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

**«Поисковый модуль для стриминговых сервисов»**

[1. Введение 3](#_nsx2269v8x8z)

[2. Установка программы 3](#_kbu5ezngwh5i)

[2.1. Требования к окружению 3](#_80551mykzuiy)

[2.2. Процесс установки 3](#_qg9wzcaidvlo)

[3. Начало работы 3](#_cp8zups7sm90)

[4. Эксплуатация модуля 4](#_outumzjck7if)

[4.1. Основные функциональные возможности 4](#_khglh7ia17wz)

[4.2. Основные сценарии поиска 4](#_1b6796zh7fr2)

[4.3. Использование API для интеграции 5](#_u3o3cy42m9cr)

[5. Интеграция модуля в Rutube 6](#_11tobmlg5gvj)

[5.1 Сценарии поиска 6](#_s15nuqn9yvwa)

[5.2 Пользователи поискового модуля в Rutube 7](#_5kp9ajdw8hwt)

[5.3 Задачи поискового модуля в Rutube 8](#_173n5zny3cdo)

[6. Спецификация интеграционного API 8](#_aj18l9f2stnc)

[6.1. Описание принципов интеграции 8](#_n8cg406re4hn)

[6.2. Основные возможности API 8](#_2q1hp5tk1l7o)

[7. Спецификация API запросов 9](#_vib2a7s3yi75)

[7.1 Поиск по контенту 9](#_fz9df4azax6a)

[7.2 Обработка ошибок 10](#_vm7sg1xsyln)

[7.3. Безопасность 10](#_5phf1xh38lfr)

#

# **1. Введение**

Этот документ содержит информацию, необходимую для эксплуатации Поискового модуля для стриминговых сервисов. Модуль используется для предоставления пользователям эффективного поиска контента на видео-платформах. Система ориентирована на обработку больших объемов данных и учета предпочтений пользователей для формирования персонализированных результатов поиска.

# **2. Установка программы**

## 2.1. Требования к окружению

Для корректной работы поискового модуля необходимо следующее окружение:

Операционная система: Ubuntu или Debian.

Система виртуализации: Yandex Cloud (или аналогичные облачные сервисы).

Контейнеризация: Docker и Kubernetes.

Базы данных: PostgreSQL, OpenSearch, Redis, ClickHouse.

Языки программирования: Python.

Модели машинного обучения: PyTorch, scikit-learn, CatBoost, XGBoost и другие.

Программные платформы: Yandex Cloud, Kubernetes, Yandex Managed Services.. Использование поискового модуля в Rutube

## 2.2. Процесс установки

1. Установите Docker и настройте контейнеризацию.
2. Разверните Kubernetes для оркестрации контейнеров.
3. Настройте PostgreSQL, OpenSearch, Redis и ClickHouse для работы с данными.
4. Установите необходимые Python-библиотеки: scikit-learn, PyTorch, Optuna, CatBoost, XGBoost, и другие.
5. Настройте и подключите облачные сервисы, такие как Yandex Object Storage и Yandex Compute Cloud.

#

# **3. Начало работы**

#

Для запуска системы необходимо выполнить несколько подготовительных шагов:

Подготовка окружения:

1. Подготовьте контейнеры с базовыми операционными системами (например, Ubuntu или Debian).
2. Настройте кластер Kubernetes для развертывания контейнеров и распределения нагрузки.
3. Разверните облачные сервисы для хранения и обработки данных (например, Yandex Cloud).
4. Убедитесь, что все сервисы, включая базы данных, работают корректно.

Для запуска системы используйте команду в командной строке:

bash

Копировать

docker-compose up

Подключите API для интеграции с другими системами, включая внешний контент и авторизацию. Настройте параметры кеширования запросов для оптимизации работы поисковой системы.

## **4.** Эксплуатация модуля

Поисковый модуль предоставляет функциональность для быстрого и точного поиска контента на платформе. Пользователи могут искать видео, каналы, плейлисты, а также получать персонализированные рекомендации на основе предыдущих поисковых запросов и предпочтений.

###

### 4.1. Основные функциональные возможности

1. Поиск по разным типам сущностей. Поиск по видео, каналам, персонам и другим сущностям.
2. Поиск по контенту. Поддержка различных типов контента: UGC, фильмы, сериалы, музыкальные клипы и другие.
3. Автодополнение и исправление ошибок. Поиск с предложением вариантов запросов в реальном времени и автоматическим исправлением ошибок.
4. Персонализация поиска. Модуль учитывает историю поиска и предпочтения пользователя для улучшения релевантности результатов.
5. Ранжирование результатов. Результаты поиска ранжируются на основе популярности контента и предпочтений пользователя.

### 4.2. Основные сценарии поиска

1. Поиск по видео: Пользователи могут найти видео по ключевым словам, тегам, названиям и описаниям.
2. Поиск по каналам: Возможность искать каналы по тематике, названию и описанию.
3. Поисковые подсказки: Автодополнение запросов и исправление опечаток в реальном времени.
4. Поиск внутри канала: Поиск видео или плейлистов в пределах конкретного канала.

### 4.3. Использование API для интеграции

Поисковый модуль предоставляет API для внешних интеграций, что позволяет интегрировать его с другими платформами и сервисами.

Пример запроса:

json

Копировать

POST /api/v2/search/

{

 "scenario": "search.test.online.random",

 "query": "Сериал Друзья",

 "user\_id": "675ceb78-3d03-41b4-af5e-907499dfb3c0",

 "limit": 10

}

Пример ответа:

json

Копировать

{

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

 "timestamp": "2024-04-11T16:27:07.468613",

 "total": 10,

 "payload": [

 {

 "content\_id": "a534f645-ce7b-4d34-b3a3-e7888ca88926",

 "score": 0.494,

 "type": "unknown"

 }

 ]

}

Обработка ошибок:

Если запрос не может быть выполнен, система вернет ошибку с кодом и описанием. Возможные коды ошибок включают:

* **400**: Неправильный запрос.
* **404**: Ресурс не найден.
* **500**: Внутренняя ошибка сервера.

# **5. Интеграция модуля в Rutube**

Поисковая система для стриминговых сервисов интегрирована в видео платформу Rutube, где используется в качестве основного инструмента для поиска и навигации по контенту на нескольких типах устройств.

Данный модуль интегрируется через API и используется для обработки поисковых запросов, формирования и ранжирования релевантных результатов, которые отображаются на видео платформе Rutube.

Поисковая система интегрирована в веб-версию rutube.ru для мобильных и desktop браузеров, а также в мобильные приложения на iOS и Android и в Smart TV (умные телевизоры).

## 5.1 Сценарии поиска

Поисковый модуль для стриминговых сервисов используется в нескольких основных сценариях поиска в видео платформе Rutube.

* Поиск по видео

Пользователи могут находить конкретные видео по ключевым словам, названиям, тегам или описаниям. Система учитывает релевантность, популярность и персонализированные предпочтения для выдачи наиболее подходящих результатов.

 

Рис.1 Поиск по каналам и поиск по видео

* Поиск по каналам

Возможность находить каналы по их названиям, тематике или описанию. Пользователи могут быстро перейти к интересующим их авторам или тематическим подборкам контента.

* Поисковые подсказки

Автодополнение запросов в реальном времени, исправление опечаток и предложение популярных или релевантных вариантов поиска, чтобы упростить и ускорить процесс ввода запросов.



Рис.2 Поисковые подсказки и поиск внутри канала

* Поиск внутри канала

Пользователи могут искать видео или плейлисты в рамках конкретного канала, что позволяет быстро находить нужный контент у любимых авторов или по определенной тематике.

## 5.2 Пользователи поискового модуля в Rutube

* Горячие пользователи

Это активные пользователи, которые регулярно используют поиск на платформе. Они формируют подробную историю поисковых запросов, просматривают результаты, взаимодействуют с найденным контентом (лайки, комментарии, подписки) и уточняют свои запросы. Их предпочтения и поведение хорошо известны системе, что позволяет предлагать им персонализированные и релевантные результаты поиска.

* Холодные пользователи

Это новые пользователи, которые только начали использовать платформу, или те, у кого еще нет истории поисковых запросов. Поскольку система не имеет данных об их предпочтениях, результаты поиска для них строятся на основе общих трендов, популярного контента, региональных предпочтений или контекста текущего запроса.

* Зарегистрировавшиеся пользователи

Пользователи, которые прошли регистрацию на платформе, предоставив свои данные (номер телефона или электронную почту). Они могут быть как горячими (активно используют поиск), так и холодными (новые пользователи). Регистрация позволяет системе лучше анализировать их поисковое поведение, учитывать историю запросов и предлагать более точные и персонализированные результаты.

* Незарегистрировавшиеся пользователи

Пользователи, которые еще не зарегистрировались на платформе. Они идентифицируются с помощью cookie или других технических методов. Незарегистрировавшиеся пользователи также могут быть как горячими (активно используют поиск), так и холодными (только начали взаимодействовать с платформой). Для них результаты поиска формируются на основе текущего запроса, популярного контента и общих трендов.

## **5.3 Задачи поискового модуля в Rutube**

* Удержание пользователя за счет быстрого и точного поиска контента, повышение его интереса и вовлеченности благодаря релевантным результатам, соответствующим запросам и предпочтениям.
* Улучшение пользовательского опыта в части навигации и поиска контента за счет предоставления персонализированных и удобных инструментов поиска, включая автодополнение, исправление ошибок и фильтрацию результатов.
* Повышение пользовательских метрик на сервисе — например, увеличение времени просмотра (watchtime) за счет быстрого нахождения интересующего контента и повышение возвращаемости пользователя (retention rate) благодаря удобству и эффективности поиска.

# **6. Спецификация интеграционного API**

## **6.1. Описание принципов интеграции**

«Поисковый модуль для стриминговых сервисов» предоставляет API для внешних интеграций с целью формирования персонализированной поисковой выдачи релевантного контента в ответ на запрос пользователя на крупных медиа платформах (Rutube или подобные платформы). Этот API осуществлять поиск на основе различных сценариев, а также персонализировать поисковую выдачу, учитывая основную мета информацию о контенте, его популярность и трендовость, предпочтения пользователей, историю просмотров и другие параметры.

##

## **6.2. Основные возможности API**

**Получение релевантных результатов:**

Возможность искать разные типы контента на основе разных сценариев - например, поисковые каналы, поиск по видео или поисковые подсказки.

Поддержка пагинации через параметры limit и offset.

Возможность использования кеширования запросов для оптимизации производительности.

**Персонализация поиска.** Учет идентификатора пользователя (user\_id) для формирования персонализированных результатов поиска.

**Кеширование:**

Настройка режима кеширования через заголовок x-cache-mode.

Возможность использования реальных данных в режиме реального времени (off), кеширования на основе параметров запроса (use\_request\_fields).

**Гибкость сценариев:**

Возможность указывать различные сценарии поиска через параметр scenario.

Поддержка запасных сценариев (fallback), если основной сценарий не может быть выполнен.

**Обработка ошибок:**

Поддержка различных HTTP-статусов для обработки ошибок (400, 401, 403, 404, 408, 422, 429, 495, 500, 503).

Возврат детализированных сообщений об ошибках в формате JSON.

#

# **7. Спецификация API запросов**

## 7.1 Поиск по контенту

**Запрос:** POST /api/v2/search/

**Параметры запроса:**

● x-request-id (опционально): Уникальный идентификатор запроса для отслеживания. Если не предоставлен, генерируется автоматически.

● x-cache-mode (опционально): Режим кеширования. Возможные значения:

● off — игнорировать кеш, использовать только real-time данные.

● use\_request\_fields — формировать ключ кеширования на основе параметров запроса.

**Тело запроса (JSON):**

● scenario (обязательно): Название сценария поиска.
Формат: [service].[source].[type].[algorithm].
Пример: search.test.online.random.

● query: Текстовый запрос пользователя.

● limit (опционально): Максимальное количество результатов в ответе. По умолчанию: 20.

● user\_id (опционально): Идентификатор пользователя для персонализации поиска.

● offset (опционально): Смещение для пагинации. По умолчанию: 0.

**Ответ (JSON):**

● version: Версия контента.

● cache: Режим кеширования, использованный для запроса.

● timestamp: Временная метка ответа.

● fallback: Список запасных сценариев, если они были использованы.

● total: Общее количество результатов в ответе.

● payload: Список найденного контента. Каждый элемент содержит:

● content\_id: Идентификатор контента.

● score: Оценка релевантности контента.

● type: Тип контента.

*Пример ответа:*

{

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

 "cache": "use\_request\_fields",

 "timestamp": "2024-04-11T16:27:07.468613",

 "fallback": [],

 "total": 10,

 "payload": [

 {

 "content\_id": "a534f645-ce7b-4d34-b3a3-e7888ca88926",

 "score": 0.494,

 "type": "unknown"

 },

 {

 "content\_id": "d0943291-8dc5-431d-b4bf-0be83b30a437",

 "score": 0.161,

 "type": "unknown"

 }

 ]

}

##

## 7.2 Обработка ошибок

API возвращает различные HTTP-статусы для индикации ошибок:

● 400: Неправильный запрос.

● 401: Отсутствие токена авторизации.

● 403: Запрет доступа.

● 404: Ресурс не найден.

● 408: Таймаут запроса.

● 422: Ошибка валидации данных.

● 429: Слишком много запросов.

● 495: Ошибка сертификата.

● 500: Внутренняя ошибка сервера.

● 503: Сервис временно недоступен.

*Каждая ошибка сопровождается JSON-объектом с деталями:*

{

 "details": "Упс, что то пошло не так...",

 "code": "P012345E"

}

## 7.3. Безопасность

API использует авторизацию через заголовок x-authorization-token. Токен должен быть передан в каждом запросе для доступа к защищенным endpoint'ам.

*Пример использования:*

POST /api/v2/search/ HTTP/1.1

Content-Type: application/json

x-authorization-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...

x-cache-mode: use\_request\_fields

{

 "scenario": "search.test.online.random",

 “query”: "Сериал Друзья",

 "user\_id": "675ceb78-3d03-41b4-af5e-907499dfb3c0",

 "limit": 10

}

*Ответ:*

{

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

 "cache": "use\_request\_fields",

 "timestamp": "2024-04-11T16:27:07.468613",

 "fallback": [],

 "total": 10,

 "payload": [

 {

 "content\_id": "a534f645-ce7b-4d34-b3a3-e7888ca88926",

 "score": 0.494,

 "type": "unknown"

 }

 ]

}