



**本科毕业设计**

题 目 短视频分享应用及后台管理系统

专 业 软件工程

作者姓名 孟贺

学 号 2016205645

单 位 计算机学院

指导教师 赵传申

**2020 年 5 月**

**教务处编**

原创性声明

本人郑重声明：所提交的学位论文是本人在导师指导下，独立进行研究取得的成果。除文中已经引用的内容外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得聊城大学或其他教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均在文中以明确的方式表明。本人承担本声明的相应责任。

学位论文作者签名： 日期：

指 导 教 师 签 名: 日期：

**短视频分享应用及后台管理系统**

# 引言

## 1.1 开发目的和开发意义

类比传统的局限于文字图片类的社交软件，视频更加生动有趣，内容更丰富，随着4G/5G的普及也使得观看视频更加流畅方便，用户观看体验更好。同时在生活节奏不断加快的生活中，短视频不需要大量占用整段的时间，更适合碎片化的时间分布，也给紧凑的工作生活带来简短而有效的休闲放松。基于这样的原因，我希望开发并分析一个短视频应用的整个建设过程，无论是对自己的提升还是帮助别人学习，都是很有益处的。

## 1.2 本文开发内容

本文开发了一个短视频APP和它的后台管理系统，实现了视频的发布，在线观看，点赞，评论及简单的后台管理等功能。后端采用springboot ，spring security，mybatis-plus等框架实现，使用nginx和ffmpeg搭建了一个简单的流媒体服务。前端APP使用flutter实现，后端管理页面使用VUE实现。

# 2.开发语言与技术简介

## 2.1 Java简介

Java是一种基于类、面向对象的编程语言。Java的目的是让应用程序开发人员编写一次，在任何地方运行（WORA），这意味着编译过的Java代码可以在所有支持Java的平台上运行，而无需重新编译。Java应用程序通常被编译成字节码，可以在任何Java虚拟机（JVM）上运行，而不管底层的计算机架构如何。java的语法类似于C++和C++，但它们的底层设备比它们中的任何一个都要低。截至2019年，根据GitHub的说法，Java是最流行的编程语言之一，特别是用于客户机-服务器web应用程序，据报道有900万开发人员。

## 2.2 Spring 简介

IntelliJ IDEA是一个收费的一系列的开发工具之首的开发工具，不仅适用于JVM且功能强大并符合人体工程学的IDE（Integrated Development Environment），高级版本支持多种语言，更有多种强大的功能使得写代码变得更容易，例如：自动选取、导航模式、JUnit的完美支持、Debug、融合Git和SVN等的代码版本控制功能。

## 2.3 Dart和Flutter 简介

Dart是一种针对多平台应用程序客户端开发优化编程语言。它由Google开发，用于构建移动、桌面、服务器和web应用程序。Dart是面向对象、基于类、垃圾收集的语言，具有C语言风格的语法。Dart可以编译为机器语言或JavaScript。它支持接口、混合、抽象类、具体化泛型和类型推断。

flutter 是 google 推出的新一代跨平台渲染框架. 它帮助开发者解决了跨平台，高性能，富有表现力和灵活的 UI 表达，快速开发等核心问题。在Flutter中，一个组件代表用户界面中[不可变](https://zh.wikipedia.org/wiki/%E4%B8%8D%E5%8F%AF%E8%AE%8A%E7%89%A9%E4%BB%B6)的一部分；包括文本、多边形以及动画在内的所有图形都是用组件创建的。

Fish Redux 是一个基于 Redux 数据管理的组装式 flutter 应用框架， 它特别适用于构建中大型的复杂应用。它的特点是配置式组装。 一方面我们将一个大的页面，对视图和数据层层拆解为互相独立的 Component|Adapter，上层负责组装，下层负责实现； 另一方面将 Component|Adapter 拆分为 View，Reducer，Effect 等相互独立的上下文无关函数。所以它会非常干净，易维护，易协作。

## 2.4 HLS和Ffmpeg 简介

HTTP Live Streaming（缩写是HLS）是由[苹果公司](https://zh.wikipedia.org/wiki/%E8%8B%B9%E6%9E%9C%E5%85%AC%E5%8F%B8)提出基于[HTTP](https://zh.wikipedia.org/wiki/HTTP)的[流媒体](https://zh.wikipedia.org/wiki/%E6%B5%81%E5%AA%92%E4%BD%93)[网络传输协议](https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE)。它的工作原理是把整个流分成一个个小的基于HTTP的文件来下载，每次只下载一些。当媒体流正在播放时，客户端可以选择从许多不同的备用源中以不同的速率下载同样的资源，允许流媒体会话适应不同的数据速率。在开始一个流媒体会话时，客户端会下载一个包含元数据的[extended M3U (m3u8)](https://zh.wikipedia.org/w/index.php?title=Extended_M3U&action=edit&redlink=1) [playlist](https://zh.wikipedia.org/w/index.php?title=Playlist&action=edit&redlink=1)文件，用于寻找可用的媒体流。

FFmpeg是一个免费的开放源代码项目，包含用于处理视频，音频以及其他多媒体文件和流的库和软件。 FFmpeg专门用于基于命令行的视频和音频文件处理，并且广泛用于格式转码，基本编辑（修剪和串联），视频缩放，视频后期制作效果。

# 3、应用分析和设计

## 3.1应用需求分析

### 3.1.1 视频创作

用户可以从相册中选择一个或从相机拍摄一个视频，选择视频的一帧作为视频的封面，用户可以为视频推荐一个标题。为了节省用户的流量，提供一个是否压缩视频选项，用户可选择是否压缩视频后再上传。

3.1.2 视频观看

APP提供一个推荐的视频列表，用户可以随意选择观看，除了视频本身，APP也应该展现出该视频的评论和视频的点赞数，因此用户也能对视频进行评论和点赞这两种操作。

3.1.3 用户信息查看

用户应该能够在APP中查看自己或其他用户的个人信息和该用户发布的内容，用户能够通过该页选择该用户的任意视频进行观看。

3.1.4 关注

用户应该能够关注除了自己之外的任意用户，用户也能够查看到自己已关注的用户列表，进行取关操作。同样用户也应该能够看到关注自己的人。

# 4、应用详细设计

## 4.1数据库设计

**4.1.1表名：tape\_clip**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | title | 标题 | varchar | 255 | YES | null |
| 3 | source\_file | 上传源文件存放路径 | varchar | 255 | NO | null |
| 4 | clip\_path | 视频存放路径 | varchar | 255 | NO | null |
| 5 | cover\_path | 封面存放路径 | varchar | 255 | NO | null |
| 6 | creator | 创建人ID | bigint | null | NO | null |
| 7 | update\_time | 更新时间 | datetime | null | YES | null |
| 8 | update\_man | 更新人ID | bigint | null | YES | null |
| 9 | create\_time | 创建时间 | datetime | null | NO | null |
| 10 | create\_man | 创建人ID | bigint | null | NO | null |
| 11 | is\_deleted | 删除标志 0否，1是 | int | null | NO | 0 |

**4.1.2表名：tape\_comment**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | text | 评论内容 | varchar | 255 | NO | null |
| 3 | to\_id | 被评论人ID | bigint | null | NO | null |
| 4 | from\_id | 评论人ID | bigint | null | NO | null |
| 5 | clip\_id | 视频ID | bigint | null | NO | null |
| 6 | update\_time | 更新时间 | datetime | null | YES | null |
| 7 | update\_man | 更新人ID | bigint | null | YES | null |
| 8 | create\_time | 创建时间 | datetime | null | NO | null |
| 9 | create\_man | 创建人ID | bigint | null | NO | null |
| 10 | is\_deleted | 删除标志 0否，1是 | int | null | NO | null |

**4.1.3表名：tape\_like**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | like\_type | 点赞类型 1视频，2评论 | int | null | NO | null |
| 3 | from\_id | 点赞人ID | bigint | null | NO | null |
| 4 | to\_id | 被点赞人ID | bigint | null | NO | null |
| 5 | target\_id | 视频或评论ID | bigint | null | NO | null |
| 6 | update\_time | 更新时间 | datetime | null | YES | null |
| 7 | update\_man | 更新人ID | bigint | null | YES | null |
| 8 | create\_time | 创建时间 | datetime | null | NO | null |
| 9 | create\_man | 创建人ID | bigint | null | NO | null |
| 10 | is\_deleted | 删除标志 0否，1是 | int | null | NO | 0 |

**4.1.4表名：tape\_user**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | username | 用户名 | varchar | 255 | NO | null |
| 3 | password | 密码 | varchar | 255 | NO | null |
| 4 | nickname | 昵称 | varchar | 255 | YES | null |
| 5 | avatar | 头像存放路径 | varchar | 255 | YES | null |
| 6 | signature | 签名 | varchar | 255 | YES | null |
| 7 | update\_time | 更新时间 | datetime | null | YES | null |
| 8 | update\_man | 更新人ID | bigint | null | YES | null |
| 9 | create\_time | 创建时间 | datetime | null | NO | null |
| 10 | create\_man | 创建人ID | bigint | null | NO | null |
| 11 | is\_deleted | 删除标志 0否,1是 | int | null | NO | 0 |

**4.1.5表名：tape\_friend**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | follower | 粉丝ID | bigint | null | NO | null |
| 3 | following | 被关注ID | bigint | null | NO | null |
| 4 | update\_time | 更新时间 | datetime | null | YES | null |
| 5 | update\_man | 更新人ID | bigint | null | YES | null |
| 6 | create\_time | 创建时间 | datetime | null | NO | null |
| 7 | create\_man | 创建人ID | binary | 255 | NO | null |
| 8 | is\_deleted | 删除标志 0否，1是 | int | null | NO | 0 |

**4.1.6表名：tape\_authority**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | int | null | NO | null |
| 2 | authority | 权限名 | varchar | 255 | NO | null |

**4.1.7表名：tape\_user\_authority**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **字段名称** | **字段描述** | **字段类型** | **长度** | **允许空** | **缺省值** |
| 1 | id | 唯一ID | bigint | null | NO | null |
| 2 | user\_id | 用户ID | bigint | null | NO | null |
| 3 | authority\_id | 权限ID | int | null | NO | null |
| 4 | update\_time | 更新时间 | datetime | null | YES | null |
| 5 | update\_man | 更新人ID | bigint | null | YES | null |
| 6 | create\_time | 创建时间 | datetime | null | NO | null |
| 7 | create\_man | 创建人ID | bigint | null | NO | null |
| 8 | is\_deleted | 删除标志 0否,1是 | int | null | NO | 0 |

4.2 Redis字段

1、用户给视频点赞：user:{userId}:like:clip:{clipId}

2、用户给评论点赞：user:{userId}:like:comment:{commentId}

3、视频收到的点赞：clip:{clipId}:likedCount

4、视频收到的评论：clip:{clipId}:commentCount

5、评论收到的点赞：comment:{commentId}:likedCount

# 4、功能实现

4.1移动端APP设计

4.1.1APP整体结构

在App工程的主目录lib主要有api、customer\_widgets、models、page、utils五个目录。(1)api下存放了在dart中流行的http请求库的二次封装，以及app中的接口封装。

(2)customer\_widgets 下存放了自定义的播放器组件。

(3)models下存放了与后端pojo对应的类，user,clip,friend,comment,like等类。

(4)page 下存放了app中页面，主要有登录注册，首页，消息，视频详情，个人资料，创作等几个页面。

(5)utils存放了两个工具类，对Shared preferences(数据持久存储支持)封装的storage类，和获取屏幕信息的adapt类。

此外lib下还有app，main，main.reflectable，main是程序的入口，在main中会拿出存储的token做出用户是否登录的判断，动态改变App的首页。 App.dart 中会初始化一个页面的路由表，方便页面间的跳转。main.reflectable 由工具生成，提供了反射支持。

4.1.2请求的封装。

（1）为了更加方便使用dio的http请求库，在项目中提供了一个HTttpManagerl类，提供了通用的get和post请求，因为项目使用了token认证，所以添加了一个dio的onRequest拦截器，为所有的请求header加上认证的token。此外还设计了一个onResponse拦截器，对后端的返回结果进行封装。如下是Interceptors.dart中的部分代码：

onResponse(Response response) async {  
 var res = response.data;  
 if(res["code"]!=1){  
 Fluttertoast.*showToast*(msg: res["message"]);  
 return new ResultData(res["data"], false, res["code"], res["message"]);  
 }else{  
 return new ResultData(res["data"], true, res["code"], res["message"]);  
 }  
}

Future onRequest(RequestOptions options) {

var token = Storage.getString("token");

var header ={"Authorization":token};

options.headers.addAll(header);

return super.onRequest(options);

}

(2) 项目中几乎所有页面都要调用api，如果把调用api的逻辑分别写在页面里，项目会混乱不堪，为此项目中对所有后端提供的API进行了封装，每一个后端的controller类都对应了前端项目中的api类。如下是user\_api.dart中的部分方法：

static *login*(String username,String password){  
 var params = new Map<String,String>();  
 params["username"] = username;  
 params["password"] = password;  
 return HttpManager.*getInstance*().post("/api/user/login", params);  
}  
  
static *getUserInfo*(String username){  
 var params = new Map<String,String>();  
 params["username"] = username;  
 return HttpManager.*getInstance*().get("/api/user/info",params);  
}

4.1.3登录注册页

登录注册是App不可缺失的一部分，借助flutter的第三方widget，很容易做出简洁大方的登录注册页面。在本页中用户点击登录或注册会先进行数据的校验。在点击注册的情况下，如果后端返回成功标志则转向登录页。在登录时，如果后端验证密码成功，会返回一个token，应用会通过shared\_preferences将token持久化，在此后的其他请求中，都会在header中加入这个token验证用户身份。

 