Выпускная квалификационная работа

Разработка фреймворка для генерации статических сайтов

Обучающейся 4 курса
очной формы обучения
Крючковой Анастасии Сергеевны
Руководитель выпускной квалификационной работы:
кандидат физико-математических наук,
доцент кафедры ИТиЭО
Жуков Николай Николаевич

Цель, предмет исследования и задачи

Цель дипломного проекта – разработка фреймворка для генерации статических сайтов.

Предметом исследования является процесс разработки и оптимизации фреймворка для генерации статических сайтов

Задачи:

1)Анализ существующих решений: Изучить текущие фреймворки для генерации статических сайтов, их преимущества и недостатки.

2)Определение требований: Сформулировать функциональные и нефункциональные требования к новому фреймворку.

3)Проектирование архитектуры: Разработать архитектуру фреймворка, обеспечивающую гибкость и расширяемость.

4)Реализация прототипа: Создать прототип фреймворка с базовыми функциями генерации статических сайтов.

5)Интеграция с веб-технологиями: Обеспечить поддержку современных веб-технологий и инструментов.

6)Тестирование и оптимизация: Провести тестирование производительности и оптимизацию фреймворка.

7)Документация и обучение: Подготовить документацию и обучающие материалы для пользователей и разработчиков.

Назначение и преимущество фреймворков для генерации статических сайтов

Оптимизация производительности

Информационная безопасность

Процесс разработки

Масштабируемость

Экономическая эффективность

Оптимизация (SEO)

Интеграции с актуальными инструментами разработки и системами непрерывной интеграции и доставки (CI/CD)

Характеристика	Pelican	MkDocs	Nikola	Sphinx	Lektor
Основное назначение	Блоги и новостные сайты	Документация	Универсальный	Техническая документация	Универсальный
Формат контента	Markdown, reStructuredText, AsciiDoc	Markdown	Markdown, reStructuredText, HTML, Jupyter Notebooks	reStructuredText, Markdown	Markdown, HTML
Шаблонизатор	Jinja2	Jinja2	Mako, Jinja2	Jinja2	Jinja2
Поддержка тем	Да	Да	Да	Да	Да
Встроенные темы	3+	2+	10+	1+	5+
Поддержка плагинов	Да	Да	Да	Да	Да
Многоязычность	Да	Да	Да	Да	Да
Поддержка тегов	Да	Да	Да	Да	Да
Поддержка категорий	Да	Нет	Да	Нет	Да
Поддержка RSS	Да	Нет	Да	Нет	Да
Поддержка Sitemap	Да	Да	Да	Да	Да
Поддержка MathJax	Через плагины	Да	Да	Да	Через плагины
Поддержка Code Highlighting	Да	Да	Да	Да	Да
Поддержка изображений	Да	Да	Да	Да	Да
Поддержка видео	Да	Да	Да	Да	Да
Поддержка LaTeX	Через плагины	Да	Да	Да	Через плагины
Поддержка CI/CD	Да	Да	Да	Да	Да
Активность разработки	Средняя	Высокая	Средняя	Высокая	Низкая
Сложность настройки	Средняя	Низкая	Средняя	Средняя	Низкая

Анализ требований к фреймворку

На основе анализа самых популярных фреймворков для генерации статических сайтов на Python можно сформулировать функциональные и нефункциональные требования к разработке. Данные требования основаны на исследовании существующих решений и отражают современные тенденции в области веб-разработки.

Архитектурное проектирование

Проектирование движка

Проектирование парсеров

Проектирование CLI

Проектирование панели администратора

Проектирование блочного редактора в панеле администратора

Проектирование системы шаблонов

Проектирование системы маршутизации

Проектирование модуля развертывания

Проектирование системы плагинов

Проектирование системы CDN

Проектирование системы локализации

Программная реализация

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.

Реализация ядра фреймворка

Методы управления конфигурационным файлом класса Config

```python
def load(self, config_path: Path) -> None:
    """Load configuration from file."""
    if not config_path.exists():
        raise FileNotFoundError(f"Config file not found: {config_path}")

    suffix = config_path.suffix.lower()
    with config_path.open("r", encoding="utf-8") as f:
        if suffix == ".toml":
            loaded_config = toml.load(f)
        else:
            raise ValueError(f"Unsupported config format: {suffix}")

        if loaded_config and isinstance(loaded_config, dict):
            self.config.update(loaded_config)
        else:
            raise ValueError(
                f"Invalid configuration format in {config_path}. "
                "Configuration must be a dictionary."
            )

def get(self, key: str, default: Any = None) -> Any:
    """Get configuration value."""
    return self.config.get(key, default)

def set(self, key: str, value: Any) -> None:
    """Set configuration value."""
    self.config[key] = value

def save(self, config_path: Optional[Path] = None) -> None:
    """Save configuration to file."""
    save_path = config_path or self.config_path
    if not save_path:
        raise RuntimeError("No config file path set")

    suffix = save_path.suffix.lower()
    with save_path.open("w", encoding="utf-8") as f:
        if suffix == ".toml":
            toml.dump(self.config, f)
        else:
            raise ValueError(f"Unsupported config format: {suffix}")
```

Реализация ядра фреймворка

Методы сборки сайта

```python
def build(self) -> None:
    """Build the site."""

    if self.site.output_dir:
        self.site.output_dir.mkdir(parents=True, exist_ok=True)

    for plugin in self.plugins:
        if hasattr(plugin, 'pre_build'):
            plugin.pre_build(self.site)

    self.site.clear()
    self.site.load_pages()
    self._process_pages()

    for plugin in self.plugins:
        if hasattr(plugin, 'post_build'):
            plugin.post_build(self.site)

    try:
        from ..admin import AdminPanel
        admin = AdminPanel(self.config, self)
        admin.copy_static_to_public()
    except Exception as e:
        print(f"Error copying admin static files: {e}")

    self._copy_static_files()
```

Реализация ядра фреймворка

Методы управления плагинами

```python
def add_plugin(self, plugin: Plugin,
               config: Optional[Dict[str, Any]] = None) -> None:
    """Add a plugin to the engine with optional configuration."""
    plugin.engine = self
    if config:
        plugin.config = config
    plugin.initialize()
    self.plugins.append(plugin)

def get_plugin(self, name: str) -> Optional[Plugin]:
    """Get a plugin by its name."""
    for plugin in self.plugins:
        if hasattr(plugin, 'metadata') and plugin.metadata.name == name:
            return plugin
    return None
```

Реализация ядра фреймворка

Методы сборки сайта

```python
def build(self) -> None:
    """Build the site."""

    if self.site.output_dir:
        self.site.output_dir.mkdir(parents=True, exist_ok=True)

    for plugin in self.plugins:
        if hasattr(plugin, 'pre_build'):
            plugin.pre_build(self.site)

    self.site.clear()
    self.site.load_pages()
    self._process_pages()

    for plugin in self.plugins:
        if hasattr(plugin, 'post_build'):
            plugin.post_build(self.site)

    try:
        from ..admin import AdminPanel
        admin = AdminPanel(self.config, self)
        admin.copy_static_to_public()
    except Exception as e:
        print(f"Error copying admin static files: {e}")

    self._copy_static_files()
```

Реализация ядра фреймворка

Заключение

В результате проведения работы были достигнуты все поставленные задачи, и был успешно разработан фреймворк для генерации статических сайтов. Фреймворк Staticflow показал себя как эффективный способ разработки статических сайтов для разработчиков с разным опытом владения языка программирования Python.