**Документация, содержащая информацию, необходимую для эксплуатации экземпляра программного обеспечения**

**Модуль «Рекомендательная система для стриминговых сервисов»**

**г. Москва 2025**

**Содержание**

[1. Введение 3](#_heading=h.h9vqitmibr9s)

[2. Запуск модуля 3](#_heading=h.a2lao0ocdbme)

[2.1. Предварительные требования 3](#_heading=h.ez42fugav00y)

[2.2. Установка и запуск 3](#_heading=h.vb1m1fkmxp6j)

[3. Основные функции модуля 3](#_heading=h.vwnz9innfno7)

[4. Работа с системой 4](#_heading=h.xii9ojd8za7h)

[5. Алгоритмы формирования тегов 4](#_heading=h.esq55hpfauk0)

[6. Алгоритм формирования ответа на запрос 5](#_heading=h.nilgvxm63uw3)

[7. Масштабируемость и производительность 5](#_heading=h.4zck84w2xiof)

[8. Завершение работы 5](#_heading=h.ypxfbzg5qwyf)

##

## **1. Введение**

Модуль "Рекомендательная система для стриминговых сервисов" представляет собой программное решение, основанное на алгоритмах машинного обучения и анализа данных, предназначенное для персонализированного подбора видеоконтента. Система учитывает предпочтения пользователей, историю просмотров, реакции на контент и общие тренды для формирования релевантных рекомендаций.

## **2. Запуск модуля**

### 2.1. Предварительные требования

Перед запуском модуля необходимо:

* Убедиться, что серверная инфраструктура развернута (Kubernetes, Docker, Nginx).
* Проверить доступность баз данных (PostgreSQL, Redis, ClickHouse, OpenSearch).
* Настроить интеграцию с платформами, использующими рекомендации (например, Rutube).

### 2.2. Установка и запуск

Модуль устанавливается и настраивается как любое API-приложение. Установка производится с использованием контейнеризации (Docker/Kubernetes) и включает развертывание всех зависимых сервисов.

Запуск осуществляется с помощью контейнеризации Docker/Kubernetes:

kubectl apply -f recommendation-system-deployment.yaml

Или в случае локального развертывания:

docker-compose up -d

После запуска необходимо проверить доступность API:

curl -X GET http://localhost:8000/api/v2/rec/status

Если система запущена корректно, в ответ будет получен статус 200 OK.

## **3. Основные функции модуля**

Персонализированные рекомендации

Анализ предпочтений пользователей.

Подбор видео на основе истории просмотров.

Категорийные рекомендации

Учет жанров и тематики контента.

Коллаборативные рекомендации

Подбор контента на основе схожих пользователей.

Контекстные рекомендации

Формирование рекомендаций, основанных на текущем видео.

Трендовые рекомендации

Выбор популярного контента по актуальным темам.

## **4. Работа с системой**

Запрос персонализированных рекомендаций

Пример запроса к API:

curl -X POST http://localhost:8000/api/v2/rec/ \

 -H "Content-Type: application/json" \

 -d '{"user\_id": "12345", "limit": 10}'

Ответ сервера:

{

 "version": "xclip\_1000\_2023\_08\_14",

 "total": 10,

 "payload": [

 {"content\_id": "abcd1234", "score": 0.89},

 {"content\_id": "xyz7890", "score": 0.75}

 ]

}

Логирование и мониторинг

Для анализа работы модуля используются:

* **Prometheus** – сбор метрик.
* **Grafana** – визуализация данных.
* **Grafana Loki** – логирование событий.

## **5. Алгоритмы формирования тегов**

Система анализирует метаданные видео (название, описание, категории) и поведенческие данные пользователей (запросы, лайки) для присвоения тегов. Используемые методы:

* NLP (Natural Language Processing) – обработка текста и извлечение ключевых слов.
* Кластеризация – объединение схожих тегов.
* Семантический анализ – анализ взаимосвязей между терминами.

## **6. Алгоритм формирования ответа на запрос**

Сбор данных

* Извлечение истории просмотров пользователя.
* Определение предпочтений на основе взаимодействия с контентом.

Применение алгоритмов

* Коллаборативная фильтрация – анализ схожих пользователей.
* Контентная фильтрация – подбор схожего контента.
* Машинное обучение – предсказание интересов.

Генерация списка рекомендаций

* Агрегация оценок по алгоритмам.
* Сортировка видео по релевантности.
* Формирование итогового списка рекомендаций.

## **7. Масштабируемость и производительность**

Автоматическое масштабирование

* Использование Kubernetes для горизонтального масштабирования.
* Балансировка нагрузки через Yandex Application Load Balancer.

Оптимизация запросов

* Кэширование данных в Redis.
* Обработка запросов в потоковом режиме через Kafka.
* Хранение больших объемов данных в ClickHouse.

A/B тестирование

* Оценка эффективности рекомендаций через экспериментальные группы.
* Анализ метрик вовлеченности (время просмотра, количество кликов).

## **8. Завершение работы**

Для корректного завершения работы модуля необходимо:

kubectl delete -f recommendation-system-deployment.yaml

Или для локального окружения:

docker-compose down

После остановки сервисов проверить состояние процессов:

ps aux | grep recommendation-system

При необходимости завершить зависшие процессы:

kill -9 <PID>