# **API документация**

Базовый URL: "http://127.0.0.1:5000"

## **Общие параметры:**

Category – строка ‘film’ / ‘music’ / ‘book’ / ‘game’, обозначающая категорию элемента

isMain – запрос по главной категории / подкатегории (для Кино: фильмы – главная, режиссеры и актеры – подкатегория)

userId – ID пользователя (в запросах с одним ID – любого, в запросах с двумя ID userID – для авторизованного пользователя, с аккаунта которого посылается запрос)

personId – ID другого пользователя

name/title – строка, которая должна содержаться в названии/имени искомого элемента/пользователя

Во всех запросах передается хэдер “Authorization Bearer {access\_token}”, если не указано обратного.

## **Analize: запросы на сравнение избранных двух пользователей.**

* 1. GET запрос элементов в категории, общих для обоих пользователей

request

url: '/common/',  
params: {  
 userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
 personId: string

}

response: SharedItemResponseJson

interface SharedItemJson {  
 id: string,  
 title: string  
}  
  
interface SharedItemsResponseJson {  
 Items: SharedItemJson[]  
}

1.2 GET запрос элементов в категории, которые есть у person, но отсутствуют у user (новые для user)

url: '/new/',  
params: {  
 userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
 personId: string

}

response: SharedItemResponseJson

interface SharedItemJson {  
 id: string,  
 title: string  
}  
  
interface SharedItemsResponseJson {  
 Items: SharedItemJson[]  
}

1.3. GET запрос 5 элементов в категории c наибольшим (isHigh = true) / с наименьшим (false) рэйтингом у пользователя

'/top/',  
params: {  
 userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
 isHigh: 'true' | 'false'  
}

response: TopItemsResponseJson

interface TopItemJson {  
 id: string,  
 title: string,  
 rating: number,  
 review?: string  
}  
  
interface TopItemsResponseJson {  
 Items: TopItemJson[]  
}

1.4. GET запрос элементов, общих для двух пользователей, но с различным рэйтингом (разница оценки элемента у user и у person > 1 звезды)

url: '/different/',  
params: {  
 userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
 personId: string

}

interface DifferentRatingJson {  
 id: string,  
 title: string,  
 rating\_user: number,  
 rating\_person: number  
}  
interface DifferentRatingResponseJson {  
 Items: DifferentRatingJson[]  
}

## **Filter : запросы на получение опций и на информацию о пользователях, подходящих под фильтр**

* 1. GET запрос пользователей, подходящих под параметры фильтра. Ожидается возврат информации о пользователе и элементы, которые подошли под фильтр. Массив id элементов в категории передается через запятую: film\_main=id1,id2,id3….idn

film\_main, film\_sub – массивы id добавочных элементов для поиска

is\_film\_main, is\_film\_sub – строка ‘true’ или параметр отсутствует, если false. Этот параметр означает, нужно ли искать совпадения с избранными элементами пользователя, посылающего запрос.

url: '/userfilter/',  
params: {  
 userid: string,  
 film\_main?: string,  
 is\_film\_main?: string,  
 film\_sub?: string,  
 is\_film\_sub?: string,  
 book\_main?: string,  
 is\_book\_main?: string,  
 book\_sub?: string,  
 is\_book\_sub?: string,  
 music\_main?: string,  
 is\_music\_main?: string,  
 music\_sub?: string,  
 is\_music\_sub?: string,  
 game\_main?: string,  
 is\_game\_main?: string,  
 game\_sub?: string,  
 is\_game\_sub?: string,  
 name?: string   
}

response: UserCardsByFilterJson

interface UserInfoJson {  
 telegram: string,  
 id: string,  
 gender: string,  
 username: string,  
 age: string,  
 avatar: string  
}  
  
interface UserCardJson {  
 User: UserInfoJson,  
 Film: string[]  
 Director: string[]  
 Book: string[]  
 Author: string[],  
 Game: string[],  
 Studio: string[],  
 Song: string[],  
 Artist: string[]  
}  
interface UserCardsByFilterJson {  
 FoundUsers: UserCardJson[]  
}

* 1. GET запросы всех опций из базы для фильтра

url: '/options\_book\_main/'

url: '/options\_book\_sub/'

url: '/options\_game\_main/'

url: '/options\_game\_sub/'

url: '/options\_music\_main/'

url: '/options\_music\_sub/'

url: '/options\_film\_main/'

url: '/options\_fulm\_sub/'

response: OptionItemModel[]

interface OptionItemModel {  
 id: string,  
 title: string  
}

## **Moderator : запрос на получение комментариев для данного модератора и на одобрение/неодобрение комментария**

* 1. GET запрос на получение комментариев, назначенных модератору

url: `/moderator/`,  
params: {

moderatorId: string

}

response: CommentModel[]

interface CommentModel{  
 itemId: string,   
 comment: string

userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
}

* 1. POST запрос на одобрение комментария

url: `/approveComment/`,

body: {

comment: CommentModel,  
 moderatorId: string

}

interface CommentModel{  
 itemId: string,   
 comment: string

userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
}

response: void

* 1. Post запрос на отклонение комментария

url: `/approveComment/`,

body: {

comment: CommentModel,  
 moderatorId: string

}

interface CommentModel{  
 itemId: string,   
 comment: string

userId: string,  
 category: string,  
 isMain: 'true' | 'false',  
}

response: void

## **Userdata: запросы, связанные с получением информации о пользователе, взаимодействию с его элементами и связанными с ним людьми**

* 1. GET запрос на получение типа связи между user и person

url: `/fetchTypeOfPersonForUser/`

params: {

userId: string,

personId: string

}

response: TypeOfConnetctionResponseJson

interface TypeOfConnetctionResponseJson {  
 type: string  
}

Ответы: "user\_is\_follower" | "friends" | "requests" | “visited”

* 1. GET запрос на получение личных данных пользователя

url: `/userdatausers/`,

params: {

userId: string

}

response: UserInfoRequestJson

interface UserInfoRequestJson {  
 User: UserInfoJson  
}

interface UserInfoJson {  
 telegram: string,  
 id: string,  
 gender: string,  
 username: string,  
 age: string,  
 avatar: string  
}

* 1. POST запрос на изменение личной информации пользователя

url: `/api/editUserPersonalInfo/`,  
body: {  
 id: string,  
 nickname: string,  
 age: string,  
 gender: string,  
 telegram: string  
}

response: void

* 1. POST запрос на удаление аккаунта

url: `/api/deleteUserAccount/`,  
body: {  
 id: string  
}

response: void

* 1. GET запрос на получение всех людей, связанных с пользователем заданным образом

url: '/users\_peers/',  
params: {  
 personType: friends" | “my\_requests” | “requests” | “visited”,  
 userId: string,  
 title: string  
}

response: UserPersonsResponseJson

interface UserPersonsResponseJson {  
 FoundUsers: UserInfoRequestJson[]  
}

interface UserInfoRequestJson {  
 User: UserInfoJson  
}

interface UserInfoJson {  
 telegram: string,  
 id: string,  
 gender: string,  
 username: string,  
 age: string,  
 avatar: string  
}

* 1. GET запрос на избранные элементы пользователя

url: '/userdataitems/'

params: {  
 userId: string,  
 category: string,  
 isMain: 'true' : 'false',  
 title: string  
}

response: UserItemsRequestJson

interface ItemJson {  
 id: string,  
 title: string,  
 review?: string,  
 rating: number  
}  
  
interface UserItemsRequestJson {  
 Items: ItemJson[]  
}

* 1. POST запрос на изменение рэйтинга элемента

url: `/api/updateItemRating/`,  
body: {  
 userId: string,  
 category: string,  
 isMain: 'true' : 'false',  
 itemId: string,  
 value: string  
}

response: void

* 1. POST запрос на изменение комментария к элементу

url: `/api/updateItemComment/`,  
body: {  
 userId: string,  
 category: string,  
 isMain: 'true' : 'false',  
 itemId: string,

value: string  
}

respone: void

* 1. POST запрос на добавление элемента в избранные

url: `/api/addItem/`,  
body: {  
 userId: string,  
 category: string,  
 isMain: 'true' : 'false',  
 itemId: string  
}

response: void

* 1. POST запрос на удаление элемента из избранного

url: `/api/removeItem/`,  
body: {  
 userId: string,  
 category: string,  
 isMain: 'true' : 'false',  
 itemId: string  
}

response: void

* 1. POST запрос на принятие заявки в друзья (пользователь становится другом)

url: `/api/acceptRequestToFriends/`  
body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. POST запрос на отклонение заявки в друзья (пользователь становится подписчиком)

url: `/api/denyRequestToFriends/`  
body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. POST запрос на отправление заявки в друзья (пользователь подписывается на человека, которому отправлена заявка)

url: /api/requestPersonToBeFriends/`  
body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. POST запрос на пометку пользователя как посещенного

url: `/api/visited/`  
body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. POST запрос на отмену своей подписки (отправки заявки в друзья) на другого человека

url: `/api/closeMyRequest/`

body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. POST запрос на удаление человека из друзей

url: /api/removePersonFromFriends/`

body: {  
 personId: string,  
 userId: string  
}

response: void

* 1. GET запрос на получение количества заявок в друзья

url: `/countFollowers/`,  
params: {

userId: string

}

response: {

count\_followers: number

}

## **Login: авторизация, обновление токенов**

* 1. GET запрос на авторизацию в приложении (без хедера авторизации)

token – accessToken, полученный от Яндекса на front-части

url: '/login',  
params: {  
 token: string  
}

response: {

user\_id: string,

access\_token: string,

refresh\_token: string

}

* 1. GET запрос на реавторизацию при перезагрузке страницы, когда access\_token имеется в localStorage

Access и refresh токены участвуют в этом запросе через интерсептор, как во

url: '/get\_id'

response: {

user\_id: string

}

* 1. GET запрос на refresh токенов (в хэдере передается refresh токен)

url: `/refresh`

response: {

user\_id: string,

access\_token: string,

refresh\_token: string

}