logfiles

To view the log files of the Docker containers running Tweeti, use the following command:

```
$ docker-compose -f docker-compose.prod.yml logs -f
```

4 Citation

When using this work for a publication, you can use the following bib-tex for citation:

```
@misc{muter2024tweeti,  
  title = {Tweeti: X Post Annotation Tool, Optimised for Speed and Accuracy},  
  author = {L.H.F. M{"u"}ter},  
  journal = {SoftwareX},  
  doi = {10.24433/CO.8056110.v1},  
  howpublished = {\url{https://www.codeocean.com/}},  
  year = 2024,  
  month = {7},  
  version = {v1}  
}
```


Bibliography