API документация - SSV Monetization Tool

📚 Содержание

Обзор

SSV Monetization Tool предоставляет программный интерфейс (API) для интеграции с другими инструментами экосистемы SSVproff.

Основные компоненты

- **utils/** — вспомогательные утилиты
- **modules/** — основная бизнес-логика
- **REST API** — веб-интерфейс для интеграции (в разработке)

Модуль utils

utils.logger

Описание: Централизованное логирование для всех модулей.

```
setup_logger(name: str, log_file: str = "monetization.log", level: str = "INFO") -> logging.Logger
```

Создаёт и настраивает logger для модуля.

Параметры:
- `name` (str) — имя logger'а (обычно `__name__`)
- `log_file` (str, optional) — путь к файлу логов (по умолчанию: `"monetization.log"`)
- `level` (str, optional) — уровень логирования (`"DEBUG"`, `"INFO"`, `"WARNING"`, `"ERROR"`) (по умолчанию: `"INFO"`)

Возвращает:
- `logging.Logger` — настроенный logger

Пример:

```python
from utils.logger import setup_logger

logger = setup_logger(__name__, log_file="custom.log", level="DEBUG")
logger.info("Инструмент запущен")
logger.debug("Отладочная информация")
```

---

utils.config_loader

Описание: Загрузка и валидация конфигурационного файла.

`load_and_validate_config(config_path: str) -> Dict[str, Any]`

Загружает конфигурацию из YAML-файла и проверяет её корректность.

Параметры:
- `config_path` (str) — путь к файлу конфигурации (например, `"monetization_config.yaml"`)

Возвращает:
- `Dict[str, Any]` — словарь с конфигурацией

Исключения:
- `FileNotFoundError` — если файл не найден
- `ValueError` — если конфигурация некорректна
- `yaml.YAMLError` — если ошибка парсинга YAML

Пример:

```python
from utils.config_loader import load_and_validate_config

try:
    config = load_and_validate_config("monetization_config.yaml")
    print(f"Стратегия: {config['monetization']['strategy']}")
except ValueError as e:
    print(f"Ошибка конфигурации: {e}")
```

---

utils.disclaimer_generator

Описание: Генерация дисклеймеров для различных типов монетизации.

`generate_affiliate_disclaimer() -> str`

Генерирует дисклеймер для партнёрских ссылок.

Возвращает:
- `str` — текст дисклеймера

Пример:

```python
from utils.disclaimer_generator import generate_affiliate_disclaimer

disclaimer = generate_affiliate_disclaimer()
print(disclaimer)
# Вывод: "⚠️ Disclaimer: Некоторые ссылки в этом описании являются партнёрскими..."
```

`generate_sponsorship_disclaimer(sponsor_name: str, config: Dict[str, Any]) -> str`

Генерирует дисклеймер для спонсорского контента.

Параметры:
- `sponsor_name` (str) — имя спонсора
- `config` (Dict[str, Any]) — конфигурация монетизации

Возвращает:
- `str` — текст дисклеймера

Пример:

```python
from utils.disclaimer_generator import generate_sponsorship_disclaimer

disclaimer = generate_sponsorship_disclaimer("Medical Equipment Inc.", config)
print(disclaimer)
# Вывод: "Контент частично спонсирован Medical Equipment Inc."
```

`generate_premium_disclaimer(config: Dict[str, Any]) -> str`

Генерирует призыв к действию для премиум-контента.

Параметры:
- `config` (Dict[str, Any]) — конфигурация монетизации

Возвращает:
- `str` — текст CTA

Пример:

```python
from utils.disclaimer_generator import generate_premium_disclaimer

cta = generate_premium_disclaimer(config)
print(cta)
# Вывод: "🎓 Получите полный доступ к курсу на https://ssvnauka.com/premium"
```

Модуль modules

modules.strategy_planner

Описание: Планирование действий монетизации на основе стратегии.

`determine_actions_for_strategy(strategy: str, config: Dict[str, Any]) -> List[str]`

Определяет список действий монетизации на основе выбранной стратегии.

Параметры:
- `strategy` (str) — стратегия монетизации (`"full"`, `"partial"`, `"masked"`, `"hidden"`)
- `config` (Dict[str, Any]) — конфигурация монетизации

Возвращает:
- `List[str]` — список действий для выполнения

Возможные действия:
- `"inject_affiliate_links"` — внедрить партнёрские ссылки
- `"add_affiliate_disclaimer"` — добавить дисклеймер партнёрских ссылок
- `"inject_sponsorship"` — внедрить спонсорский контент
- `"add_sponsorship_disclaimer"` — добавить дисклеймер спонсорства
- `"add_premium_cta"` — добавить призыв к действию для премиум-контента

Пример:

```
from modules.strategy_planner import determine_actions_for_strategy

config = load_and_validate_config("monetization_config.yaml")
actions = determine_actions_for_strategy("masked", config)
print(actions)
# Вывод: ['inject_affiliate_links', 'add_premium_cta']
```

---

modules.content_injector

Описание: Внедрение элементов монетизации в контент.

`inject_monetization_elements(content: Dict[str, Any], actions: List[str], config: Dict[str, Any]) -> Dict[str, Any]`

Внедряет элементы монетизации в контент согласно списку действий.

Параметры:
- `content` (Dict[str, Any]) — исходный контент с полями:
- `id` (str) — уникальный идентификатор
- `title` (str) — заголовок
- `description` (str) — описание
- `actions` (List[str]) — список действий для выполнения
- `config` (Dict[str, Any]) — конфигурация монетизации

Возвращает:
- `Dict[str, Any]` — модифицированный контент

Пример:

```python
from modules.content_injector import inject_monetization_elements

content = {
    'id': 'video_001',
    'title': 'Техника операции',
    'description': 'Используем хирургические инструменты...'
}

actions = ['inject_affiliate_links', 'add_affiliate_disclaimer']
result = inject_monetization_elements(content, actions, config)

print(result['description'])
# Вывод: "Используем хирургические инструменты (https://store.com/tools?ref=ssv)...
#         ⚠️ Disclaimer: Некоторые ссылки являются партнёрскими..."
```

---

modules.compliance_checker

Описание: Проверка контента на соответствие политикам платформ.

`check_youtube_description_compliance(description: str) -> List[str]`

Проверяет описание видео на соответствие политикам YouTube.

Параметры:
- `description` (str) — текст описания видео

Возвращает:
- `List[str]` — список обнаруженных проблем (пустой список, если проблем нет)

Проверяемые параметры:
- Длина описания (макс. 5000 символов)
- Количество ссылок (макс. 15)
- Спам-паттерны
- Избыточные CAPS LOCK
- Избыточные восклицательные знаки

Пример:

```python
from modules.compliance_checker import check_youtube_description_compliance

description = "КУПИТЬ СЕЙЧАС!!! СКИДКА 90%!!!"
issues = check_youtube_description_compliance(description)

if issues:
    print("Проблемы:")
    for issue in issues:
        print(f"  - {issue}")
# Вывод:
# Проблемы:
#   - Excessive caps detected
#   - Excessive exclamation marks
#   - Spam patterns detected
```

`check_amazon_kdp_compliance(description: str) -> List[str]`

Проверяет описание книги на соответствие политикам Amazon KDP.

Параметры:
- `description` (str) — текст описания книги

Возвращает:
- `List[str]` — список обнаруженных проблем

Проверяемые параметры:
- Длина описания (макс. 4000 символов)
- Внешние ссылки (запрещены)
- Контактная информация (запрещена)

Пример:

```python
from modules.compliance_checker import check_amazon_kdp_compliance

description = "Свяжитесь со мной: email@example.com или https://mysite.com"
issues = check_amazon_kdp_compliance(description)

if issues:
    print("Проблемы Amazon KDP:")
    for issue in issues:
        print(f"  - {issue}")
```

`check_general_compliance(text: str) -> List[str]`

Общая проверка текста на соответствие базовым требованиям.

Параметры:
- `text` (str) — текст для проверки

Возвращает:
- `List[str]` — список обнаруженных проблем

---

modules.analytics_tracker

Описание: Аналитика и отслеживание метрик монетизации.

`generate_unique_affiliate_link(base_url: str, content_id: str, source: str, medium: str = "description") -> str`

Генерирует уникальную партнёрскую ссылку с UTM-метками.

Параметры:
- `base_url` (str) — базовая URL ссылки
- `content_id` (str) — идентификатор контента
- `source` (str) — источник трафика (например, `"youtube"`, `"amazon_kdp"`)
- `medium` (str, optional) — тип размещения (по умолчанию: `"description"`)

Возвращает:
- `str` — уникальная ссылка с UTM-параметрами

Пример:

```python
from modules.analytics_tracker import generate_unique_affiliate_link

link = generate_unique_affiliate_link(
    base_url="https://amazon.com/product",
    content_id="video_001",
    source="youtube",
    medium="description"
)

print(link)
# Вывод: https://amazon.com/product?
utm_source=youtube&utm_medium=description&utm_campaign=video_001
```

`calculate_monetization_metrics(content: Dict[str, Any]) -> Dict[str, Any]`

Вычисляет метрики монетизации для контента.

Параметры:
- `content` (Dict[str, Any]) — контент с внедрёнными элементами монетизации

Возвращает:
- `Dict[str, Any]` — словарь с метриками:
- `description_length` (int) — длина описания
- `affiliate_links_count` (int) — количество партнёрских ссылок
- `has_disclaimer` (bool) — наличие дисклеймера
- `has_cta` (bool) — наличие CTA

Пример:

```python
from modules.analytics_tracker import calculate_monetization_metrics

metrics = calculate_monetization_metrics(result)
print(f"Длина описания: {metrics['description_length']}")
print(f"Партнёрских ссылок: {metrics['affiliate_links_count']}")
```

`prepare_monetization_report(strategy: str, methods: List[str], metrics: Dict[str, Any]) -> Dict[str, Any]`

Подготавливает отчёт о монетизации.

Параметры:
- `strategy` (str) — использованная стратегия
- `methods` (List[str]) — использованные методы
- `metrics` (Dict[str, Any]) — метрики контента

Возвращает:
- `Dict[str, Any]` — отчёт о монетизации

Пример:

```python
from modules.analytics_tracker import prepare_monetization_report

report = prepare_monetization_report(
    strategy="masked",
    methods=["affiliate_links", "premium_content"],
    metrics=metrics
)

print(f"Стратегия: {report['strategy']}")
print(f"Методы: {report['methods_used']}")
```

`track_monetization_event(event_type: str, content_id: str, metadata: Dict[str, Any]) -> None`

Отслеживает событие монетизации для аналитики.

Параметры:
- `event_type` (str) — тип события (например, `"content_processed"`, `"link_clicked"`)
- `content_id` (str) — идентификатор контента
- `metadata` (Dict[str, Any]) — дополнительные данные о событии

Пример:

```python
from modules.analytics_tracker import track_monetization_event

track_monetization_event(
    event_type='content_processed',
    content_id='video_001',
    metadata={
        'strategy': 'masked',
        'actions': ['inject_affiliate_links'],
        'timestamp': '2025-10-22T10:00:00'
    }
)
```

REST API для интеграции

Endpoints (в разработке)

`POST /api/v1/monetize`

Применяет монетизацию к контенту.

Request Body:

```
{
  "content": {
    "id": "video_001",
    "title": "Техника операции",
    "description": "Описание контента..."
  },
  "strategy": "masked",
  "methods": ["affiliate_links", "premium_content"]
}
```

Response:

```
{
  "success": true,
  "result": {
    "id": "video_001",
    "title": "Техника операции",
    "description": "Модифицированное описание с партнёрскими ссылками...",
    "metrics": {
      "description_length": 350,
      "affiliate_links_count": 2,
      "has_disclaimer": false,
      "has_cta": true
    }
  }
}
```

`GET /api/v1/compliance/youtube`

Проверяет соответствие описания политикам YouTube.

Query Parameters:

- `description` (string) — текст описания

Response:

```
{
  "compliant": false,
  "issues": [
    "Excessive caps detected",
    "Too many links (18), YouTube limit is 15"
  ]
}
```

Примеры использования

Пример 1: Простая интеграция

```python
from utils.config_loader import load_and_validate_config
from modules.strategy_planner import determine_actions_for_strategy
from modules.content_injector import inject_monetization_elements

# Загрузка конфигурации
config = load_and_validate_config("monetization_config.yaml")

# Контент
content = {
    'id': 'video_001',
    'title': 'Техника операции',
    'description': 'Используем современные инструменты...'
}

# Определение действий
actions = determine_actions_for_strategy(config['monetization']['strategy'], config)

# Применение монетизации
result = inject_monetization_elements(content, actions, config)

print(result['description'])
```

Пример 2: Полный pipeline

```python
from utils.config_loader import load_and_validate_config
from utils.logger import setup_logger
from modules.strategy_planner import determine_actions_for_strategy
from modules.content_injector import inject_monetization_elements
from modules.compliance_checker import check_youtube_description_compliance
from modules.analytics_tracker import calculate_monetization_metrics, pre-
pare_monetization_report

# Настройка
logger = setup_logger(__name__)
config = load_and_validate_config("monetization_config.yaml")

# Контент
content = {'id': 'video_001', 'title': 'Операция', 'description': '...'}

# Pipeline
logger.info("Starting monetization pipeline")

# 1. Определение действий
strategy = config['monetization']['strategy']
actions = determine_actions_for_strategy(strategy, config)
logger.info(f"Actions: {actions}")

# 2. Применение монетизации
result = inject_monetization_elements(content, actions, config)
logger.info("Monetization applied")

# 3. Проверка соответствия
issues = check_youtube_description_compliance(result['description'])
if issues:
    logger.warning(f"Compliance issues: {issues}")

# 4. Расчёт метрик
metrics = calculate_monetization_metrics(result)
logger.info(f"Metrics: {metrics}")

# 5. Генерация отчёта
report = prepare_monetization_report(strategy, config['monetization']['methods'], met-
rics)
logger.info(f"Report: {report}")

print("✅ Pipeline completed!")
```

Дополнительные ресурсы

- Руководство пользователя (USAGE.md)
- Примеры кода (EXAMPLES.md)
- GitHub репозиторий (https://github.com/Serg2206/ssv-monetization-tool)

Автор: Профессор С.В. Сушков
Проект: SSVproff-Ecosystem