

快手网红分析师项目

四堡一

1. 项目概述

1.1 项目背景

短视频行业的飞速发展，诞生了许多短视频领域的创业者。众所周知，想要做好一个短视频账号，内容质量方面必须要高，除此之外，日常的数据分析也是非常的重要。通过专业的视频分析工具，不仅能了解到行业内的最新玩法，还能学到竞品上热门的套路。下面我们就准备为快手的短视频作者做一款分析工具 -- 快手网红分析师。

1.2 项目目标

快手上每天有数千万作者上传自己的视频和图片作品，这些作品会被在快手平台全网分发，快手用户能看到这些作品并对它们进行浏览、评论和点赞。为了帮助视频作者更好的了解自己拍摄作品的受欢迎程度，我们需要一个提供一个面向作者的辅助分析工具，通过使用这个工具用户可以比较清晰的了解自己作品每天被浏览的情况。

1.3 功能需求

模块	功能描述
登录注册授权	设置本站登陆注册功能，且登陆之后用户能够与快手账号进行绑定
首页信息展示	展示用户基本信息；展示用户近三天活跃度增长情况
用户通知中心	获取用户收到的所有通知、并进行通知状态管理
作品展示管理	展示所有作品信息、对视频实现标签管理
作品数据分析	展示作品近三天内各项指标变化及环比变化情况、作品各项指标变化情况可视化

1.4 用户故事

我是一位快手视频创作者，刚刚创建了自己的账号，发了几条视频，粉丝不多，但是我非常想知道自己的每条账号的活跃度以及粉丝观看的情况，从而对我之后的策略做出调整。

我希望有一个数据分析平台，他的账号要能够和快手独立，方便我和其他的团队成员进行管理，并且要能快捷的绑定我的快手账号。

这个平台应该可以让我看到我的快手账号整体的统计信息，比如评论、播放、点赞数；可以看到过去一段时间的视频指标的变化；还应该让我看到我发的每一条视频的相应数据和图表展示，最好能让我在看视频数据时预览我发的视频。

同时，我希望我能对自己发布的视频做出标注，比如标上一些视频种类的tag，也希望机器能够自动帮我做出这一步，让我更好的分析视频。

由于我会时刻关注视频账号数据上视频的播放数据，所以我希望这个平台能够在我的视频数据指标出现比较大增长或者下降时，给我发送通知；同时我还希望这个视频平台能够每天给我发送日报来展示当天的数据变化情况。

我希望这个平台能够在PC端使用、并且拥有美观性和易用性。我还希望这个平台能够保证数据安全，保证我的快手账号和快手数据信息不会泄露。

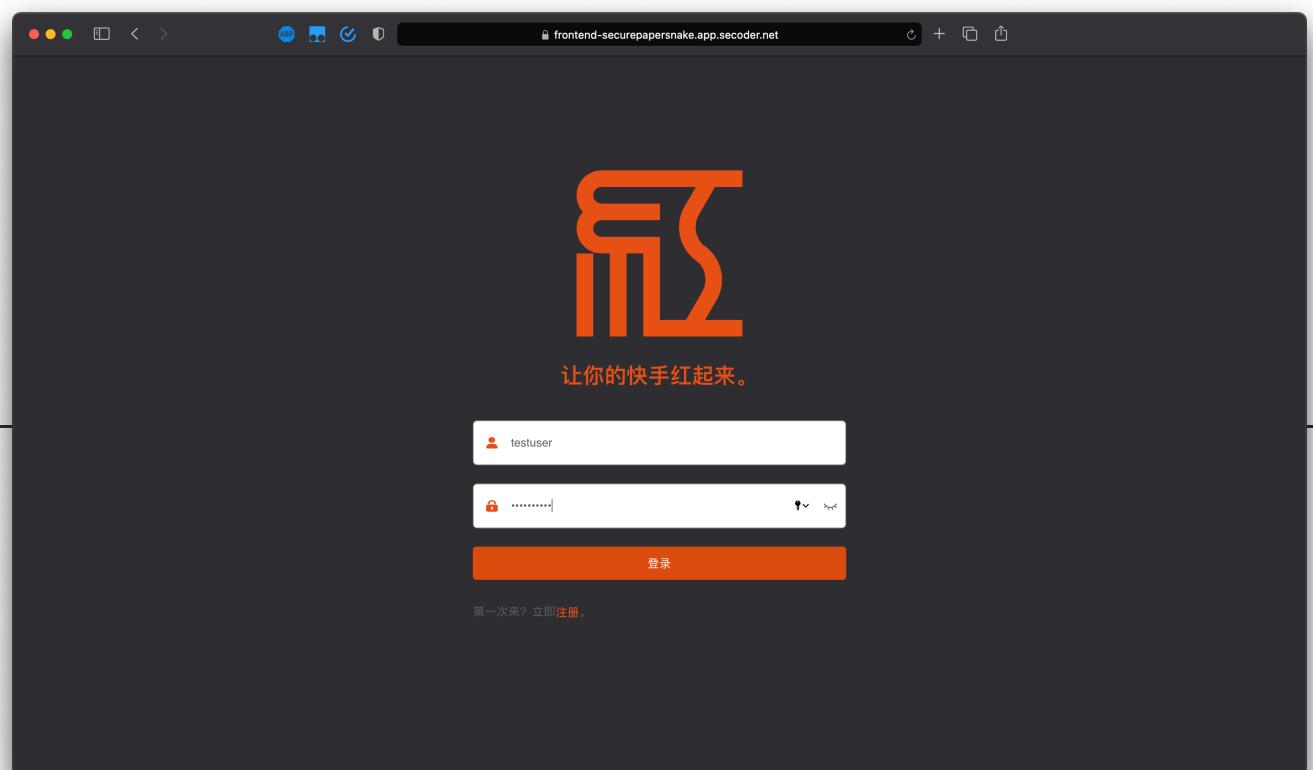
我希望这个平台能让我创造出用户更加喜爱的视频，并实现视频数、评论观看点赞数的稳步增长，让我成功成为一名具有影响力的网红。

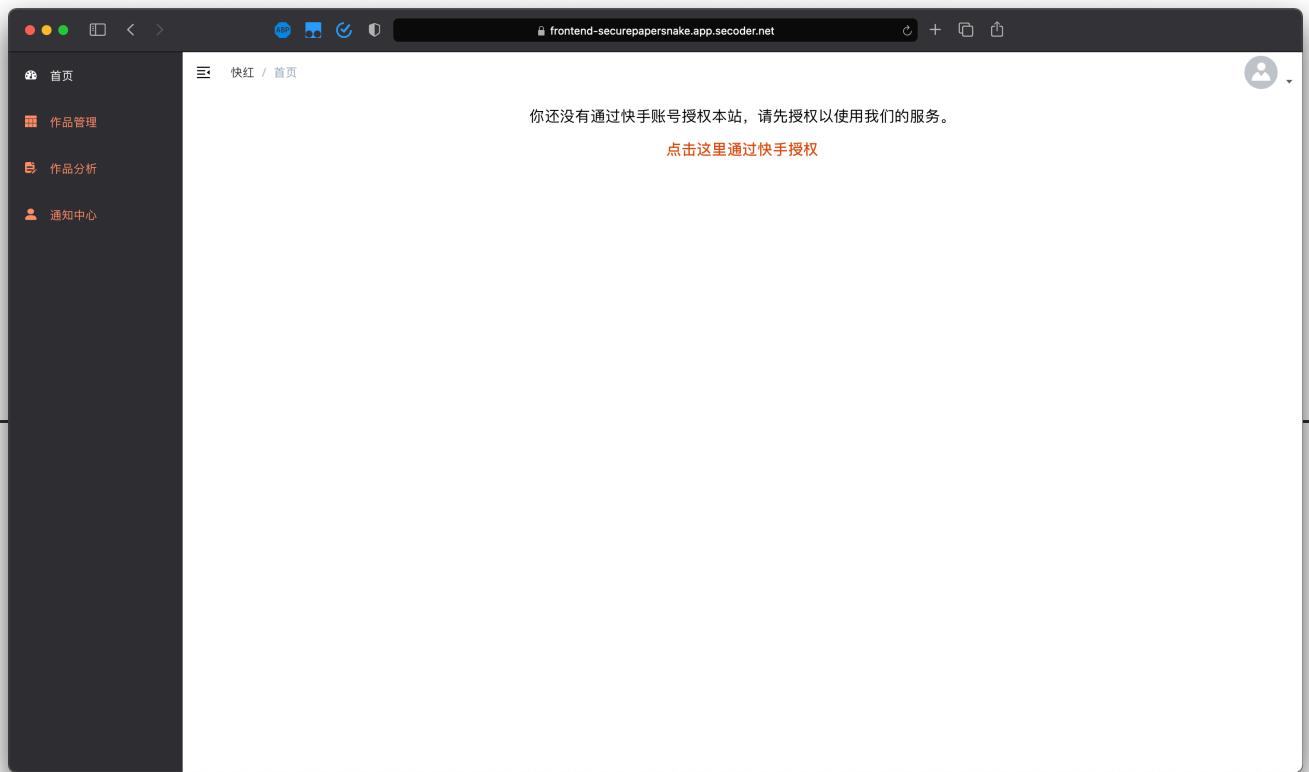
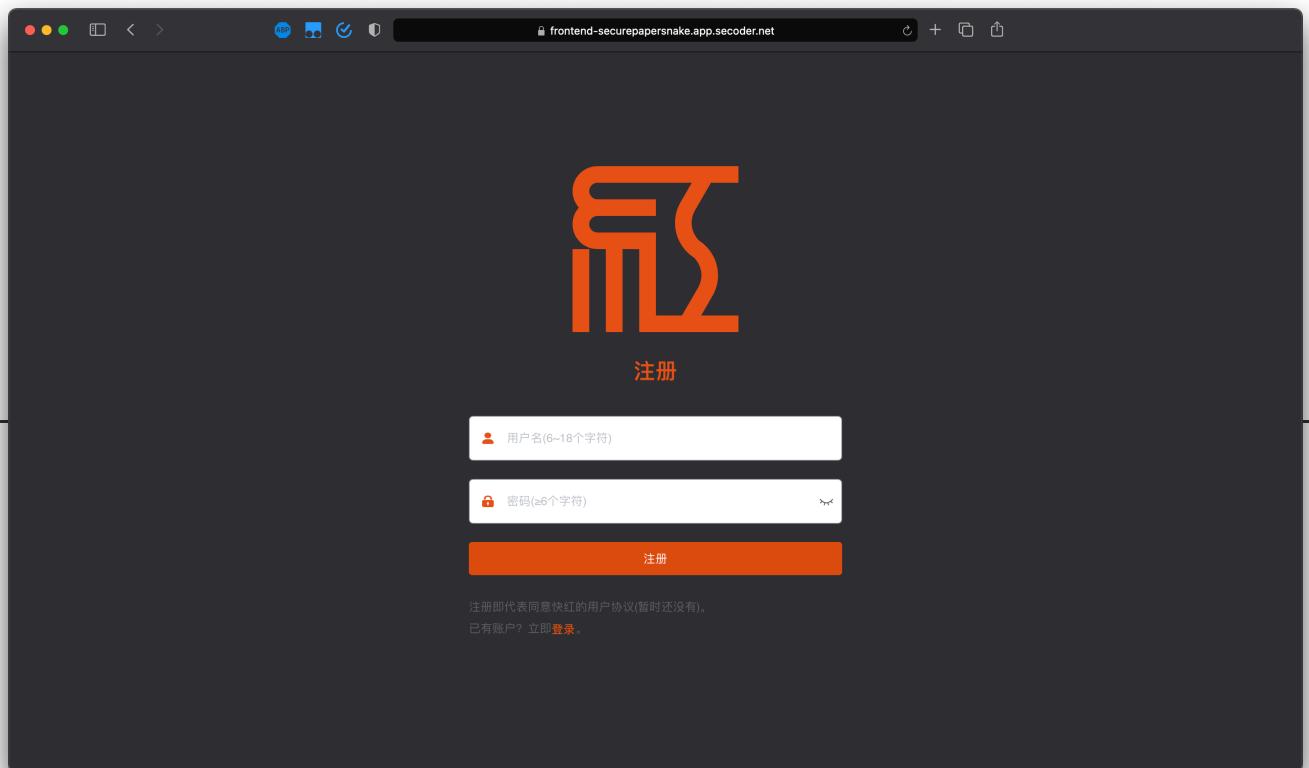
2. 登录注册授权

简要说明

1. 用户首次使用本站时需要进行用户注册，注册信息为用户名与密码；注册之后进入本站首页，需要通过首页提供的链接进行快手账号授权。
2. 用户再次使用本站可以通过之前已有的账号登录，不需要再次授权账号。

界面原型





逻辑规则

1. 注册

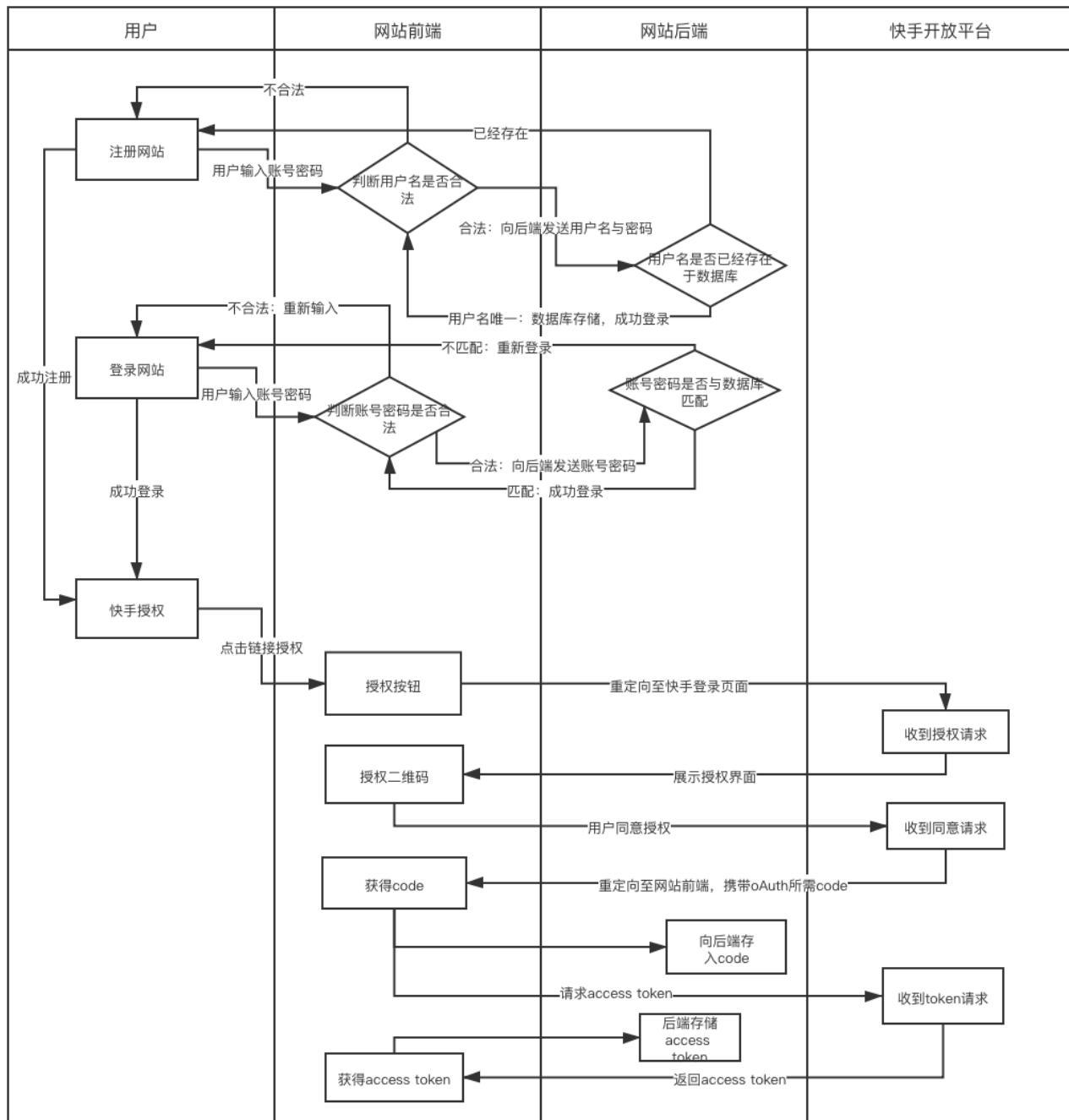
字段	必填	格式要求	校验点	错误提示语
用户名	是	字符串，长度大于5小于19	长度要求、不能与已有用户名重复	'Username: length between 6 and 18' 'This user name had already been used!'
密码	是	字符串，长度大于5	长度要求	'Password: length longer than 6'

2. 登录

字段	必填	格式要求	校验点	错误提示语
用户名	是	字符串，长度大于5小于19	长度要求、用户名与密码输入正确	'Please enter the correct username(length between 6-18)' 'Account and password are incorrect.'
密码	是	字符串，长度大于5	长度要求	'Please enter the correct password(longer than 6)'

3. 授权

利用快手提供的开放接口获取用户的个人信息和视频数据。具体流程为，首先向快手提供的api接口请求权限，在请求中包含回调url，用户在授权界面完成授权后，将code回调给后端提供的url，后端获取code后使用code取得用户的access_token，使用access_token向快手请求用户数据。



3. 首页信息展示

简要说明

- 授权登录后需要在首页展示用户的基本信息，如头像、昵称、粉丝数量、共获点赞/评论/播放数
- 展示24h内点赞、评论、播放增量
- 将24h内个小时的活跃度增量可视化展示
- 可视化展示标签数量

界面原型



逻辑规则

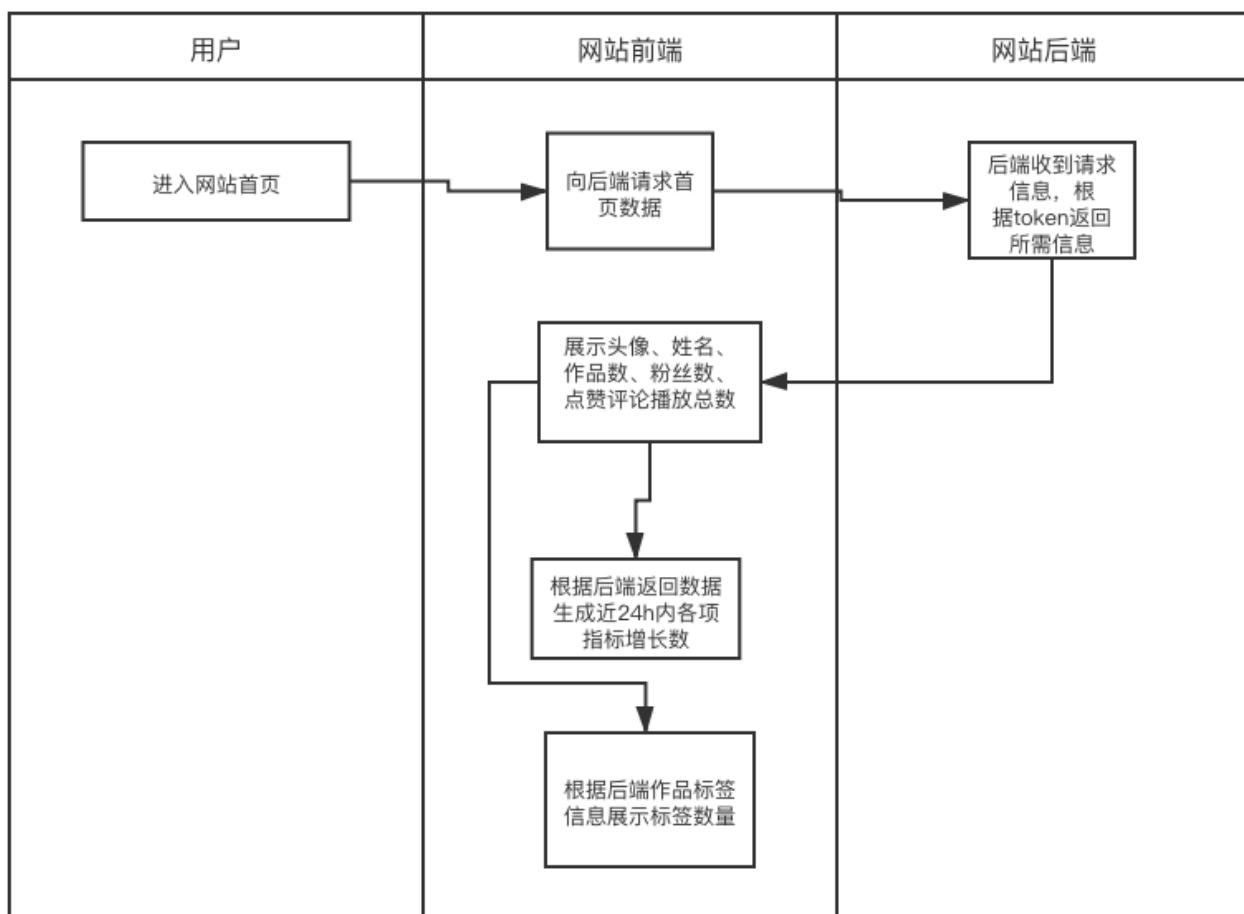
1. 进入首页之后，调用后端接口返回所有数据。

数据包括：用户基本信息；三个数组，分别代表点赞、评论、播放，每个数组包括近24小时内每个小时对应指标的总量。前端在获取这些数据之后，计算出个小时内对应指标的增量，传入图表组件进行可视化展示。

2. 图表展示规则

横坐标表示从当前时间往前数的24个小时；纵坐标有四个，分别表示点赞、评论、播放以及三者总和（a.k.a 总活跃度）。图表要有标题说明图表内容。另外应该提供数据展示选项：通过对应的开关选择在图表中是否展示某一数据。

3. 上方表格中的数据如果过大可能会产生溢出，应该进行缩写处理。

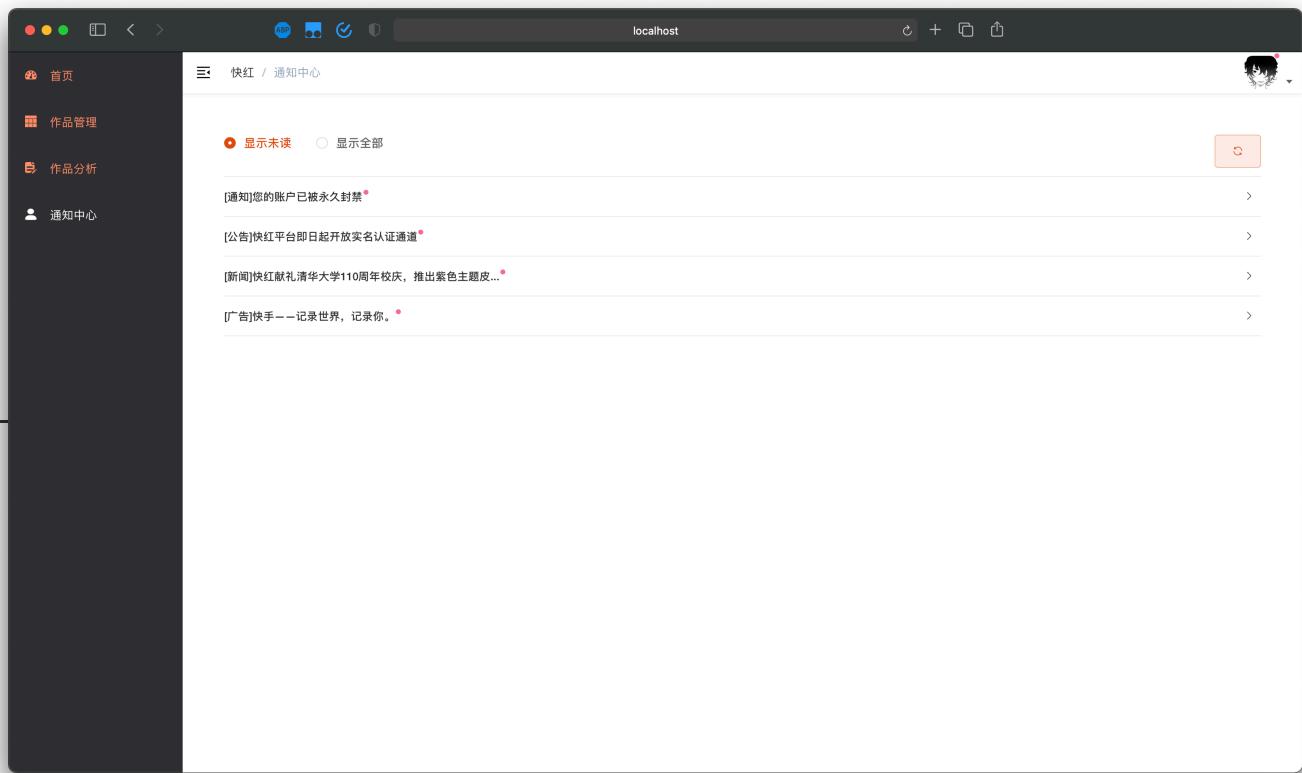


4. 用户通知中心

简要说明

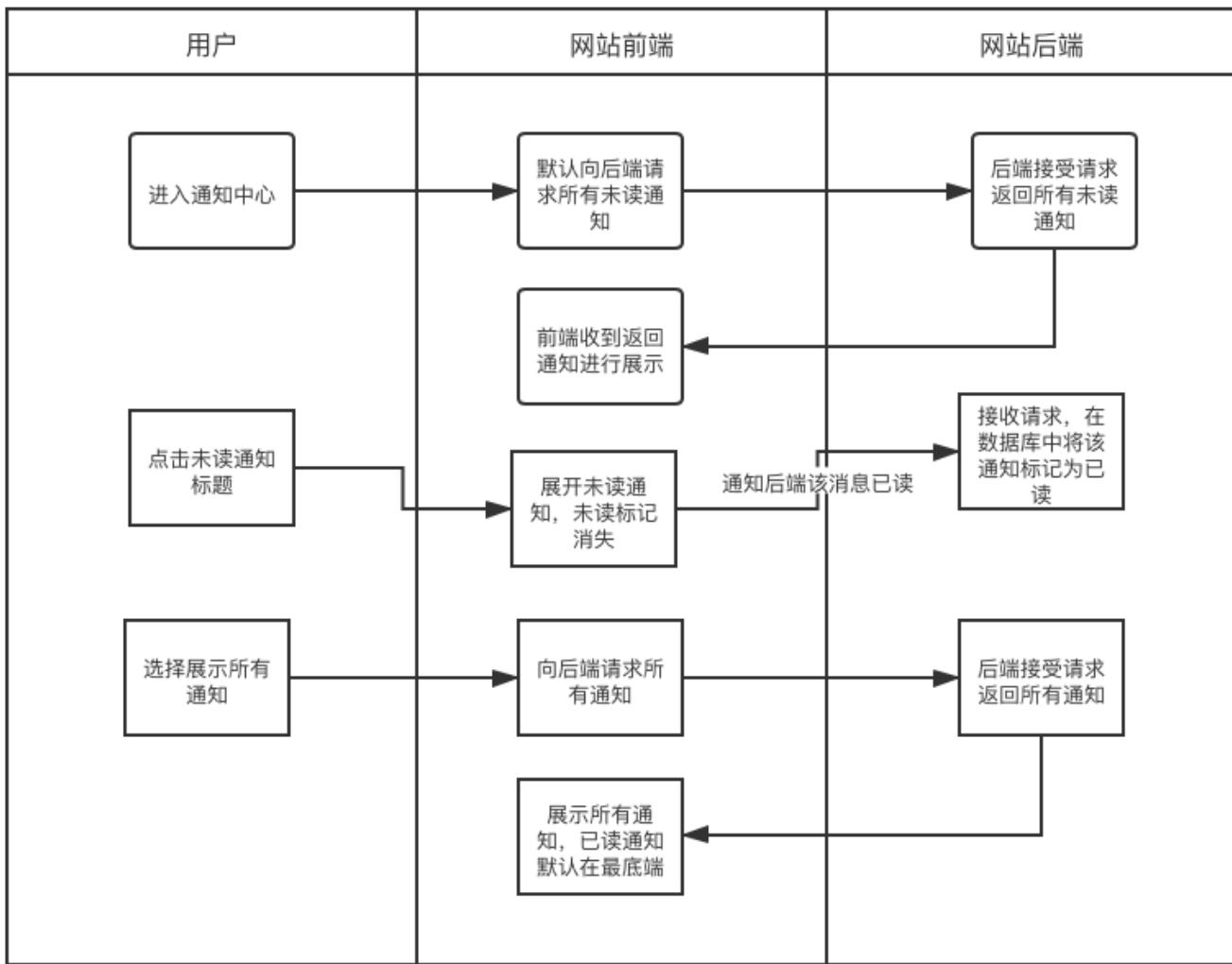
1. 向用户展示所有通知，用户可对通知进行读取操作
2. 用户可以编辑通知读取状态，并对通知范围进行选取

界面原型



逻辑规则

1. 通知中心页面默认展示未读通知，以标题为项目陈列，点击标题可展开阅读内容。排列顺序为：未读优先于已读、较新的通知靠前
2. 未读通知标题旁会有醒目标记，一旦展开标题，标记消失，阅读完毕后该通知加入已读状态
3. 页面顶部可以选择是否展示所有通知，用户若选择是，已读的通知也会显示在最下方
4. 导航栏中，点击用户头像出现的下拉菜单中也有通知中心的入口，未读通知数量会标记在该条目旁
5. 通知中心的展开条目的方式为风琴式，一次只能阅览一条通知的内容



5. 作品展示管理

简要说明

- 通过对时间范围的选取，展示一段时间内发布的视频信息，包括发布时间、获得点赞、评论、播放数据等。
- 实现添加标签功能：包括通过视频标题自动生成标签，手动添加标签
- 单击视频标题进入单个视频的数据分析界面
- 提供分页功能对单页视频数量进行选择

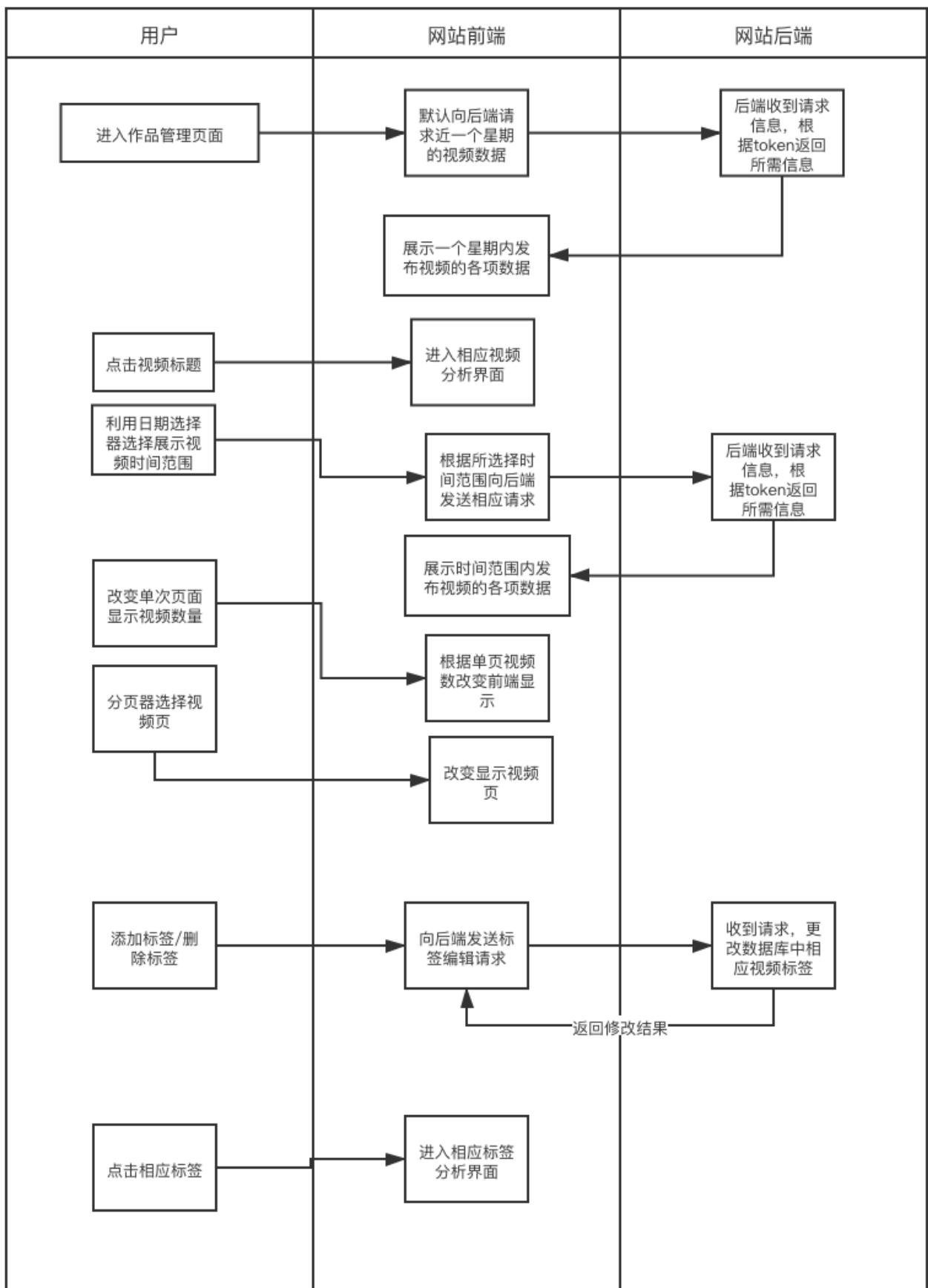
界面原型

Title	Display_time	Likes	Comments	Plays	Tags
《画皮》中国 马丽 邓家佳 非行 钱永强 魏巍 奇幻/103分钟 主演: 马丽、邓家佳、非行、钱永强、魏巍 导演: 陈凯歌 #经典大片免费看	2021-03-17 06:16:50	390512	18359	43164311	孙俪 × 陈嘉上 × 钱永强 × Add Tag
《夏洛特烦恼》 刘烨 夏洛特 狄龙 刘亦菲 黄圣依 梁家辉 灰太狼 凯文 张国荣 叶问 李心 喜羊羊 乔杉 何润东 吴宇森 张翰 林允	2018-05-10 05:54:10	91562	2924	23917412	岩田 × 阿基拉 × 佐佐木 × Add Tag
《灰太狼之牛气冲天》中国大陆/2009/喜剧/动画/90分钟 主演: 喜羊羊、灰太狼、沸羊羊、美羊羊、懒洋洋、村长、小灰灰 导演: 简耀宗 #经典大片免费看	2009-04-25 04:52:08	173401	14553	19532185	喜羊羊 × 灰灰 × 灰太狼 × Add Tag
《喜羊羊与灰太狼之虎虎生威》中国/2010/动画/儿童/89分钟 主演: 喜羊羊、灰太狼、沸羊羊、美羊羊、懒洋洋、村长、小灰灰 导演: 赵崇邦 #经典大片免费看	2010-03-17 04:40:54	249840	18484	23620212	喜羊羊 × 兔年 × 顶呱呱 × Add Tag
《喜羊羊与灰太狼之开心闯龙年》中国大陆/2012/喜剧/动画/儿童/95分钟 主演: 喜羊羊、灰太狼、沸羊羊、美羊羊、懒洋洋 导演: 简耀宗 #经典大片免费看	2012-03-16 04:46:02	340164	15544	23437143	喜羊羊 × 龙年 × 灰太狼 × Add Tag
《蜜月酒店杀人事件》中国大陆/2016/剧情/悬疑/97分钟 主演: 张静初、何润东、金永敏、倪虹洁、吴维彬 导演: 张哲洙 #经典大片免费看	2016-03-12 03:36:52	455074	17046	42865416	张静初 × 何润东 × 金永敏 × Add Tag

逻辑规则

- 进入作品管理页面时通过后端接口默认返回最近一周的视频数据。通过上方的日期选择其他时间范围。
- 底部用一个分页器选择单页最多展示视频数量，提供点击跳转和输入跳转方式。
- 表格第一栏展示作品标题，提供点击跳转至对应作品分析界面，鼠标悬浮时应该显示视频封面。中间四栏分别展示发布时间、获得点赞、评论、播放数。

最右侧一栏是对应视频的标签管理页，可以添加或删除视频标签。默认标签在后端自动生成；可以选择手动给视频添加标签，点击添加标签时会产生云图显示较多的标签作为参考。标签应当为字符串类型，且添加标签字数不得超过20个字符。



6. 作品数据分析

简要说明

- 提供一个作品数据分析页面，需要显示作品近三天内各项指标（点赞、评论、播放）变化数及环比变化情况
- 通过时间范围选择器与图表展示选定时间范围内各项指标变化情况
- 如果没有选定单个作品则分析所有视频

界面原型



This screenshot shows a movie analysis page for the film 'The Painted Skin'. The main card displays the movie's title, '中国大陆 中国香港 新加坡/2008/剧情 爱情 ...', its rating of 39.0万, 1.8万 comments, and 4316.4万 views. Below this is a timeline selector set to '24h' and a bar chart showing engagement metrics for three days: May 5th, May 6th, and May 7th. The chart includes three series: likes (blue), comments (green), and views (yellow). The views series shows a significant peak on May 7th.

日期	点赞	评论	播放
5/5	~500	~300	~100,000
5/6	~550	~350	~150,000
5/7	~580	~300	~164,900

This screenshot is similar to the one above, but it includes a modal window displaying the movie's credits. The modal lists the production companies involved: Ningbo Film Studio, Shanghai Film Group Corporation, Shenzhen Starlight Film Co., Ltd., and Hong Kong Starlight Media Management Co., Ltd. The main card and chart are visible in the background.

规则逻辑

- 左侧导航栏默认进入全局作品分析页，作品管理页点击标题根据视频id跳转到单个作品分析页
- 全局作品分析页展示所有作品最近三日的播放、点赞、评论的变化量与环比变量；单作品分析页展示单个作品最近三日的播放、点赞、评论的变化量与环比变量。

单作品分析页左侧标题如果过长需要进行切割处理，通过鼠标悬浮完整展示。

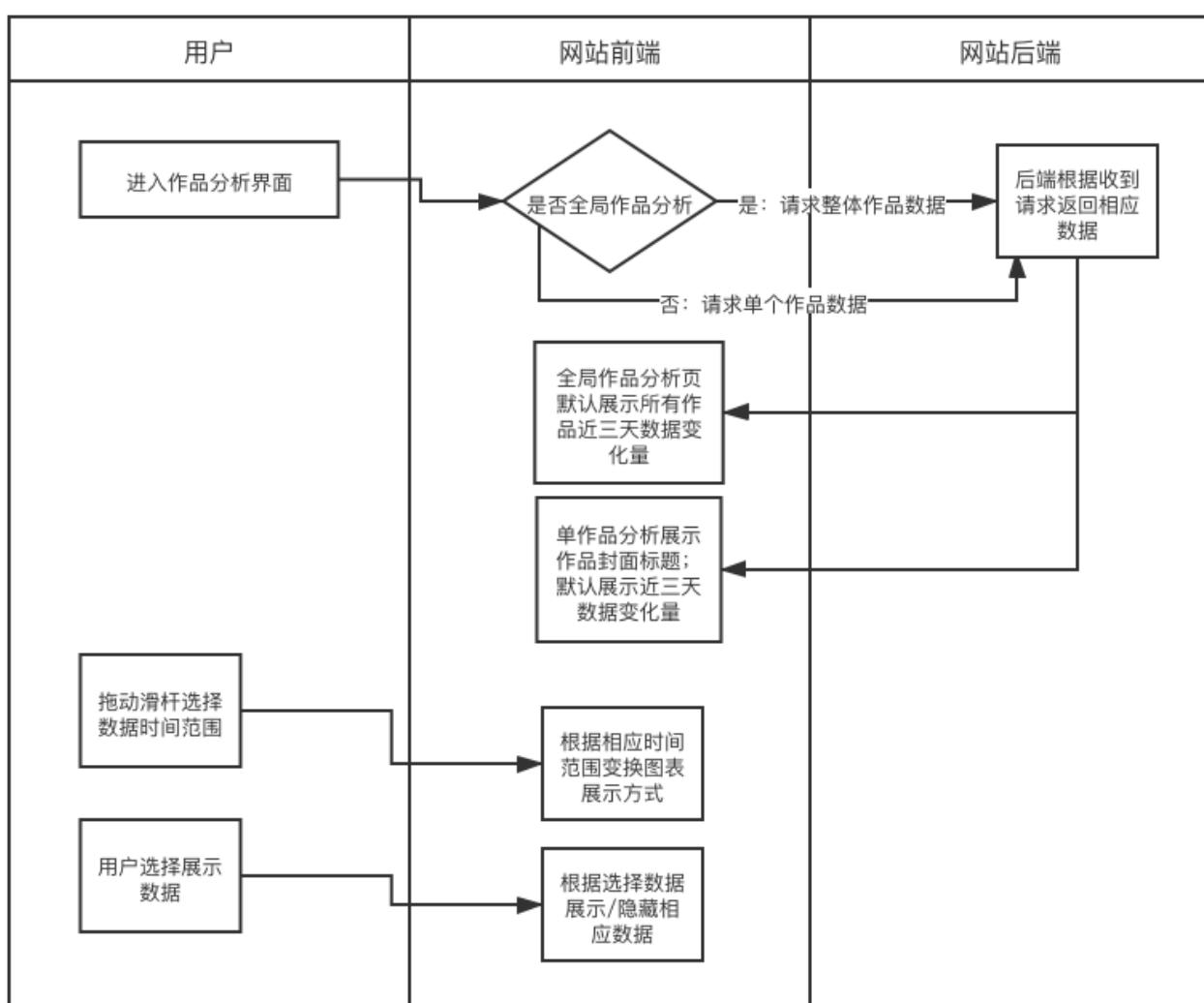
3. 底部图表默认展示最近三天的数据变化量，通过滑动上方时间范围选择器改变图表展示方式。

当滑杆两点同时处于24h时，图表展示最近24小时数据变化量，标题显示为过去24小时的作品数据。

当滑杆两点同时处于其他点时，图表展示当天数据变化量，标题显示为过去第x天的数据。

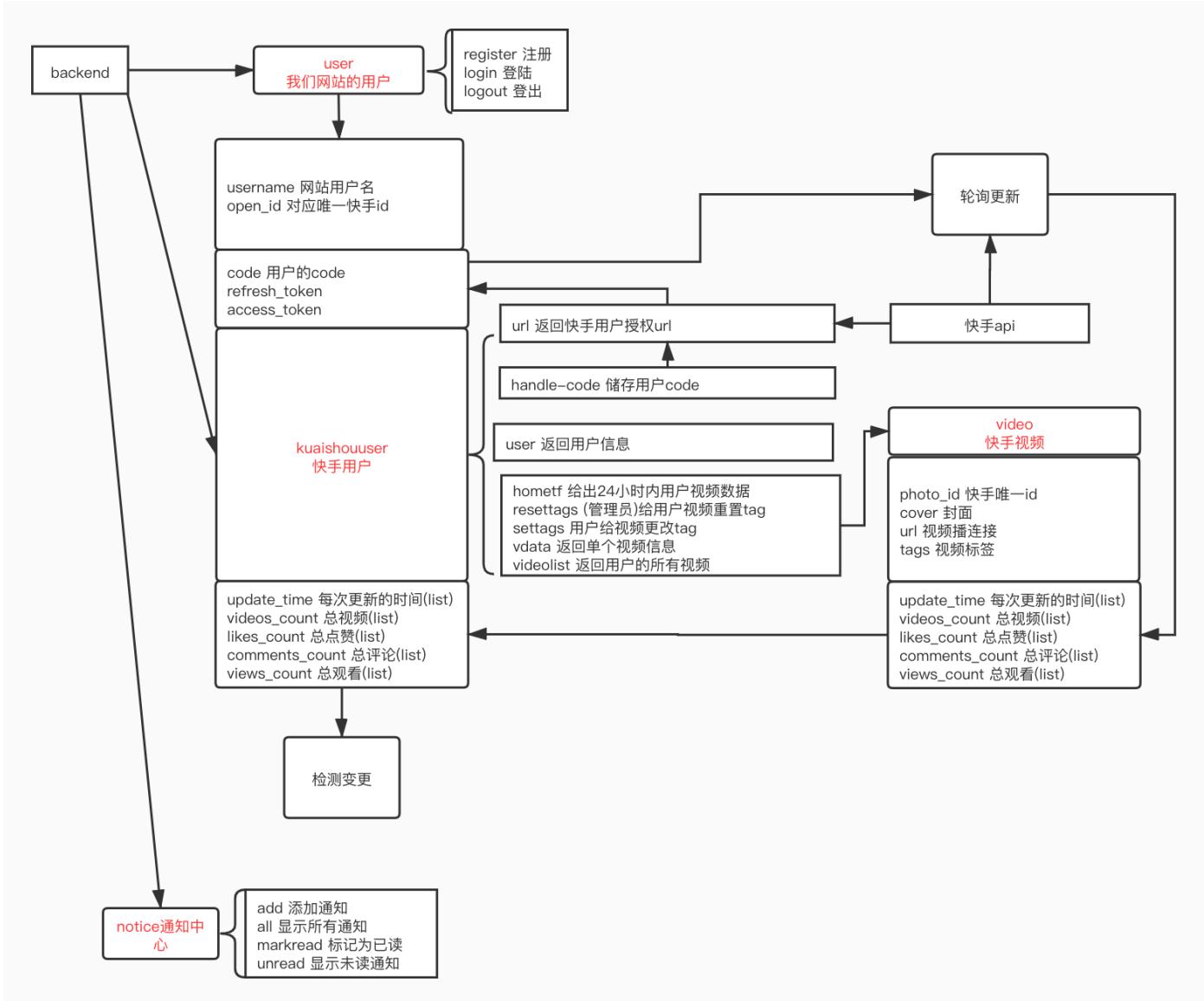
当滑杆处于其他状态时，图表展示的是该段时间内每天的数据变化量，标题显示为过去第x天到第y天的作品数据。

[



7. 后端逻辑

数据库中储存了四个类:User, Kuaishouuser, Video, Notice，实现了后端与快手api的分离，其交互示意
图如下：



User类： 本网站的用户类，为了让用户登陆网站而设计。

Kuaishouuser类： 快手用户类，记录了快手的唯一id与快手用户进行一一对应，以 User 类的用户名作为主键与 User 类形成对应。

Video类： 快手视频类，以快手的唯一视频id与快手视频进行一一对应，以 Kuaishouuser 类作为外键与 Kuaishouuser 类形成对应。

Notice类： 网站通知类，以 User 类作为外键与 User 类形成对应。

后端与快手api分离： 后端仅会在用户授权和轮询时调用快手api，在前端进行请求时只会从数据库中拉取数据，从而实现后端与快手api分离。

轮询： 后端部署完成后轮询就会开始，按照一定周期访问 快手api 并对数据库中的 Video 类和 Video 类对应的 Kuaishouuser 类进行更新，如果有用户活跃度激增则调用 Notice 类进行通知。

上述实现保证了在用户请求用户信息时后端直接从数据库中获取信息，而不是调用快手api获取信息。

8. 前后端交互逻辑及API文档

后端采用规范的 RESTFUL API 格式书写，并将通过 SWAGGER 自动生成的文档交付前端，方便前端开发人员在线预览调试，减少前后端沟通成本。

API文档详情可见在线版本：[API Document](#)，此处由于软件工程平台自身的安全性远低于我们的标准设计，以及方便第三方检查，只能部署在公网之上。

The screenshot shows the Swagger UI interface for a Django application. At the top, there's a navigation bar with the Swagger logo, the URL <https://backend-securepapersnake.app.secoder.net/swagger/?format=openapi>, and an [Explore](#) button. Below the header, the main content area is divided into sections based on API tags:

- kuaishou**:
 - GET /kuaishou/handle-code/ (kuaishou_handle_code_list)
 - POST /kuaishou/hometf (kuaishou_hometf_create)
 - GET /kuaishou/resettags/ (kuaishou_resettags_list)
 - POST /kuaishou/settags (kuaishou_settags_create)
 - GET /kuaishou/url/ (kuaishou_url_list)
 - GET /kuaishou/user/ (kuaishou_user_list)
 - POST /kuaishou/vdata (kuaishou_vdata_create)
 - POST /kuaishou/videolist (kuaishou_videolist_create)
- notice**:
 - POST /notice/add (notice_add_create)
 - POST /notice/all (notice_all_create)
 - POST /notice/markread (notice_markread_create)
 - POST /notice/unread (notice_unread_create)
- test**:
 - GET /test/ping/ (test_ping_list)
 - POST /test/ping/ (test_ping_create)
- user**:
 - POST /user/login (user_login_create)

At the bottom of the interface, a note states: "前端在post中包含用户名和密码，后端验证通过后登录用户并将session_id在cookie中返回" (Frontend includes username and password in the post, backend logs in the user and returns the session_id in the cookie).

The screenshot shows a detailed API endpoint configuration. In the 'Parameters' section, there are two fields: 'username' (string, query) with a placeholder 'username - username of user' and 'password' (string, query) with a placeholder 'password - password of user'. In the 'Responses' section, it lists two status codes: 200 (success) and 400 (username duplicate). Below this, two endpoints are defined: a GET endpoint for '/user/logout/' with a response type of 'user_logout_list' and a POST endpoint for '/user/register' with a response type of 'user_register_create'. A 'VALID' button with a refresh icon is located at the bottom right.

9. 解决用户信息获取过慢问题

问题： 在网站的1.1版本遇到了访问 `api/kuaishou/videolist` 等接口时加载速度过慢的问题，其中对于一个拥有2000+视频的用户网站的加载时间需要3秒左右。

1.1版本的后端逻辑如下：

在网站收到用户请求所有视频时调用快手的 `access_token` 接口保证用户的token没有过期，之后调用快手的 `videoklist` 接口，等待快手返回所有视频的信息，这样就能得到最新的用户信息。

这样做是为了保证能够在新注册的用户立即使用网站时可以查询得到正确的信息，如果不在这儿调用 `videoklist` 接口，那么新注册的用户只能在下一次轮询后才能看到自己的视频（如果是以1小时为周期做轮询，那么在1小时后才能看道信息）。

当用户的视频数较多时，快手api需要将所有视频的基本信息传回来，网站需要等待得到所有视频信息后才能加载，所以很慢。

其他接口由于每次在调用前需要调用快手的 `access_token` 来接口验证用户的正确性，所以当快手端返回数据较慢时仍然会出现网站加载速度慢的问题。

解决方案：

我们将轮询函数的工作周期改为了1分钟一次。每次只对上次更新时间与现在时间相差一小时的视频进行更新，同时对从未进行更新过的视频进行更新。

这样每个视频仍然是1小时更新一次，但是我们将之前的1小时一次的更新分散成了现在的分批次更新，一小时内服务器对快手api的访问次数与之前相同。并且我们删除了在网站收到用户请求所有视频时对快手的 `videoklist` 接口的调用，这是由于我们的轮询逻辑保证新注册的用户信息将会在1分钟内储存在数据库中。

此外，我们同样删除了调用快手的 `access_token` 接口保证用户的token没有过期的步骤，转而在用户等我们网站时统一验证用户的token没有过期，如果过期了就让他授权。

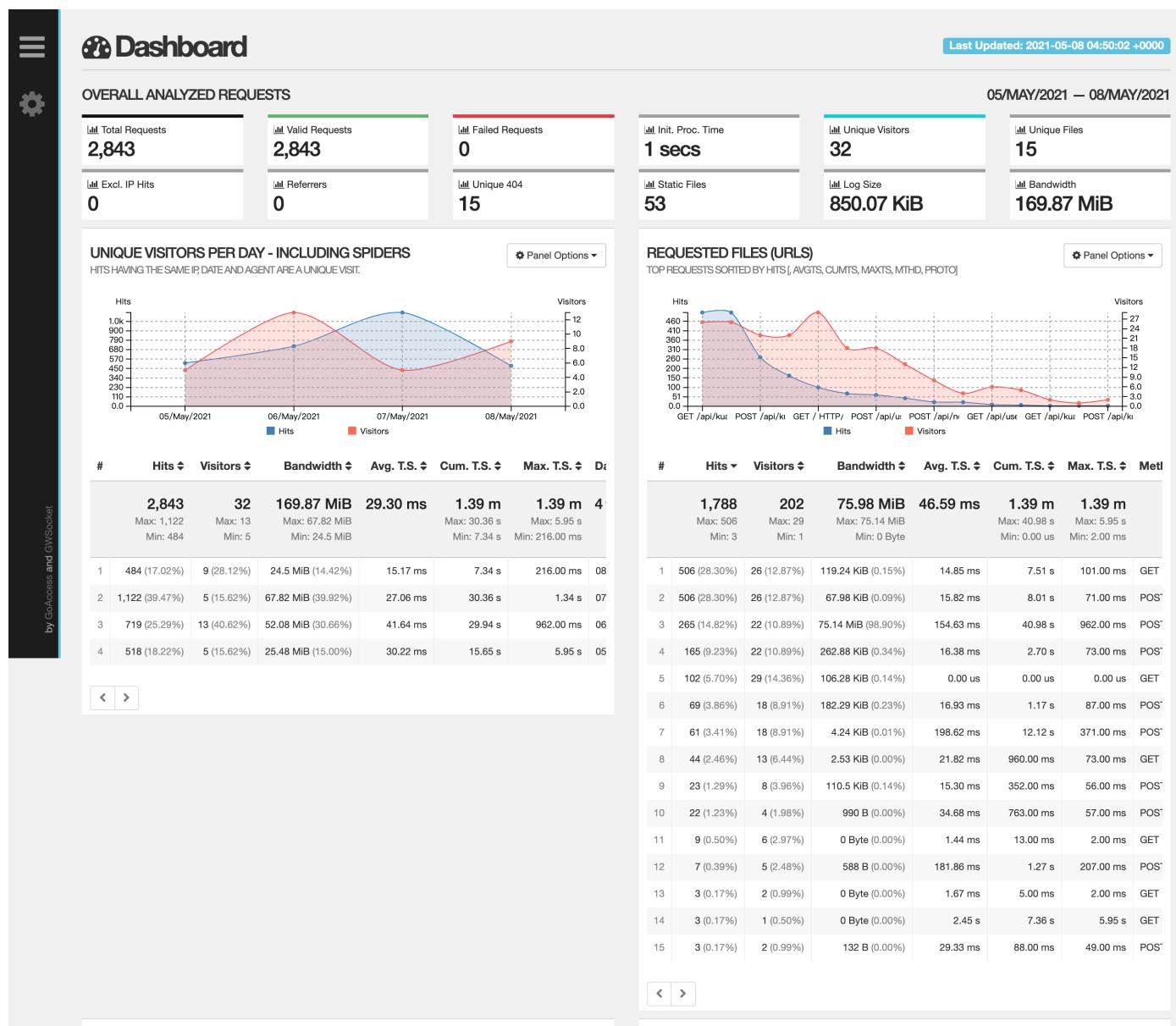
这样就保证了所有的用于返回用户信息的函数中没有调用快手api。

在实际的测试中，我们将平均3s的后端处理时间缩短到平均50ms，P99时间大约200ms。

10. 监控

我们采用 goaccess 对 nginx 进行打点统计，用于优化前后端响应时间，并方便维护人员在线预览。

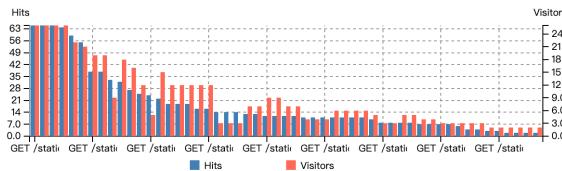
在线版本可参考<https://goaccess-securepapersnake.app.secoder.net/>，此处由于软件工程平台自身的安全性远低于我们的标准设计，以及方便第三方检查，只能部署在公网之上。



STATIC REQUESTS

TOP STATIC REQUESTS SORTED BY HITS [AVGTS, CUMTS, MAXTS, MTHD, PROTO]

Panel Options ▾



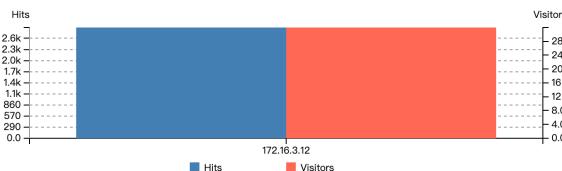
#	Hits	Visitors	Bandwidth	Avg. T.S.	Cum. T.S.	Max. T.S.	Method
992	467	93.88 MiB	0.00 us	0.00 us	0.00 us	0.00 us	
Max: 65	Max: 26	Max: 49.68 MiB	Max: 0.00 us	Max: 0.00 us	Max: 0.00 us	Max: 0.00 us	
Min: 2	Min: 2	Min: 982 B	Min: 0.00 us	Min: 0.00 us	Min: 0.00 us	Min: 0.00 us	
1	65 (6.55%)	26 (5.57%)	9.35 MiB (9.96%)	0.00 us	0.00 us	0.00 us	GET
2	65 (6.55%)	26 (5.57%)	26.87 MiB (28.62%)	0.00 us	0.00 us	0.00 us	GET
3	65 (6.55%)	26 (5.57%)	49.68 MiB (52.93%)	0.00 us	0.00 us	0.00 us	GET
4	64 (6.45%)	26 (5.57%)	142.86 KiB (0.15%)	0.00 us	0.00 us	0.00 us	GET
5	59 (5.95%)	22 (4.71%)	963.87 KiB (1.00%)	0.00 us	0.00 us	0.00 us	GET
6	55 (5.54%)	21 (4.50%)	787.03 KiB (0.82%)	0.00 us	0.00 us	0.00 us	GET
7	38 (3.83%)	19 (4.07%)	43.03 KiB (0.04%)	0.00 us	0.00 us	0.00 us	GET
8	38 (3.83%)	19 (4.07%)	89.23 KiB (0.09%)	0.00 us	0.00 us	0.00 us	GET
9	33 (3.33%)	9 (1.93%)	206.86 KiB (0.22%)	0.00 us	0.00 us	0.00 us	GET
10	32 (3.23%)	18 (3.85%)	211.73 KiB (0.22%)	0.00 us	0.00 us	0.00 us	GET
11	27 (2.72%)	16 (3.43%)	192.71 KiB (0.20%)	0.00 us	0.00 us	0.00 us	GET
12	25 (2.52%)	12 (2.57%)	8.92 KiB (0.01%)	0.00 us	0.00 us	0.00 us	GET
13	24 (2.42%)	5 (1.07%)	96.24 KiB (0.10%)	0.00 us	0.00 us	0.00 us	GET
14	22 (2.22%)	15 (3.21%)	557.4 KiB (0.58%)	0.00 us	0.00 us	0.00 us	GET
15	19 (1.92%)	12 (2.57%)	190.02 KiB (0.20%)	0.00 us	0.00 us	0.00 us	GET
16	19 (1.92%)	12 (2.57%)	13.59 KiB (0.01%)	0.00 us	0.00 us	0.00 us	GET
17	19 (1.92%)	12 (2.57%)	197.51 KiB (0.21%)	0.00 us	0.00 us	0.00 us	GET
18	16 (1.61%)	12 (2.57%)	11.6 KiB (0.01%)	0.00 us	0.00 us	0.00 us	GET
19	16 (1.61%)	12 (2.57%)	130.55 KiB (0.14%)	0.00 us	0.00 us	0.00 us	GET
20	14 (1.41%)	3 (0.64%)	148.97 KiB (0.15%)	0.00 us	0.00 us	0.00 us	GET

< >

VISITOR HOSTNAMES AND IPS

TOP VISITOR HOSTS SORTED BY HITS [AVGTS, CUMTS, MAXTS]

Panel Options ▾



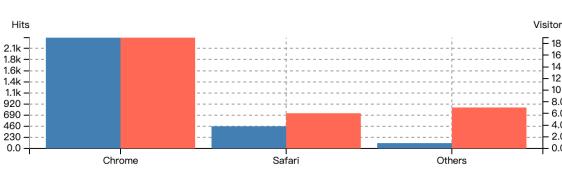
#	Hits	Visitors	Bandwidth	Avg. T.S.	Cum. T.S.	Max. T.S.
2,843	32	169.87 MiB	29.30 ms	1.39 m	1.39 n	
Max: 2,843	Max: 32	Max: 169.87 MiB	Max: 29.30 ms	Max: 1.39 m	Max: 5.95 s	
Min: 2,843	Min: 32	Min: 169.87 MiB	Min: 29.30 ms	Min: 1.39 m	Min: 5.95 s	
1	2,843 (100.00%)	32 (100.00%)	169.87 MiB (100.00%)	29.30 ms	1.39 m	5.95

< >

BROWSERS

TOP BROWSERS SORTED BY HITS [AVGTS, CUMTS, MAXTS]

Panel Options ▾

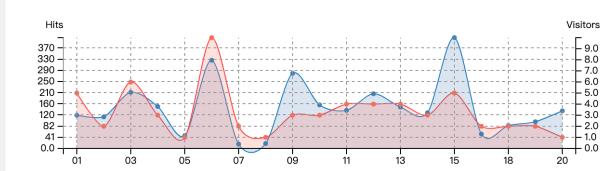


#	Hits	Visitors	Bandwidth	Avg. T.S.	Cum. T.S.	Max. T.S.
2,843	32	169.87 MiB	29.30 ms	1.39 m	1.39 n	
Max: 2,843	Max: 11	Max: 81.66 MiB	Max: 49.11 s	Max: 5.95 s	Min: 0.00 us	Min: 2.00 ms
Min: 1	Min: 1	Min: 2.26 KIB	Min: 0.00 us	Min: 0.00 us	Min: 0.00 us	Min: 0.00 us
1	1,372 (48.26%)	9 (28.12%)	81.66 MiB (48.07%)	35.79 ms	49.11 s	5.95 s
2	1,295 (45.55%)	13 (40.62%)	72.24 MiB (42.53%)	24.06 ms	31.16 s	1.34 s
3	132 (4.64%)	6 (18.75%)	11.15 MiB (6.57%)	20.81 ms	2.75 s	747.00 ms
4	25 (0.88%)	1 (3.12%)	2.7 MiB (1.59%)	11.04 ms	276.00 ms	46.00 ms
5	12 (0.42%)	2 (6.25%)	2.1 MiB (1.24%)	333.00 us	4.00 ms	2.00 ms
6	7 (0.25%)	1 (3.12%)	15.05 KIB (0.01%)	0.00 us	0.00 us	0.00 us

TIME DISTRIBUTION

DATA SORTED BY HOUR [AVGTS, CUMTS, MAXTS]

Panel Options ▾

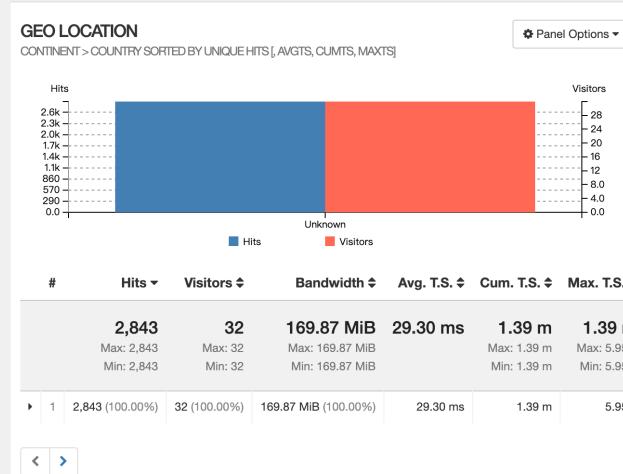
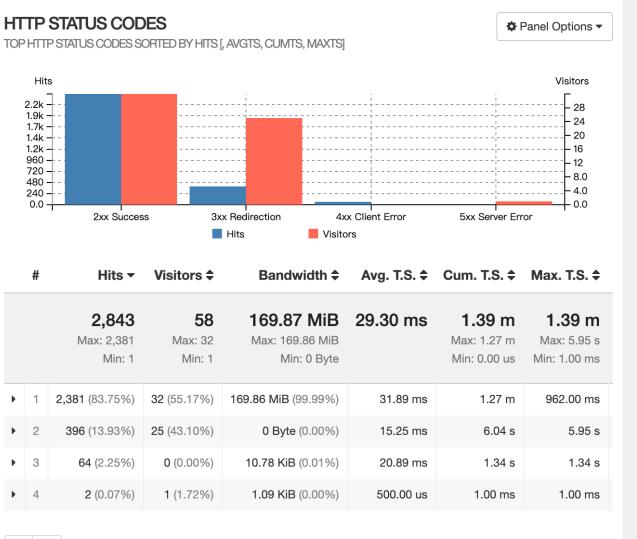
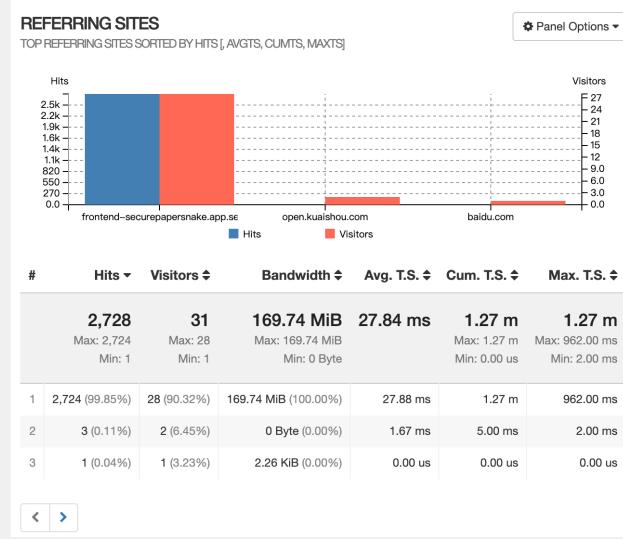


#	Hits	Visitors	Bandwidth	Avg. T.S.	Cum. T.S.	Max. T.S.
1	2,843 (80.30%)	32 (59.38%)	169.87 MiB (79.82%)	29.30 ms	1.39 m	1.39 m
	Max: 1,717 Min: 1	Max: 12 Min: 1	Max: 102.9 MiB Min: 2.26 KIB	Max: 50.39 s Min: 0.00 us	Max: 5.95 s Min: 2.00 ms	
2	455 (16.00%)	6 (18.75%)	21.7 MiB (12.78%)	15.84 ms	7.21 s	216.00 ms
3	105 (3.69%)	7 (21.88%)	12.58 MiB (7.40%)	20.81 ms	2.19 s	747.00 ms

◀ ▶

#	Hits	Visitors	Bandwidth	Avg. T.S.	Cum. T.S.	Max. T.S.	1
1	122 (4.29%)	5 (7.94%)	6.08 MiB (3.58%)	14.54 ms	1.77 s	159.00 ms	0
2	116 (4.08%)	2 (3.17%)	6.14 MiB (3.61%)	16.51 ms	1.92 s	216.00 ms	0
3	206 (7.25%)	6 (9.52%)	9.75 MiB (5.74%)	14.44 ms	2.97 s	188.00 ms	0
4	154 (5.42%)	3 (4.76%)	10.74 MiB (6.32%)	29.89 ms	4.60 s	492.00 ms	0
5	47 (1.65%)	1 (1.59%)	2.06 MiB (1.21%)	27.02 ms	1.27 s	362.00 ms	0
6	324 (11.40%)	10 (15.87%)	27.24 MiB (16.04%)	31.79 ms	10.30 s	649.00 ms	0
7	16 (0.56%)	2 (3.17%)	1.1 MiB (0.65%)	76.69 ms	1.23 s	747.00 ms	0
8	18 (0.63%)	1 (1.59%)	2.31 MiB (1.36%)	23.67 ms	426.00 ms	277.00 ms	0
9	276 (9.71%)	3 (4.76%)	11.48 MiB (6.76%)	22.54 ms	6.22 s	417.00 ms	0
10	160 (5.63%)	3 (4.76%)	4.45 MiB (2.62%)	23.07 ms	3.69 s	712.00 ms	1
11	140 (4.92%)	4 (6.35%)	10.47 MiB (6.16%)	50.38 ms	7.05 s	961.00 ms	1
12	201 (7.07%)	4 (6.35%)	13.72 MiB (8.08%)	52.34 ms	10.52 s	962.00 ms	1
13	152 (5.35%)	4 (6.35%)	7.6 MiB (4.48%)	31.71 ms	4.82 s	709.00 ms	1
14	131 (4.61%)	3 (4.76%)	4.07 MiB (2.40%)	14.05 ms	1.84 s	175.00 ms	1
15	407 (14.32%)	5 (7.94%)	23.87 MiB (14.05%)	34.84 ms	14.18 s	5.95 s	1
16	53 (1.86%)	2 (3.17%)	9.86 MiB (5.80%)	30.47 ms	1.61 s	262.00 ms	1
17	84 (2.95%)	2 (3.17%)	3.49 MiB (2.05%)	54.87 ms	4.61 s	1.34 s	1
18	98 (3.45%)	2 (3.17%)	6.01 MiB (3.54%)	13.00 ms	1.27 s	282.00 ms	1
19	138 (4.85%)	1 (1.59%)	9.43 MiB (5.55%)	21.59 ms	2.98 s	252.00 ms	2

◀ ▶



前端

框架: nodejs + Vue + ElementUI + Echarts

Build Setup

```
git clone https://gitlab.secoder.net/securepapersnake/frontend  
cd frontend  
yarn install  
yarn dev
```

浏览器访问 <http://localhost:8000>

发布

```
# 构建生产环境  
yarn build
```

调试

```
# 构建开发环境  
yarn dev  
  
# 预览发布环境效果 (反向代理后端api)  
HOST=<api_url> yarn preview  
  
# 预览发布环境效果 + 静态资源分析  
HOST=<api_url> yarn preview --report  
  
# 代码格式检查  
yarn lint  
  
# 代码格式检查并自动修复  
yarn lint --fix  
  
# Unit-test
```

```

> git commit -m 'feat(all): unit-test

fix all unit-test, update coverage to almost 100%
yarn run v1.22.10
$ eslint --fix --ext .js,.vue src
✨ Done in 2.81s.
yarn run v1.22.10
$ jest --clearCache && vue-cli-service test:unit
Cleared /private/var/folders/mm/p89jxztx2j1bchyhx_k8d7qhh0000gn/T/jest_dx
PASS  tests/unit/components/Breadcrumb.spec.js
PASS  tests/unit/utils/parseTime.spec.js
PASS  tests/unit/utils/formatTime.spec.js
PASS  tests/unit/views/Notice.spec.js (9.993s)
PASS  tests/unit/layout/layout.spec.js
PASS  tests/unit/views/Video.spec.js (10.32s)
PASS  tests/unit/views/Videolist.spec.js (10.419s)
PASS  tests/unit/views/Register.spec.js (10.323s)
PASS  tests/unit/views/Login.spec.js (10.56s)
PASS  tests/unit/views/Home.spec.js (10.791s)
PASS  tests/unit/components/Hamburger.spec.js
PASS  tests/unit/utils/validate.spec.js
PASS  tests/unit/components/SvgIcon.spec.js
PASS  tests/unit/views/Tree.spec.js
PASS  tests/unit/utils/param2Obj.spec.js
PASS  tests/unit/utils/get-page-title.spec.js
PASS  tests/unit/views/Form.spec.js
PASS  tests/unit/views/404.spec.js

===== Coverage summary =====
Statements : 100% ( 291/291 )
Branches   : 100% ( 83/83 )
Functions   : 97.59% ( 81/83 )
Lines      : 100% ( 284/284 )
=====

Test Suites: 18 passed, 18 total
Tests:      55 passed, 55 total
Snapshots:  0 total
Time:       15.174s
Ran all test suites.
✨ Done in 17.62s.

```

后端

框架：主体使用django，数据库mysql，后端api文档：restful

后端部署运行：

```

pip install -r requirements.txt
sh script/hook.sh
sh script/start.sh

```

config文件：后端的所有有关密码均储存在了位于服务器的文件中，一个config文件的内容示例如下：

```
{  
    "DJANGO_SUPERUSER_EMAIL": "prnake@gmail.com",  
    "DJANGO_SUPERUSER_USERNAME": "papersnake",  
    "DJANGO_SUPERUSER_PASSWORD": "password",  
    "DJANGO_SECRET_KEY": "key",  
    "DJANGO_DEBUG": "False",  
    "MYSQL_NAME": "django",  
    "MYSQL_USER": "root",  
    "MYSQL_PASSWORD": "password",  
    "MYSQL_HOST": "mysql.securepapersnake.secoder.local",  
    "MYSQL_PORT": 3306,  
    "app_id": "ks12345678",  
    "app_secret": "12345678",  
    "redirection_uri": "https://frontend-  
securepapersnake.app.secoder.net/api/kuaishou/handle-code",  
    "refresh_time": 3600,  
    "renew_time": 3600,  
    "salt": "salt",  
    "admin": "testuser"  
}
```

```
> .hook/pre-commit  
===== test session starts =====  
platform darwin -- Python 3.8.5, pytest-6.2.2, py-1.10.0, pluggy-0.13.1  
django: settings: secoder.settings (from ini)  
rootdir: /Users/pka/Develop/secoder/backend, configfile: pytest.ini  
plugins: django-4.1.0  
collected 8 items  
  
tests/ping/testping.py .... [ 50%]  
tests/user/testuser.py . [ 62%]  
tests/usercenter/testnotice.py ... [100%]  
  
===== 8 passed in 2.18s =====  
  
-----  
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

Secoder 部署流程

详情可参考 **frontend**, **backend**, **goaccess**, **mysql**, **phpmyadmin** 五个仓库的 **Dockerfile**。

13. 项目工作流程示意图

