Darksite API Documentation

Release v0.0.0

UNC Darkside

Jan 11, 2019

Contents

CHAPTER 1

Overview

The Darksite API is a content management system built specifically to manage content for UNC Darkside's website. The resources are exposed through a GraphQL endpoint which allows for querying exactly as much or as little information as required.

CHAPTER 2

Development Environment

As a Python project, there are certain steps we recommend for setting up your developer environment.

2.1 Recommended Setup

These steps will walk you through setting up our recommended developer environment.

2.1.1 Tooling

The following tools are required for our recommended environment setup:

1. Python 3.6
2. Pip
3. Pipenv

2.1.2 Source Code

The first step is to clone the source code from GitHub. You can clone the project using SSH or over HTTPS.

```
# Over SSH (Recommended)
git clone git@github.com:UNCDarkside/DarksiteAPI
```

Or,

```
# Over HTTPS
git clone https://github.com/UNCDarkside/DarksiteAPI
```

Once you have the source code, `cd` into the repository.

```
cd DarksiteAPI
```

2.1.3 Local Environment

Our project requires a few environment variables to work correctly. We can set these properties in a `.env` file that is read before executing any `pipenv` command.

```
# .env
DJANGO_DEBUG=true
DJANGO_MEDIA_ROOT=/path/to/your/clone/darksite/media
```

2.1.4 Install Dependencies

We use Pipenv to manage our dependencies. We recommend installing all the development requirements, but if you only want to run the project locally you can omit the `--dev` flag.

```
pipenv install --dev
```

2.1.5 Code Style

We use a combination of *black* and *flake8* to ensure the code style for the project is consistent. These tools are used by our CI process to check every pushed commit. To ensure code is well-formatted before we push it, we use the *pre-commit* tool. After installing it, it will format your code on every commit.

```
pipenv run pre-commit install
```

2.2 Running the Project

To run the application locally, run the database migrations and then start the application.

```
pipenv run darksite/manage.py migrate
pipenv run darksite/manage.py runserver
```

This will launch the application locally on `http://localhost:8000`.

Note: You must run the `migrate` command whenever additional migrations are added to the source. The `runserver` command will log a warning if you forget to do this, and any new logic relying on the presence of the new tables will cause crashes.

2.2.1 First Time Setup

The first time you run the project, you will want to create a super-user that you can use to access the admin interface.

```
pipenv run darksite/manage.py createsuperuser
```

2.3 Testing

The project has a comprehensive test suite to ensure correctness. Tests are run using pytest. For more information on our testing process, see the *Testing* page.

```
# Unit Tests
pipenv run pytest darksite/

# Functional Tests
pipenv run pytest darksite/functional_tests/
```

CHAPTER 3

Testing

This project has a comprehensive test suite used to ensure correctness. The test suite is run using pytest.

3.1 Functional Tests

Functional, or end-to-end (E2E), tests are arguably the most important tests we have in the project. As a general rule, we write a functional test to cover each use case a consumer of the API would have. The role of these tests is to ensure that these use cases are runnable while making no assumptions about the internal workings of the application. Because these tests do not have any knowledge of internal workings, they are not designed to cover every edge case.

Note: The required functional tests for a feature can typically be pulled from the "Acceptance Tests" section of the related GitHub issue.

These E2E tests are placed in the `darksite/functional_tests/` directory. This directory is organized loosely by "feature" and is not necessarily correlated to the organization of the apps that make up the Django project.

Since these tests take much longer to run than the unit tests, they must be targeted specifically in order to run them.

```
pipenv run pytest darksite/functional_tests/
```

3.2 Unit Tests

Each component of the project also has its own unit tests. These tests are much smaller than the functional tests and cover an isolated component. Because these tests require much less setup than the E2E tests, they are useful for covering edge cases and other scenarios that may be hard to set up without some knowledge of the application implementation.

Unit tests are located in the `test/` directory of each app, and are organized such that there is a directory for each module (`.py` file) and each function or class has a test file.

To run the unit tests, point `pytest` at the `darksite` directory and it will find and run all the unit tests. You can also execute a specific test file or only tests that match a certain name.

```
# All unit tests
pipenv run pytest darksite/

# Unit tests from a specific file
pipenv run pytest darksite/account/test/models/test_user_model.py

# Tests that have "user" in their name
pipenv run pytest -k user darksite/
```

CHAPTER 4

Darksite API

API Root https://api.uncdarkside.cdriehuys.com/graphql/

Documentation https://darksiteapi.readthedocs.io

4.1 About

This project is the GraphQL API that powers UNC Darkside's website.

4.2 License