### 2.3.1 Структура фреймворка

Фреймворк Staticflow будет реализован как Python-проект с модульной архитектурой. Основные компоненты системы будут организованы в следующие директории:

* core/ - ядро фреймворка, содержащее основные классы и интерфейсы,
* parsers/ - модули для обработки различных форматов контента,
* admin/ - компоненты административной панели,
* plugins/ - система плагинов и встроенные плагины,
* utils/ - вспомогательные утилиты и общие функции,
* deploy/ - модуль развертывания,
* cli/ - интерфейс командной строки,
* templates/ - система шаблонов.

### 2.3.2 Реализация ядра фреймворка

Начнем с реализации основных компонентов фреймворка. Первым шагом создадим ключевой класс Config для управления конфигурацией. Этот класс обеспечивает загрузку и сохранение настроек из файлов формата TOML, управляет многоязычностью, а также валидирует конфигурацию. Класс конфигурации обеспечивает удобный интерфейс для доступа к настройкам через методы get и set, а также обеспечивает их сохранение в файл.

На рисунке 2.1 показаны функции для управления конфигурационным файлом.

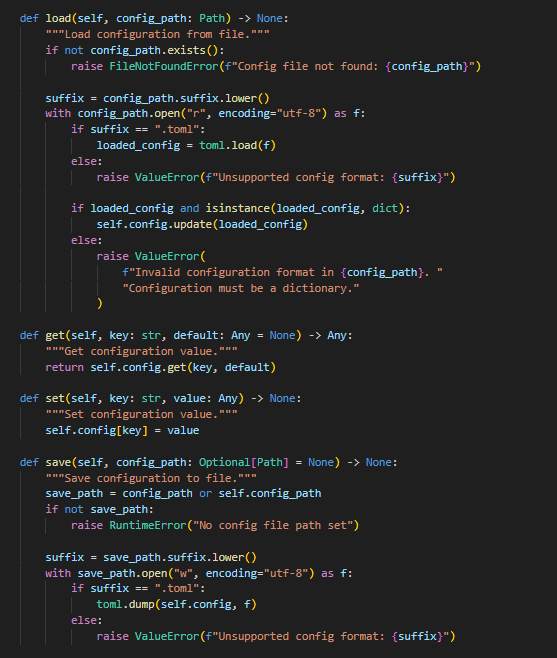


Рисунок 2.1. Функции управления конфигурационным файлом класса Config

За процесс генерации статического сайта отвечает класс Engine. Он координирует работу всех остальных компонентов системы и управляет процессом сборки. На рисунках 2.2, 2.3, 2.4 и 2.5 представлены функции для процесса сборки сайтов, обработку страниц, управления статическими файлами и управления плагинами соответственно.

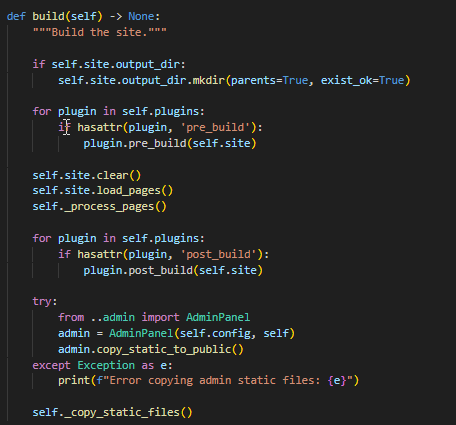


Рисунок 2.2 Функция сборки сайта

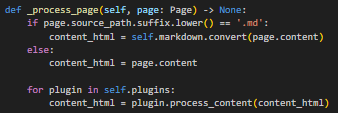


Рисунок 2.3 Функция обработки страницы

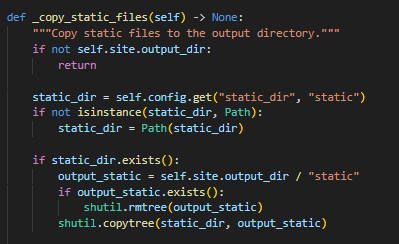


Рисунок 2.4 Функция обработки статических файлов

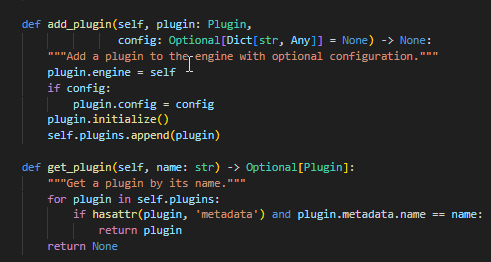


Рисунок 2.5 Функции управления плагинами

Следующий фундаментальный компонент фреймворка - класс Page, который предоставляет отдельную страницу сайта и управляет ее содержимым и метаданными. Функции обработки страницы представлены на рисунке 2.6.

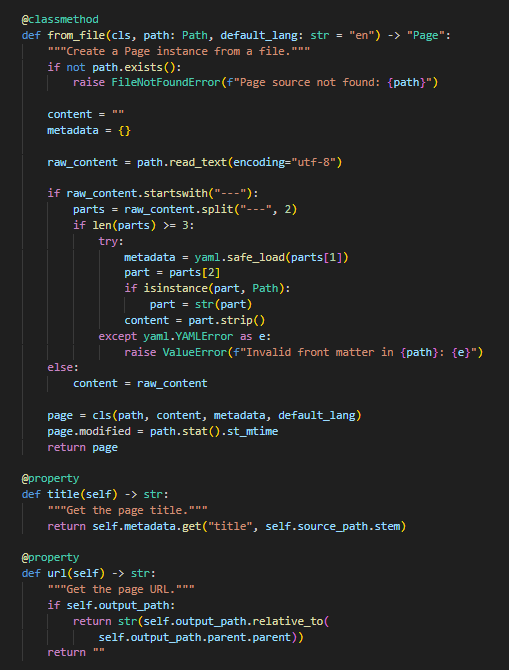


Рисунок 2.6 Функции управления страницей

Класс Router представляет собой компонент системы маршрутизации фреймворка StaticFlow, реализующий паттерн "URL Router". Он обеспечивает абстракцию между структурой контента и его URL-представлением через систему шаблонов, где каждый тип контента имеет предопределенный паттерн URL.

Система маршрутизации основана на принципе "Convention over Configuration", предоставляя стандартные шаблоны для различных типов контента (страницы, посты, теги и т.д.), которые могут быть переопределены через конфигурацию. Паттерны URL реализуют механизм подстановки переменных, позволяющий формировать структурированные URL-адреса на основе метаданных контента.

Класс Router поддерживает многоязычность через систему префиксов языков и обеспечивает разделение логического представления контента (URL\_PATTERNS) от его физического хранения (SAVE\_AS\_PATTERNS). Это позволяет гибко настраивать структуру URL без изменения базовой логики маршрутизации, сохраняя при этом слабую связанность компонентов системы.

На рисунках 2.7, 2.8, 2.9 изображены следующие ключевые части кода класса Router: определение паттернов URL\_PATTERNS и SAVE\_AS\_PATTERNS, основной метод генерации URL, метод форматирования шаблонов.

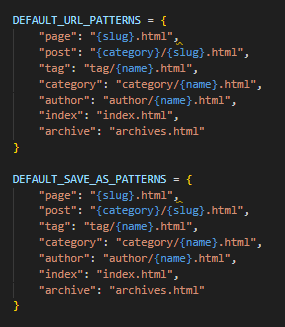


Рисунок 2.7 Определение паттернов URL\_PATTERNS и SAVE\_AS\_PATTERNS

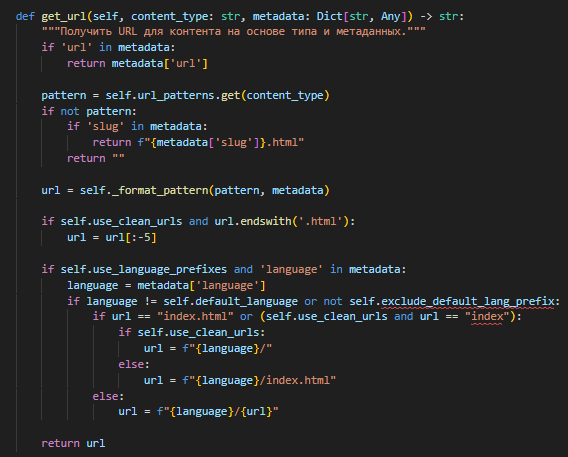


Рисунок 2.8 Основной метод генерации URL



Рисунок 2.9 Метод форматирования шаблонов

Класс Server представляет собой ключевой компонент фреймворка StaticFlow, реализующий функциональность веб-сервера для разработки и обслуживания статических сайтов. В его основе лежит асинхронный веб-фреймворк aiohttp, который обеспечивает эффективную обработку HTTP-запросов.

Основная задача класса Server заключается в организации взаимодействия между различными компонентами системы: маршрутизацией, шаблонизацией, обработкой статических файлов и административной панелью. Особое внимание уделено режиму разработки, который включает автоматическую валидацию структуры проекта и пересборку сайта при изменениях.

На рисунках 2.10 и 2.11 представлены настройка маршрутизации и обработка запросов соответственно.

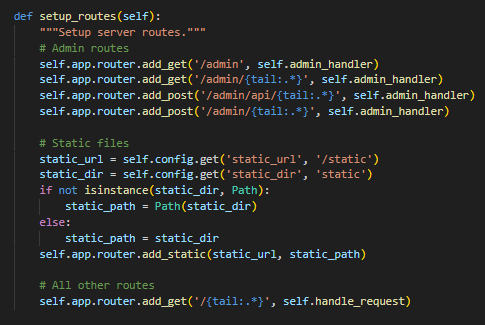


Рисунок 2.10 Метод настройки маршрутизации

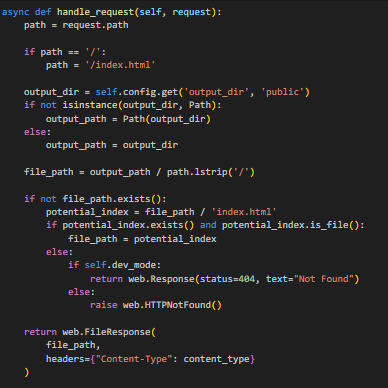


Рисунок 2.11 Метод обработка запросов

## 2.4 Тестирование

## 2.5 Разработка документации