树莓拍开发文档

1. 项目介绍

七牛云1024编程马拉松作品。

1.1 基础功能（赛题最低要求，已实现）

- 视频播放：播放、暂停、进度条拖拽
- 内容分类：视频内容分类页，如热门视频、体育频道
- 视频切换：可通过上下键翻看视频

1.2 高级 + 自行实现功能

- 视频处理：视频转码、封面生成
- 视频播放：HLS（m3u8）播放
- 搜索功能：分词、标签检索
- 创作中心：视频上传、查看播放、查看点赞、收藏数据
- 账户系统：微信扫码登录、个人信息修改
- 视频互动：视频点赞、评论点赞、收藏、关注、取关

2. 项目分工

前端：李建君 后端：方宇杰

3. 项目实现

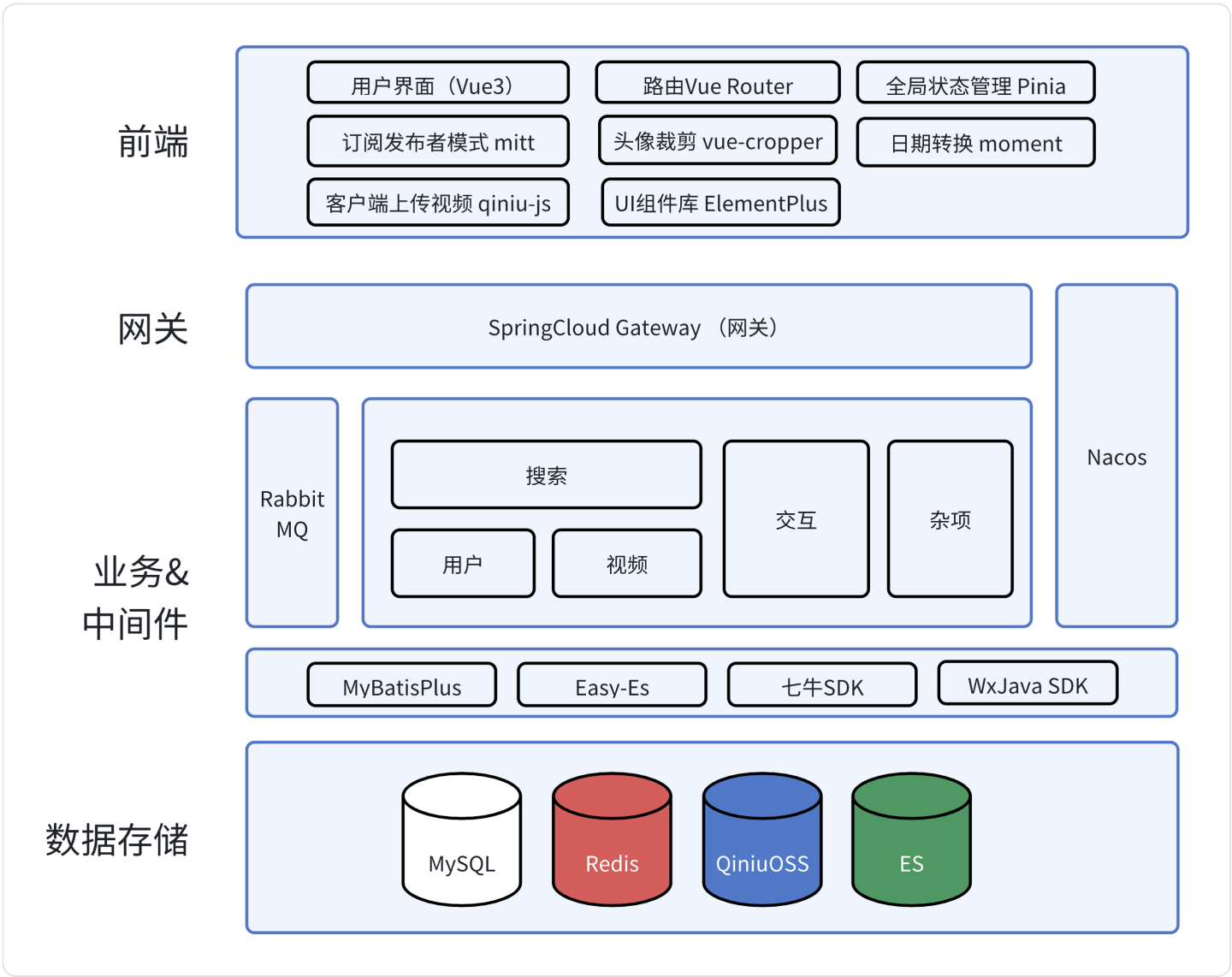
3.1 技术选型

前端：Vue3 + JavaScript + Vue Router + Pinia + ElementPlus + hls.js

后端：SpringCloud + MyBatisPlus + Easy-Es + WxJava + OpenFeign + Nacos + QiniuSDK

中间件&数据库：MySQL、Redis、ElasticSearch、RabbitMQ

3.2 架构设计



3.3 主要业务实现

3.3.1 视频播放

组件封装:

- 视频播放器
- 视频大屏查看组件
- 视频列表组件
- 评论组件
- Model模态框封装(基于ElementPlus二次封装)
- 头像上传组件
- 用户个人基本信息卡片
- 扫码登录组件
- 搜索组件

自定义指令:

- v-scroll

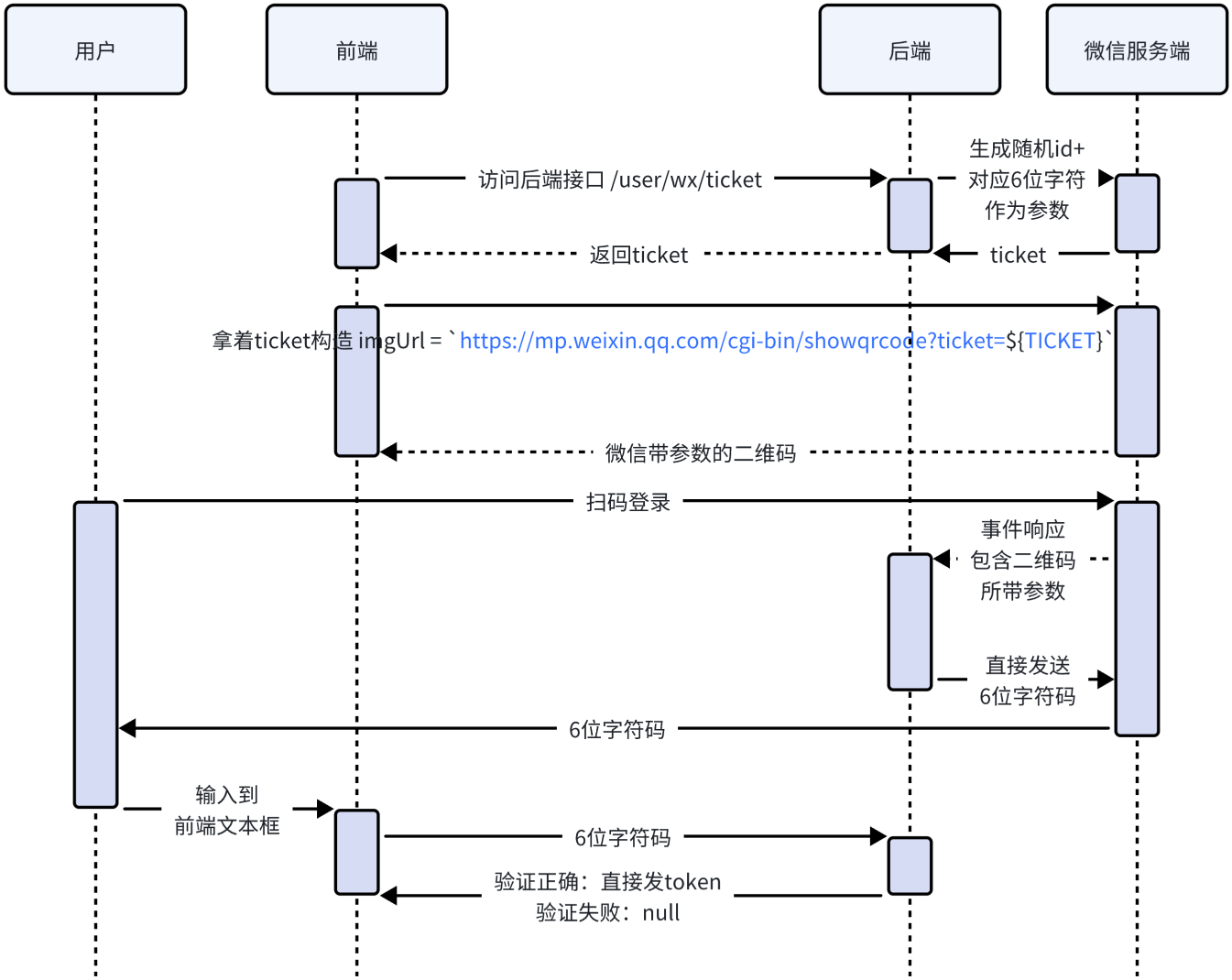
自定义样式

自定义工具函数:

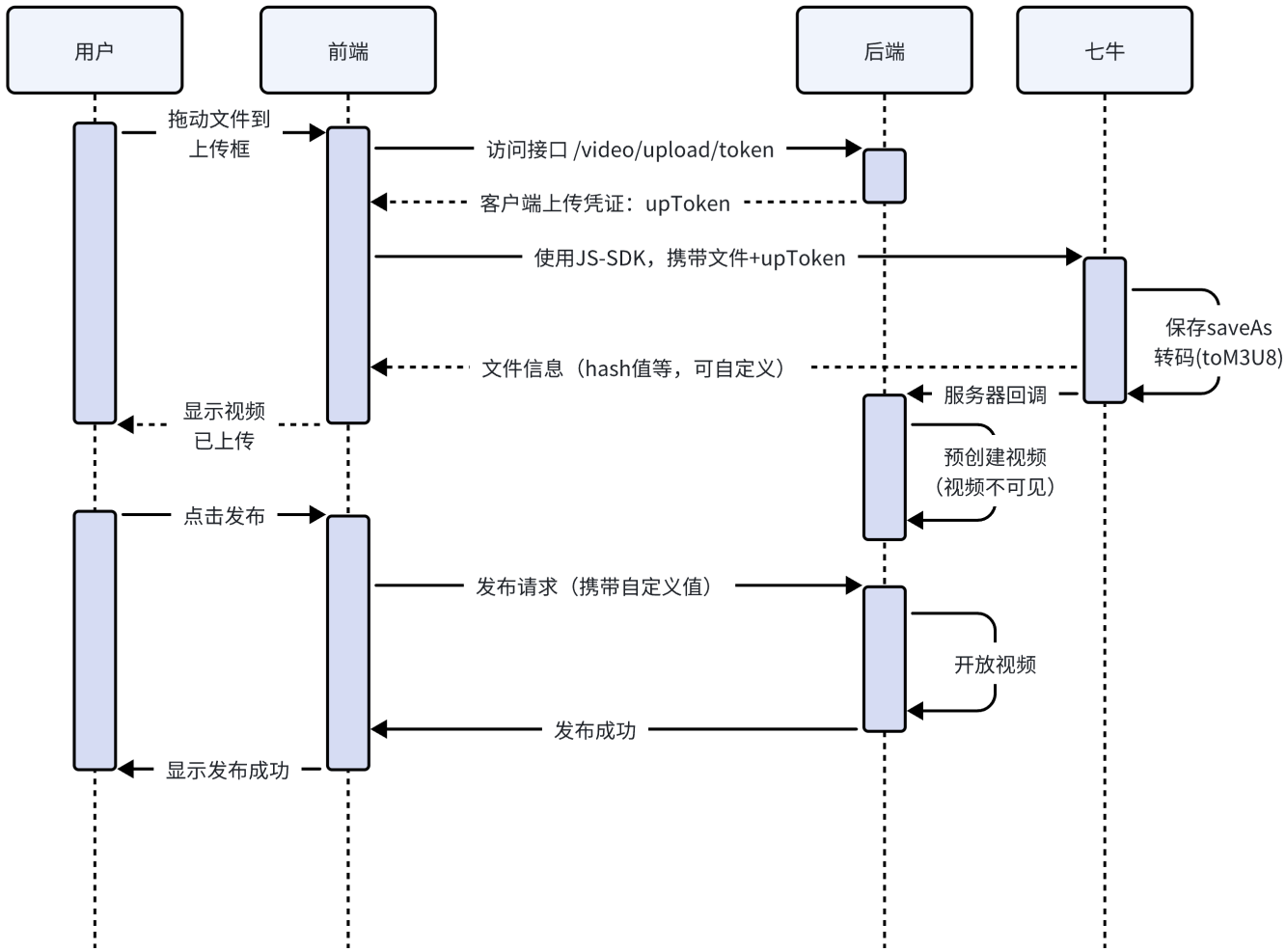
```
import { muted, continuous, isCommoning } from './videoConfig'
import request from './request'
import { getToken, setToken, clearToken } from './token'
import { createUuid } from './createUuid'
import { getUpToken, setUpToken, clearUpToken } from './upToken'
import { getUserInfoStorage, setUserInfoStorage, clearUserInfoStorage } from
import { isLogin } from './validate'
import { parseTime, fromTime } from './parseTime'
import { getChannelListStroge, setChannelListStroge } from './channelList'
import { emitter } from './emitter'

// 统一导出
export {
  // 视频是否支持连续播放
  continuous,
  // 视频是否静音
  muted,
  isCommoning,
  // 二次封装axios
  request,
  // token持久化
  getToken,
  setToken,
  clearToken,
  // uuid
  createUuid,
  // uptoken持久化
  getUpToken,
  setUpToken,
  clearUpToken,
  // userInfo持久化
  getUserInfoStorage,
  setUserInfoStorage,
  clearUserInfoStorage,
  // 判断当前是否登录状态
  isLogin,
  // 日期时间格式转换
  parseTime,
  // 将时间转换为 n天前
```

```
    fromTime,
    // 频道持久化
    getChannelListStroge,
    setChannelListStroge,
    // 订阅发布者
    emitter
}
```

3.3.2 微信扫码登录



3.3.3 客户端上传视频流程



3.3.4 点赞、收藏、评论点赞、关注

> 先将数据存在Redis里，再通过定时任务持久化到MySQL中。

由于点赞、收藏、评论点赞、关注这几个业务比较相似（都是由一个id对另一个id执行的操作且状态为true/false两种），所以通过写一个通用的 `ActionService`，实现代码复用。

MAP_KEY_${REDIS_KEY}	
${fromId}::${toId}	0 or 1
${fromId}::${toId}	0 or 1
...	...

记录fromId对toId进行action操作的Hash表

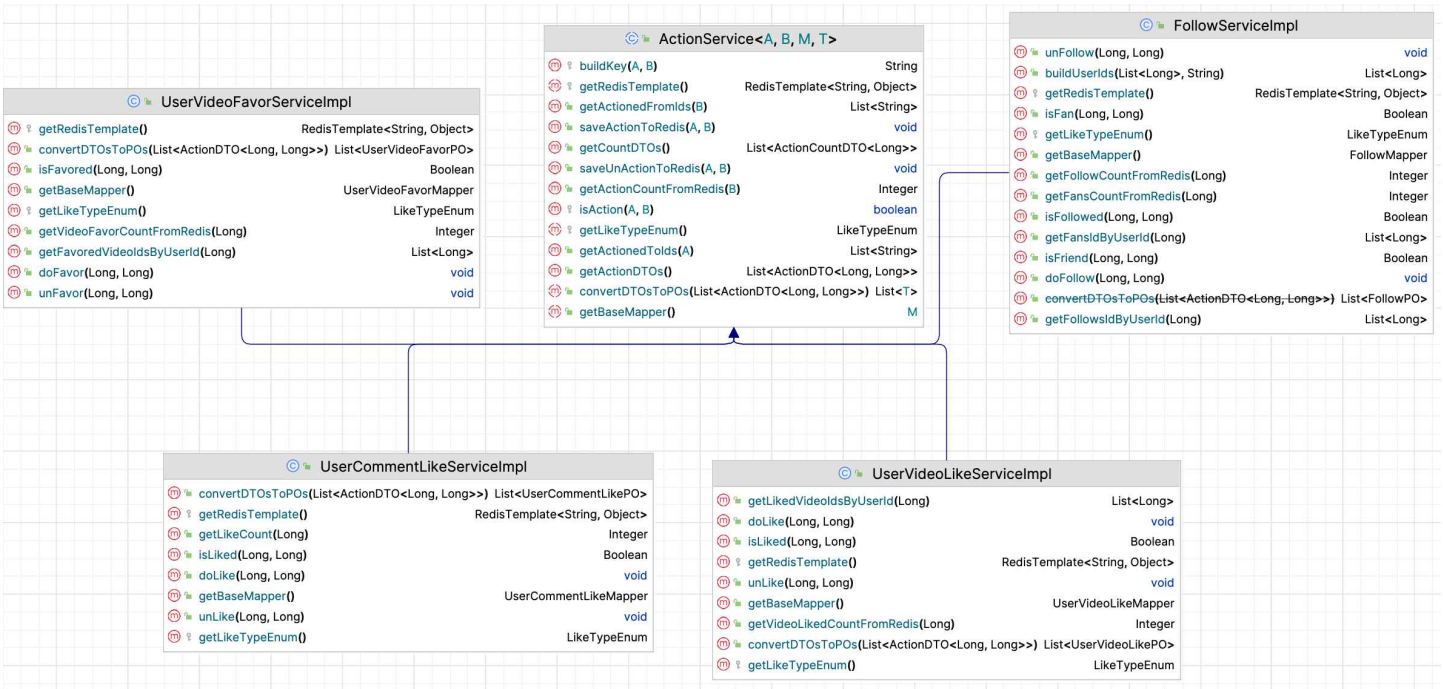
MAP_KEY_${REDIS_KEY_COUNT}	
${toId}	count(${toid})
${toId}	count(${toid})
...	...

记录toId被进行action操作的计数Hash表

其中的${}中的部分值在一个枚举类中被定义

```
@Getter
@AllArgsConstructor
public enum LikeTypeEnum implements Serializable {
```

```
    VIDEO(1, "视频点赞", "MAP_KEY_VIDEO_LIKED", "id", "userId", "videoId", "statu
    COMMENT(2, "评论点赞", "MAP_KEY_VIDEO_COMMENT_LIKED", "id", "userId", "commen
    FAVOR(3, "收藏点赞", "MAP_KEY_VIDEO_FAVORED", "id", "userId", "videoId", "sta
    FOLLOW(4, "关注", "MAP_KEY_USER_FOLLOW", "id", "fromId", "toId", "status");

    @EnumValue
    private final Integer code;
    private final String desc;
    private final String redisKey;
    private String dbIdColumn = "id";
    private String dbFromIdColumn = "userId";
    private String dbToIdColumn = "videoId";
    private String dbStatusColumn = "status";

    public String getRedisHashCountKey() {
        return redisKey + "_COUNT";
    }
}
```



ActionService中各方法通过 `getRedisTemplate()` 和 `getLikeTypeEnum()` 获取实现类的redisTemplate及类型，由于不同表的 主键ID类型可能不同，这里使用泛型设计：<A,B,M,T>，其中 A代表fromId，B代表toId，M代表其业务在MySQL里对应表的mybatis mapper接口类类型，T代表MySQL对应表里的pojo类类型。

3.3.5 搜索

借助了 ElasticSearch的分词查询功能，Java调用的部分使用了开源项目 `Easy-Es` 简化。

当第一次运行项目时，berry-search服务中的某个Manager类实现的 InitializingBean 接口中会先判断索引Index是否存在（即Es里与MySQL中user、video表的对应同步），不存在则执行全量同步，将MySQL中的数据批量插入到ES中。之后每隔一段时间，会执行增量定时任务；从 MySQL中查询updateTime在两倍时间间隔内的数据，将其更新到Es中。

查询数据时，前端先请求后端服务，后端服务将keyword送至ES进行分词的分页查询，获取结果的id集合，再用id集合去MySQL查询数据，最后封装成一个分页类返回给前端。