万方检测 原文标注报告（原简明版）【个人文献版-新文献】报告编号 PL-20210504-6328371A-BZ

送检文献信息

【题名】nutstore

作　　者：nutstore

检测时间：2021-05-04 12:43:53

5.34%
总相似比

检测范围：

- [x] 中国学术期刊数据库
- [x] 优先出版论文数据库
- [x] 国内外重要学术会议论文数据库
- [x] 中国博士学位论文全文数据库
- [x] 中国优秀硕士学位论文全文数据库
- [x] 中国优秀报纸全文数据库
- [x] 互联网学术资源数据库
- [x] 学术网络文献数据库
- [x] 中国专利文献全文数据库
- [x] 特色英文文摘数据库
- [x] 中国标准全文数据库

详细检测结果

原文总字符数	检测字符数	参考文献相似比	辅助排除参考文献相似比	可能自引相似比	辅助排除可能自引相似比
28369	24770	0.20%	5.14%	0.00%	5.34%

相似文献列表（仅列举前10条）

序号	相似比(相似字符)	相似文献	类型	是否引用
1	0.66% 158字符	**流媒体** ；百度百科（网址：http://baike.baidu.com/view/794.html）；2008-04-20	学术网文	否
2	0.44% 106字符	**视频监控系统技术发展的探讨** 张钢， 东北高速公路股份有限公司， 刘辉， 东北高速公路股份有限公司， 杨寅权， 东北高速公路股份有限公司；《第十届中国高速公路信息化管理及技术研讨会》；2008-04-01	会议	否
3	0.41% 98字符	**一种测试用例设计方法和装置CN202010705925.1** 卡斯柯信号(北京)有限公司；INVENTION_PUBLICATION；2020-07-19 00:00:00.0000000	专利	否
4	0.41% 97字符	**基于Spring Boot的源代码在线评测系统的设计与实现** 詹皇彬（导师：韩建民）；浙江师范大学，硕士（专业：软件工程）；2016	学位	否
5	0.36% 87字符	**基于Android网络直播平台的设计与实现** 刘蕴倩（导师：黄虎杰;周智丽）；哈尔滨工业大学，硕士（专业：软件工程）；2019	学位	否
6	0.33% 80字符	**实时视频监控系统的设计与实现** 索郎邓珠（导师：杨刚;邹智勇）；西安电子科技大学，硕士（专业：软件工程）；2016	学位	否
7	0.28% 67字符	**可信高压缩视频监控网络相关问题研究** 任侨（导师：蒋体钢;刘金钟）；电子科技大学，硕士（专业：电子与信息工程）；2011	学位	否
8	0.26% 63字符	**网络视频监控系统现状与安全问题探究** 王涛；《网络安全技术与应用》；2020-09-28	期刊	否
9	0.23% 56字符	**智能船舶的数据分发方法、装置及系统CN202010206181.9** 上海船舶研究设计院(中国船舶工业集团公司第六0四研究院)；INVENTION_PUBLICATION；2020-03-21 00:00:00.0000000	专利	否
10	0.20% 47字符	**基于流媒体技术的数字化校园文化设计与实现** 万梅芬；《中国管理信息化》；2018-09-07	期刊	是

原文标注

班级 1703014

学号 17039110002

本科毕业设计论文

ヤレハオシハモヲモテヨミ」ャアセオトオ・ノォコへキエーラハケモテメイスマホェウ」シ訪ワャネ邏ォユ譯「クエモ。。「オ・ノォモ。ヒ「オネホ゛キィヤヒモテイハノォアセオトヌ鯀

ツ」ャシエソノイホユユオ・ノォキエーラへシミホヨニラケモテ。」ヤレハケモテハア」ャオラノォサメカネエレオネモレ50%」ャアセキエーラ。」

题目基于 springboot 的视频实

时监控系统的设计与实现

学院计算机科学与技术

专业计算机科学与技术

学生姓名陈鑫辉

导师姓名鱼滨

毕业设计（论文）诚信声明书

本人声明：本人所提交的毕业论文《基于 springboot的视频实时监控系统的设计与实现》是本人在指导教师指导下独立研究、写作成果，论文中所引用他人的无论以何种方式发布的文字、研究成果，均在论文中加以说明；有关教师、同学和其他人员对本文本的写作、修订提出过并为我在论文中加以采纳的意见、建议，均已在我的致谢辞中加以说明并深致谢意。

本文和资料若有不实之处，本人承担一切相关责任。

论文作者：（签字） 时间：年月日

指导教师已阅：（签字） 时间：年月日

摘要

摘要

关键词：流媒体传输技术 Spring Boot HTTP-FLV 实时视频监控系统

摘要

Abstract

Abstract

Key words: streaming media transmission protocol Spring Boot HTTP-FLV real-time video surveillance system

Abstract

目录 I

目录

第一章引言

1.1 研究目的和意义

视频监控系统是多媒体技术、计算机网络、工业控制和人工智能等技术的综合运用， 它正向着视频/音频的数字化、系统的网络化和管理的智能化方向不断发展 [1]。视频监控系统是安防领域中的研究热点，随着近年来各类智能设备数量的爆发性增长，视频监控系统正朝着数字化、智能化、网络化、人性化的方向发展。

它在交通违法抓拍、人脸识别辨别逃犯、停车场车牌识别等安防领域发挥着不容小觑的作用。

近些年来，随着人们生活水平的提高，视频监控已经深入每个人的生活，在它的帮助下人们能更好的保护自己和家人的人身以及财产安全。对于安装在公共场所的监控摄像头，其拍摄的视频数据和音频数据都会被相关部门收集并同一管理，而这类监控摄像头设备大多是使用电子线缆进行数据传输，会受到时间和地域的双重限制。但是安装在人们家中的监控摄像头就不可能做到由有关部门统一管理，更不可能在每个用户家中安装一个视频监控的查看终端，因此就需要一个基于网络的视频监控系统，可以提供在线观看、视频回放等功能。基于此，网络摄像头和网络视频监控系统应运而生。

以网络为基础的视频监控突破了对时间、地域的限制，只要有网络存在的地方就可以建立网络监控系统，省去了传统的布线和线路维护费用，降低了监控成本；用户在授权的情况下，就可以不受地域时间限制随时按需监控，实现即插、即用、即看 [2]。

1.2 视频监控的研究现状和发展趋势

1.2.1 研究现状

视频监控的兼容性。最近几年，监控摄像头的普及非常迅速，现在几乎每家每户都会安装有一个或几个监控摄像头，这些摄像头可能是同一家公司的产品，也

可能不是用一家公司的产品。这其中产生的问题就是兼容性问题，例如摄像头厂商 A的摄像头需要在平台 A上进行查看，摄像头厂商 B的摄像头需要在平台 B上进行查看，每个厂商都有自己的协议和平台。这就需要制定一个统一个标准，然后所有的厂商按照这个标准去生产摄像头和研发监控摄像头系统。

视频监控的智能化。现在市面上已经存在很多智能的监控摄像头，它们具有目标追踪、人物检测等功能。这些设备的出现，大大减少了人力成本，比如以前要实现摄像头跟踪一个目标，就需要手动去调整摄像头的朝向但是现在可以使用只能摄像头解决。因此要想转变原本的人力监控，就需要进一步改进当前的视频监控和分析技术

，通过视觉技术来分析视频的背景和目标，从而实现目标智能化监控。

1.2.2 发展趋势

视频监控支持移动场景。传统的视频监控摄像头大多是固定不动的，可以认为是固定在某一个静止的物体上，例如房屋的墙上、路灯杆上等等。在这种场景下，可以通过网络布线的方式解决监控中心与被监控点的通信。但是一旦遇上部署在移动物体上的监控摄像头，例如公交车、高铁等，传统的网络布线方式解决通信问题显然不可取，还有一种场景就是遇上了复杂地形，高山深谷、河流沙漠等等，同样有限网络也是束手无策。这个时候支持移动网络的监控摄像头就显得尤为重要，3G、4G、5G等移动网络技术的出现，使得视频监控支持移动场景变得更加简单。

视频监控支持 IPv6协议。众所周知，IPv4的地址已经耗尽，想在公网上继续增加新的 IPv4地址的网络监控摄像头有点困难，如果是在局域网中搭建的监控摄像头系统，那么就不会收到这个限制，但是局域网中的监控摄像头只能在局域网中查看（排除将监控视频上传的公网的情况）。因此需要尽快研发支持 IPv6 v6协议的摄像头。

视频监控支持家庭数字化。数字家庭网络可以由一个的统一的公共网管控制家中的所有智能家居，在视频监控普及的过程中，不可避免的要和智能家居这个概念联系起来。最终的结果就是视频监控也能支持家庭数字化的场景，能使用一个统一网关进行操作（观看实时录像、视频回放、抓拍等功能）。

第一章引言 3

1.3 本文主要工作

为了满足视频监控系统数字化、智能化、网络化、人性化发展的需求，本文基于 Spring Boot设计并实现了一个视频实时监控系统。本系统将传统的视频监控系统与网页结合，拥有链接安防摄像头、查看实时视频、视频保存和查看功能。

系统主要分为服务器端和网页客户端。服务器端主要使用 Spring Boot 编写实现，采用 HTTP协议与网页客户端交互，接口依照 RESTful风格编写，并使用Java-CV依赖对摄像头视频数据进行采集保存。网页客户端主要使用Vue.js框架编写，主要实现了用户登录、实时监控查看、监控回放查看、增减设备、个人中心、权限管理等功能。总的来说，本文的主要工作可以概括为一下几点：

1. 链接安防摄像头，实时查看视频监控。通过安防摄像头厂商提供的 SDK，代

码中引入并调用即可，可以对安防摄像头的视频数据进行读取。此外也可以直接使用网络摄像头自带的 RTMP或者 HTTP-FLV协议格式的视频流地址，实现对摄像头视频的查看和录制。

2. 视频按时间顺序保存并支持检索。对从安防摄像头上获取的视频数据进行编

码，保存MP4格式的视频文件到本地磁盘或者分布式存储系统中，以“安防摄像头唯一标识 +视频时间”作为视频文件的文件名，以达到按时间存储的目的，同时在借助数据库保存视频文件的相关信息，支持根据视频文件的开始时间和结束时间为搜索条件进行检索。

3. 用户登陆和权限管理。不同的用户对监控设备有不一样的可见权限，同时不

同的用户对系统的菜单也有不同的可见权限。本文基于 RBAC设计了一个权限系统。

4. 前端页面开发和后端服务开发。前端页面采用 Vue.js 框架进行编写，采用

Element-UI开源前端组件。后端采用 SpringBoot框架实现服务，采用前后端分离的方式实现，定义 RESTful接口交互，实现解耦。

1.4 本文结构

第一章介绍了本课题的研究目的和意义，简述了当前视频监控技术和视频监

控系统的研究现状和发展趋势，最后对课题的主要工作进行了概括。

4 基于 springboot的视频实时监控系统的设计与实现

第二章介绍了系统设计过程中涉及到的关键技术。首先对流媒体传输技术进行了详细介绍，紧接着对开发框架 Spring Boot、MapStruct和数据存储技术 Redis进行了简要叙述。最后介绍了 RABC权限系统的概念以及设计思路。

第三章介绍了系统的总体设计思路。

第四章介绍了系统的具体开发实现细节。

第二章关键技术研究 5

第二章关键技术研究

2.1 流媒体传输技术

实时的视频监控系统需要基于流媒体传输技术来实现，需要摄像头本身或者后置的图像处理服务器支持流媒体传输，常见的三种流媒体传输技术有：RTMP、HTTP-FLV和 HLS.

在对这三个协议个优势和劣势进行调研比较之后，决定采用 HTTP-FLV来实现摄像头实时视频的传输和播放。

2.1.1 流媒体以及流式传输

流媒体流媒体 （Streaming Media）是在网络上将压缩的多媒体数据流进行传输的技术，然后用户将压缩包进行解压后就能播放数据流 [3]。它能将一连串的媒体数据（包含音频、视频数据）压缩后并采用流式传输的方式发送，此技术使得数据包得以像流水一样发送；如果不使用此技术，就必须在使用前下载整个媒体文件。流媒体在播放前不会下载整个文件，只将开始部分存入内存，同时也会在用户访问时对数据包进行缓存，让媒体数据正确地输出，流媒体数据流随时传送随时播放。

流式传输流式传输是指客户端通过链接流媒体服务器实时传输音频、视频数据，实现“边下载边播放”。流式传输是实现流媒体的关键技术。流式传输时，音频、视频等时基媒体由音视频服务器向用户计算机的连续、实时传送，用户不必等到整个文件全部下载完毕，而只需经过几秒或十数秒的启动延时即可进行观看。当音频、视频等时基媒体在客户机上播放时，文件的剩余部分将在后台从服务器内继续下载。流式不仅使启动延时成十倍、百倍地缩短，而且不需要太大的缓存容量。流式传输避免了用户必须等待整个文件全部从网络上下载才能观看的缺点。

6 基于 springboot的视频实时监控系统的设计与实现

工作过程

图 2.1 展示了流媒体以及流式传输的工作过程。流媒体数据存储区存放着视

频、音频等多媒体数据，这些数据经过发送缓冲区，被编码器编码后发送给客户端。发送缓冲区和编码器对发送的数据流量进行负反馈调节，防止因为网络拥塞造成卡顿或者数据丢失。服务端和客户端之间通过流媒体传输协议交互。客户端收到服务端发送的数据之后，进行解码然后在播放器中播放。

流媒体数据存储区域发送缓冲区编码器反馈调节数据流量服务端解码器播放器解码器播放器解码器播放器客户端客户端客户端

图 2.1 流式传输示意图

第二章关键技术研究 7

2.1.2 基于 RTMP的流媒体传输技术

实时消息传输协议（即 Real-Time Messaging Protocol，缩写 RTMP）最初是Macromedia 公司为了满足在互联网上传输流媒体音视频而开发的一个私有协议。

RTMP是工作在 TCP之上的明文协议，默认使用端口 1935。

RTMP 协议为了维持稳定连续传递，避免单次传输数据量问题，采用了传输层封包，数据流切片的实现形式。被用来对当前带宽进行划分和复用的最小传输单位，被称为 Chunk即消息块 [4]。

一个完整的数据块包含两个部分：Chunk Header和 Chunk Data，这两者组合在一起，构成了一个有效的消息类型，结构如图 2.2 所示。Chunk Header 由基础数据头（Basic Header）、消息数据头（Message Header）和扩展时间戳（ExtendedTimestamp)构成.

+--------------+----------+-------------------+-------------+| Basic Header | Message Header | Extended Timestamp | Chunk Data |+--+|||<------------------ Chunk Header ----------------->|

图 2.2 Chunk格式示意图

通常情况下，一个有效的消息，如果数据量超出当前 Chunk Size的话，则会被拆分成多个分块来分批传输。默认情况下，Chunk Size的值为 128字节。

图 2.3展示了一则当 Chunk Size为 128字节是的消息拆分发送过程。协议中

的基本数据单元称为消息（Message），传输的过程中消息会被拆分为更小的消息块（Chunk）单元，最后将分割后的消息块通过 TCP协议传输，接收端再反解接收的消息块恢复成流媒体数据。一个大小为 500字节的 RTMP消息，被拆分为 4个大小分别为 128字节、128字节、128字节、116字节的消息块，最终通过 TCP发送出去。

8 基于 springboot的视频实时监控系统的设计与实现

RTMP Message (500 Bytes)Chunk Data (128 Bytes)Chunk HeaderChunk Data (128 Bytes)Chunk HeaderChunk Data (128 Bytes)Chunk HeaderChunk Data (116 Bytes)Chunk HeaderChunk Size = 128TCP

图 2.3 消息拆分示意图

RTMP主要有以下几个优点：

1. RTMP是专门为流媒体开发的协议，对底层的优化比其它协议更加优秀。

2. 基本上所有的编码器（摄像头之类）都支持 RTMP输出，。

3. RTMP适合长时间播放，并且具有较低的延时，一般延时在 1-3s之间。

RTMP并不是完美的，也有不足之处。一方面是它是基于 TCP传输，没有像HTTP那样的公共端口，被防火墙拦截的可能性也更大；另一方面是 Adobe并没有完全开源 RTMP协议，这就导致很多设备无法需要使用第三方解码器才能播放。

第二章关键技术研究 9

2.1.3 基于 HTTP-FLV的流媒体传输技术

HTTP-FLV协议是将流媒体数据封装成 FLV格式，然后在通过 HTTP协议进行传输。

Flash Video（简称 FLV），是一种网络视频格式，用作流媒体格式，它的出现有效地解决了视频文件导入 Flash后，使导出的 SWF文件体积庞大，不能在网络上有效使用等缺点 [5]。

FLV文件格式由 FLV Header和 FlV Body组成。FLV Body由多个 Tag构成，每个 Tag包含 Tag Header和 Tag Data组成。FLV协议的格式如图 2.4所示。

FLV Header (9 Bytes) FLV BodyFLV Tag FLV Tag FLV Tag .....

Tag Header (11 Bytes) Tag DataPrevious Tag Size (4 Bytes) Tag Header Data (7 Bytes)

图 2.4 FLV协议格式

FLV Tag共有三种类型，分别是 Video Tag、Audio Tag和 Script Tag，通过 TagHeader Data中的第一个字节来区分。Video Tag后的 Tag Data中存放的是视频相关数据；Audio Tag后的 Tag Data中存放的是音频相关数据；Script Tag后的 Tag Data中存放的是音视频元数据。

FLV Tag中的前四个字节用来表示前一个 Tag的长度，即图 2.4中的 PreviousTag Size字段。因此，整个 FLV Body部分可以通过这个字段完全串联起来，形成一个完整的文件，如图 2.5所示。

HTTP-FLV是基于 HTTP进行数据传输的，可以使用 HTTP协议的公共端口80，因此可以有效避免出现被防火墙拦截的情况。此外，还可以通过 Nginx等软件实现负载均衡，减轻服务端的压力，进行流量的灵活调度。对于需要数据加密的场景，可以通过 HTTPS实现。同时它还能很好的支持 HTML5，只需要使用 <video>10 基于 springboot的视频实时监控系统的设计与实现FLV HeaderPrevious Tag Size 0Tag 1Previous Tag Size 1Tag 2Previous Tag Size N-1Tag N.

.

.

.

图 2.5 FLV文件组成结构

标签就可以实现在网页中播放 HTTP-FLV视频流。

但是由于它的传输特性，浏览器会缓存部分的流媒体资源在本地客户端，在保密性方面存在问题。

2.1.4 基于 HLS的流媒体传输技术

HLS协议是由苹果公司在 2009年提出的基于 HTTP的流媒体传输协议，主要应用于 PC端和苹果终端，可以提供近似实时的流媒体服务，苹果公司在 i Phone 3.0中首次尝试了该技术 [6]。

HLS协议的原理是将输入的是流切分成若干个小的文件，然后通过一个索引文件将它们串联起来。播放器需要播放时，会先请求索引文件，再根据索引文件从文件服务器上获对应的音视频等多媒体文件。其工作原理如图 2.6所示。

在图 2.6 中，音视频输入设备产生的数据经过 Media encoder 编码后发送到Stream segmenter进行切分操作。完成切分操作之后，会产生索引文件和对应的多媒体文件，即图中的 index file和。ts文件。之后客户端可以通过 HTTP请求的方式获取索引文件和多媒体文件进行播放。

HLS也是基于 HTTP进行数据传输的，所以同样可以避免被防火墙拦截，支

第二章关键技术研究 11

图 2.6 HLS协议工作原理

持负载均衡。但是由于它需要对输入流进行切分操作，所以会产生较大的延迟，同时也会产生海量的小文件，对存储和缓存由一定的要求。

2.1.5 RTMP、HTTP-FLV、HLS简单对比

表 2.1对 RTMP、HTTP-FLV、HLS这三种协议在底层传输协议、延迟、使用

场景等方面进行了对比。

表 2.1 RTMP、HTTP-FLV、HLS对比 [7]

协议底层传输协议视频封装格式延迟数据分段 HTML5RTMP TCP flv tag 2秒连续流不支持HTTP-FLV HTTP flv 2秒连续流支持HLS HTTP m3u8 10秒以上切片支持

RTMP是为流媒体设计的，许多设备使用的推流协议都是 RTMP，应用范围比较广，同时拥有较低的延时。但是相比于其他两个个协议，它并不 HTML5。

HTTP-FLV基于 HTTP的长连接特点进行数据的传输，支持 HTML5，具有较低的延时。

HLS是苹果公司提出的直播协议，在苹果的生态系统占据着首要地位，不需要安装任何应用就可以实现播放。但是与其他两个相比，最致命的缺点就是具有很高的延时。

12 基于 springboot的视频实时监控系统的设计与实现

2.2 Spring Boot开发框架

2.2.1 Spring Boot开发框架简介

SpringBoot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

Spring Boot不仅继承了 Spring框架原有的优秀特性，而且还通过简化配置来进一步简化了 Spring应用的整个搭建和开发过程。在 Spring Boot出现之前，使用 Spring框架进行应用开发需要编写繁琐的配置文件，然而这些配置文件的内容大多都是类似的。

如图 2.7所示，Spring框架约由 20个模块组成，它们被分成 Core Container（核心容器）、Data Access/Integration（数据访问集成）、Web（网页）、AOP（面向切面编程）、Instrumentation（加载引入）、Messaging（消息）和 Test（测试）七个大模块。

图 2.7 Spring框架模块结构

1. Core Container模块是 Spring框架的基础，提供了控制反转（IoC）和依赖注入

（DI）的能力。其内部的 BeanFactory帮助开发者消除了手动创建单例的过程，将单例的创建和组装从业务代码中解耦。开发者可以通过 ApplicationContext

第二章关键技术研究 13

提供的接口访问容器内的创建的对象。容器同时还提供了表达式计算、第三方框架集成的能力。

2. AOP and Instrumentation模块提供了提供了面向切面编程以及第三方模块集

成的能力。开发者可以通过面向切面编程可以通过方法拦截器或切点将非业务代码从业务代码中解耦。

3. Messaging模块提供了使用消息队列的统一接口。

4. Data Access/Integration模块由 JDBC、ORM、OXM、JMS和 Transaction构成。

提供了屏蔽数据库供应商的数据库访问接口，并且支持事物。对象映射支持ORM、OXM等映射方式。同时还封装了面向消息队列的接口，支持对消息的生产和消费。

5. Web模块提供了面向Web开发的基本功能的集成，例如基于其的 Servlet加

载、文件上传等功能。Web开发能力。可以借助该模块设计并实现一个基于MVC设计模式的Web应用程序。

6. Test模块提供了单元测试和集成测试的能力，使得普通的单元测试和集成测

试也能使用 Spring的容器。

2.2.2 控制反转和依赖注入特性简介

Spring 框架具有控制反转（IOC）特性，它通过对象元数据配置文件，借助Java反射机制提供的能力，将程序生命周期周内的对象的创建和使用统一收口到Spring的容器中，由这个容器进行管理。

控制反转在大多数时候都是通过依赖注入来实现的。当对象 A持有了类 B的一个实例，就可以说对象 A依赖对象 B。依赖注入要求对象与对象之间的依赖仅仅通过构造函

数、工厂方法和属性写入方法来定义。

Spring容器就可以将通过这种方式声明的被依赖对象从容器中找到并且通过构造函数、工厂方法和属性写入方法来注入到需要它的对象内。因为这个被依赖对象的创建过程是由容器完成的并在需要它的时候才进行注入，而不是被需要它对象的创建的，所以这个过程就体现了控制反转和依赖注入。

14 基于 springboot的视频实时监控系统的设计与实现Spring 容器完备系统产生POJOs对象元数据配置文件<？xml version=“1.0" encoding=“UTF-8"？><beans xmlns="xmlns">.....

</beans>

图 2.8 IoC和 DI工作过程

在 Spring框架的定义中，控制反转又被称为依赖注入。如图 2.8所示，Spring框架会通过对象元数据配置文件和开发者编写的 POJOs（Plain Old Java Objects）来构造 Spring容器，进而启动一个完备的系统。

2.2.3 面向切面编程简介

面向切面编程（AOP）可以理解是面向对象编程（OOP）的扩展。面向切面编程引入，使得面向对象编程具有更高扩展性和更低的耦合性。面向对象编程中最重要的概念就是对象（Class），而在面向切面编程中，它变成了切面（Aspect）。

在 Spring框架中，AOP是最重要的组成部分之一。在 AOP中，有一下几个重要的概念：

1. 切入点（Join point）：程序执行过程中的一个时间点，例如方法执行、异常处

理、变量赋值等等。在 Spring AOP中，只能是方法执行。

2. 增强（Advice）：在某个特定的切入点上执行的一段额外逻辑。可以是在切入点

之前、之后或者环绕执行。在多数的AOP框架中，是通过拦截器（Interceptor）来实现的，并维护者一个拦截器链。

3. 切点（Pointcut）：是一个谓词表达式，用来计算是否匹配这个切入点。对于

方法执行的切入点，切点可以是“当参数为某一指定值”。

4. 切面（Apsect）：切面是切点和增强的组合。

第二章关键技术研究 15

在程序执行的过程中，有若干个切入点，每个切入点都可以成为 Aspect的目标。

AOP的最终目标就是在程序运行的某一个时间点，可以执行一段额外的逻辑。

图 2.9简单描述了 AOP的原理和执行过程。在图 2.9中，切面（Aspect）对应的切

入点（Join point）为 Join point N，切点（Pointcut）的谓词表达式为 pointcut ==Join point N ，而增强（Advice）是在 Code N 执行之前输出 before，执行之后输出 after.

Code N-2Code N-1Code NCode N+1Code N+2程序执行.

.

.

.

.

.

Join point N-2Join point N-1Join point NJoin point N+1Join point N+2Join point N+3pointcut == Joint point NBefore:Print(before)Before:Print(after)AspectPointcutAdvice

图 2.9 AOP原理示意图

16 基于 springboot的视频实时监控系统的设计与实现

第三章系统需求分析与总体设计 17

第三章系统需求分析与总体设计

3.1 需求分析

需求分析也称为软件需求分析、系统需求分析或需求分析工程等，是开发人员经过深入细致的调研和分析，准确理解用户和项目的功能、性能、可靠性等具体要求，将用户非形式的需求表述转化为完整的需求定义，从而确定系统必须做什么的过程。[8]本节将从功能性需求和非功能性需求两个角度进行阐述，前者确定系统需要的功能，后者确定系统的扩展性和兼容性。

3.1.1 功能性需求

本系统的主要目标是设计并实现一个基于 Spring Boot 的视频实时监控系统，其核心功能是使用户能够通过登陆访问本系统，并在可见权限范围内进行实时监控视频查看和回放视频查看。在此基础上，系统引入了一个基于 RBAC设计的权限系统，用户可以拥有不同的角色，每个角色可以拥有不同的权限，这些权限可以包括功能权限、菜单权限等等。

18 基于 springboot的视频实时监控系统的设计与实现

图 3.1是系统用户用例图，体现了系统的大致功能。

用户登录设备管理实时监控录像回放个人中心用户管理系统设置角色管理权限管理<<包含>><<包含>><<包含>>视频监控验证码记住密码修改密码修改头像修改昵称<<包含>><<包含>><<包含>><<包含>><<包含>>文件列表视频录制设备列表增加设备修改设备删除设备<<扩展>><<扩展>><<包含>><<包含>><<包含>><<包含>><<包含

>><<包含>><<包含>><<包含>><<包含>>

图 3.1 系统用户用例图

用户登陆在本系统中，用户必须完成登陆才能继续使用，用户能看到的设备列表信息和视频文件信息都是在当前登陆用户的可见范围内的数据。在登陆时，用户需要正确输入用户名、密码和图形验证码才能顺利登陆。如果用户选择了记住密码选

项，则下次登陆时就不需要再次输入用户名和密码。

此部分的用例描述如表 3.1所示。

表 3.1 用户登录用例描述

用例名称用户登录用例标识号 Login参与者用户前置条件无基本事件流

1. 用户打开系统登陆页面

2. 用户输入用户名、密码和图形验证码

3. 用户点击登陆按钮

4. 提示登陆成功，跳转到系统主界面

其他事件流在点击登陆按钮之前，用户名、密码、验证码输入框都不能为空异常事件流

1. 提示错误信息

2. 更新验证码

后置条件系统进入主界面，选择记住密码后用户名和密码保存到本地设备管理设备管理部分会展示设备列表，其中包括了当前登陆用户可见的所有设备。用户可以根据设备名称和设备类型进行搜索筛选，支持分页查询。用户可以进行设备的增加、修改和删除操作。被删除的设备不会从列表中消失，仅仅只是更改“删除状态”。

此部分的用例描述如表 3.2所示。

表 3.2 设备管理用例描述

用例名称设备管理用例标识号 Device参与者用户前置条件用户已经登陆，拥有相关菜单权限和数据权限基本事件流

1. 用户进入设备列表页面，可以翻页查看、条件搜索。

2. 用户更改设备删除状态，对设备进行删除和恢复

3. 用户点击添加按钮，填写设备相关信息后提交完成添加

4. 用户点击修改按钮，弹出对话框，更改信息后提交完成修改

其他事件流在添加设备、修改设备时，用户填写的数据必须符合要求异常事件流

1. 提示信息填错错误

2. 用户重新填写后再次提交

后置条件无视频监控视频监控包含录像回放和实时监控。用户可以在录像回放下查看所有未被删除和被删除的设备的历史监控视频文件，可以在线播放和下载。实时监控可以查看未被删除的设备的实时监控视频。此部分的用例描述如表 3.3所示。

表 3.3 视频监控用例描述

用例名称视频监控用例标识号 Video参与者用户前置条件用户已经登陆，拥有相关菜单权限和数据权限基本事件流

1. 用户选择录像回放功能，展示文件列表，支持搜索和分页查询

2. 在文件列表中可以选择任意文件进行在线播放和下载

3. 用户选择实时监控功能，展示设备列表

4. 在设备列表中可以选择未被删除的设备进行实时监控视频的查看

5. 后台持续进行设备的视频录制

其他事件流无异常事件流无后置条件无个人中心个人中心只要包括用户信息的展示和修改。只有用户的头像、密码和昵称支持修改。

此部分的用例描述如表 3.4所示。

表 3.4 个人中心用例描述

用例名称个人中心用例标识号 Info参与者用户前置条件用户已经登陆，拥有个人中心的菜单权限和数据权限基本事件流

1. 用户选择图片进行上传，更改头像

2. 用户修改用户昵称和密码

3. 点击提交按钮完成修改

其他事件流用户头像、昵称和密码的修改需要符合规范异常事件流

1. 提交无效的数据，提示相关错误信息

2. 用户修正后重新提交

后置条件无系统设置系统设置主要是对权限相关的部分进行操作。具有这部分权限的用户可以对系统用户进行增删改查，增加系统角色系统权限，对用户和角色、角色

和权限进行关联。

此部分的用例描述如表 3.4所示。

22 基于 springboot的视频实时监控系统的设计与实现

表 3.5 系统设置用例描述

用例名称系统设置用例标识号 Syetem参与者用户前置条件用户已经登陆，拥有系统设置的菜单权限和数据权限基本事件流

1. 用户打开系统设置界面

2. 在用户列表可以对用户进行增删改查

3. 在角色列表可以进行角色的增删改查

4. 在权限列表可以进行权限的增删改查

5. 在角色列表选择角色之后，可以关联权限到该角色上

6. 在用户列表选择用户之后，可以关联角色到该用户上

其他事件流在进行关联、修改数据等操作时，需要符合数据规范异常事件流

1. 提交数据不符合规范，提示错误信息

2. 用户修正信息后重新提交

后置条件无

3.1.2 非功能性需求

兼容性系统基于设备的 HTTP-FLV直播流进行实时监控查看和历史视频查看，屏蔽了不同厂商设备底层的差异，只要设备能提供 HTTP-FLV地址，就可以在本系统中添加使用。

数据完整性在进行监控视频录制时，会冗余部分视频数据，这样在某一段时间的视频文件丢失时，可以在这段时间之前和之后的视频文件中找到。

第三章系统需求分析与总体设计 23

3.2 系统数据库设计

本系统借助MySQL数据库进行数据的存储和管理。MySQL是关系型数据库，需要先通过 E-R图来展示系统之间各个实体的属性和它们之间的联系。然后在借助 E-R图，完成数据表的设计。系统的实体关系图如图 3.2所示，因为篇幅限制，只展示了实体的部分核心属性。

图 3.2 实体关系图

根据图 3.2，可以推导出数据库中数据表的结构，下面列出了数据库中部分关键的数据表。

3.3 系统架构设计

本系统架构设计分为五个模块，如表 3.6和图 3.3所示。本节将对每个模块的功能做详细介绍。

表 3.6 系统模块表

模块名称模块功能公共模块提供公共工具的能力数据存储模块提供持久组件的读写的能力业务逻辑模块系统的主要业务逻辑视频文件存取模块离线视频文件的存取前端接口交互模块为前端提供接口交互24 基于 springboot的视频实时监控系统的设计与实现前端Web业务逻辑模块业务接口Interface业务实现Implementation数据存储MySQL Redis OSS公共模块前端接口交互模块控制器Controller视频文件存取模块数据存储模块对象实体Entity访问接口Mapper对象转换Converter视频录制Record视频存取Save本地文件系统

图 3.3 软件总体框架图

3.3.1 公共模块

公共模块（monitoring-system-common）提供公共工具的能力。模块内主要包括开发过程中需要用到的注解（annotation）、常量（constant）、枚举（enumerate）、异常（exception）以及公共的工具封装（utility）。

公共模块中封装的工具包括时间转换工具类、HTTP请求工具类、Spring工具类以及登陆验证码工具类。

时间转换工具类提供将 java.util.Date 对象转为具有一定格式的字符串（例如“yyyy-MM-dd HH:mm:ss”），同时也提供逆向转换的能力，即将字符串格式的时间转化为 java.util.Date对象。

HTTP请求工具类封装了 HTTP请求中相关的 GET、POST两种请求方法，借助第三方依赖 OkHttp二次开发实现。支持带参数的 GET请求、带请求体参数的POST请求.

Spring 工具类提供了对 Spring 上下文的 ApplicationContext 的静态访问，可以在全局状态下任何一个地方访问上下文，即可以访问容器内管理的对象和使用Spring提供的观察者模式。

登陆验证码工具类主要提供在登录时生产验证码图片，如下图所示。

第三章系统需求分析与总体设计 25

图 3.4 登陆验证码

3.3.2 数据存储模块

数据存储模块（monitoring-system-repo）提供持久组件的读写的能力。持久组件就是提供数据存储的服务，在本系统的设计中，持久组件包括 MySQL数据库、Reids内存数据库、本地文件系统。

系统设计使用 Mybatis 框架对数据库进行访问，因此该模块中的大部分代码有时有 Mybatis-Generator自动生成，包括实体对象类、Mapper接口和 Mapper配置文件。除此之外，该模块还包括了各种实体的封装类，比如前端参数封装类（用于封装前端传递给后端的参数）、HTTP响应类（封装后端接口返回对象）以及各个模块使用

的实体类（VO、BO）。

该模块还负责对象之间的转换，即将VO转换成 BO或者将 BO转换成 BO等。

该功能是通过对象拷贝技术MapStruct生成的对象转换代码实现的。

3.3.3 业务逻辑模块

业务逻辑模块（monitoring-system-service）提供业务逻辑处理的功能。该模块是本系统最核心的一个模块，主要负责业务模块的逻辑应用设计，支持登陆鉴权、设备信息的增删改查、个人信息修改等。

业务逻辑模块的主要设计思路是面向接口编程。接口主要用于描述类具有什么功能，而并不给出每个功能的具体实现。一个类可以实现一个或多个接口，并在需要接口的地方，随时使用实现了相应接口的对象。

首先设计接口，再设计其对应的实现的类，接着利用 Spring的依赖注入特性，就可以在上层模块声明接口类型的变量，而无需关心其具体实现。换句话说，业务逻辑模块的变动不会对上层模块的调用产生任何影响。

这就是软件六大设计原则之一的依赖倒置原则，即面向接口编程。上层模块不应该依赖底层模块，它们都应该依赖于抽象。抽象不应该依赖于细节，细节应该依赖于抽象。上层模块不应依赖底层模块，即上层的业务模块不应该依赖底层的实现模块。如：人出行使用交通工具，只需要知道是交通工具就可以，不用知道26 基于springboot的视频实时监控系统的设计与实现是哪种具体的交通工具。抽象不应该依赖于细节，细节应该依赖于抽象。在具体的 Java编程中，抽象指代接口、抽象函数，统称为接口，而细节指代接口的具体实现。

3.3.4 视频文件存取模块

视频文件存取模块（monitoring-system-media）提供实时监控视频录制和离线查看检索功能。该模块提供了两个HTTP的接口，分别是监控回放文件的列表查询接口和监控回放文件下载接口。对于已经在本系统中绑定的监控摄像头设备，每隔一分钟视频文件存取模块会执行定时任务，对该设备进行录制，并且录制的时长为 80秒，目的是做到前后视频文件能有重合部分，防止视频数据丢失。

对于视频文件的存放，可以在本地文件系统和云存储之间无缝切换。云存储采用阿里云的 OSS对象存储服务或本地搭建的 Hadoop分布式存储系统。对于视频文件的读取，该模块提供了一个 GET方法的 HTTP接口，入参为视频文件的名称，返回值为该文件内容。同时还提供搜索功能，支持按照监控视频的开始时间和结束时间进行范围查找。

3.3.5 前端接口交互模块

前端接口交互模块（monitoring-system-web）提供与前端交互的能力。该模块负责接受前端的请求，然后调用下层模块（如视频文件存取模块）提供的接口方法，实现业务逻辑，然后将下层模块的返回值进行一定程度的封装后返回给前端进行展示。

第四章系统的详细设计与实现 27

第四章系统的详细设计与实现

4.1 开发环境搭建

4.1.1 前端开发环境搭建

前端使用 Vue.js 框架进行开发，通过 yarn 和 npm 进行项目依赖管理。开发IDE使用 Visual Code.

4.1.2 后端开发环境搭建

前端使用 Spring Boot 框架进行开发，通过 Maven 进行项目依赖管理。开发IDE使用 IDEA.

4.2 前端界面设计与实现

本系统采用前后端分离的开发模式，因此前端页面与后端服务是完全独立的。

本节将介绍前端界面的设计，主要采用 Element-UI开源组件设计。

4.2.1 登陆界面

登陆界面如图 4.1所示，分为三个主要部分：信息输入、图形验证码和按钮。

信息输入部分有三个输入框，分别是用户名输入框、密码输入框、图形验证码输入框和“记住密码”选择框。图形验证码部分用于展示图形验证码。登陆按钮用于向服务器提交用户名、密码和图形验证码。登陆成功之后会跳转到基础管理的设备管理菜单。

28 基于 springboot的视频实时监控系统的设计与实现

图 4.1 登陆页面

4.2.2 实时监控和监控回放播放

实时监控的播放和监控回放的播放都是通过 Bilibili的开源组件 flv.js实现的。

数据库中记录这每一台设备的 HTTP-FLV视频流地址，前端在获取到这个地址后，通过 flv.js组件可以在 HTML页面中实时播放这个链接，如图 4.2所示。

系统会对实时的监控视频进行录制，为了保证数据的完整性，录制视频的有效时长为 1分钟，多余的部分为冗余数据。可以在回放文件的列表中对视频进行在线的播放或者下载到本地，如图 4.3所示。

第四章系统的详细设计与实现 29

图 4.2 实时监控播放示例

图 4.3 监控回放播放示例

4.2.3 页面布局

页面布局分为三个部分：顶部导航栏、左侧菜单栏和主窗口，如图 4.4所示。

图 4.4 页面布局

顶部导航栏顶部导航栏使用 Element-UI中的 NavMenu导航菜单组件开发，用于展示用户名和头像，可以通过点击用户名或者头像打开下拉菜单，如图 4.5所示。

30 基于 springboot的视频实时监控系统的设计与实现

图 4.5 顶部导航栏

用户单击个人中心可以跳转到个人中心页面；用户单击退出会弹出二次确认对话框，选择是后会退出登陆并跳转到登陆页面。

左侧菜单栏系统的总菜单布局在页面的左侧，分为三个一级菜单，每个一级菜单下又有二级菜单，如图 4.6所示。

（a）基础管理菜单 （b）设备中心菜单 （c）系统管理菜单

图 4.6 左侧菜单

基础管理菜单下有设备管理菜单和个人中心菜单。设备中心菜单下有实时监控菜单和监控回放菜单。系统管理菜单下有用户管理菜单、角色管理菜单和权限管理菜单。

左侧菜单会因为每个用户拥有不同的菜单权限而有不同的展示。单击对应的菜单可以跳转到对应的页面。

第四章系统的详细设计与实现 31

主窗口主窗口会根据不同的二级菜单而展示不同的内容，主要用于承载信息、完成操作。

4.2.4 二级菜单页面

各个二级菜单的页面结构大致相同，如图4.7 所示，主要分为四部分：导航、搜索、列表、分页。

图 4.7 二级菜单页面结构

导航导航部分使用 Element-UI的 Breadcrumb面包屑组件开发，可以现实从首页到当前页面的路径。例如，当选择基础管理菜单下的二级菜单设备管理，这部分就会现实“首页 >设备管理”，如图 4.8。

图 4.8 一个导航实例

32 基于 springboot的视频实时监控系统的设计与实现搜索搜索栏用于对列表展示的内容进行搜索，可以通过指定搜索条件来展示不同的列表内容。每一个二级菜单都有不同的搜索内容和选项。

例如在设备管理页面，搜索部分的内容如图 4.9所示，而在监控回放的页面，就如图 4.10所示。

图 4.9 设备管理页面的搜索部分

图 4.10 监控回放页面的搜索部分

列表列表页面展示数据，有后端接口返回。类似的，不同的页面列表部分展示的数据也不尽相同。该部分展示的内容会受到搜索部分和分页部分影响。

分页当数据过多时，后端会采用分页的方式返回数据，这一部分就是为了配合后端实现分页功能。用户可以通过这个部分更改列表展示的数据条数和数据页数。

4.3 后端服务设计与实现

本节将介绍后端服务的设计，主要采用 Spring Boot框架设计开发，通过REST-ful形式的 HTTP接口与前端交互。

第四章系统的详细设计与实现 33

4.3.1 图形验证码接口设计与实现

图形验证码的作用是过滤无效的请求非认为请求，防止爬虫。图形验证码接口时序图如图 4.11所示。

前端页面图形验证码接口公共模块用户打开页面请求 /api/auth/code 接口服务端 Session调用公共模块工具类返回 base64 格式的图片字符串将图片对应的图形验证结果存入服务端 Sesion返回返回 base64 格式的图片字符串展示图形验证码

图 4.11 图形验证码接口时序图

当用户打开本系统时，如果没有登陆过，则会跳转到登陆页面，之后就会请求图形验证码接口 /api/auth/code。接口内部会调用公共模块的 VerifyUtil类中的相关方法生产图形验证码，并返回二进制图片，Base64编码格式的图片字符串和图形验证码。图形验证码接口拿到公共模块的返回值后，会将图形验证码存入 Session，并将 Base64 编码格式的图形验证码返回给前端界面，最后前端展示图片，如图

4.12所示。

图 4.12 图形验证码

4.3.2 登陆接口设计与实现

用户在登陆页面输入用户名、密码和图形验证码，点击登陆按钮之后会调用后端接口。接口地址为 /api/auth/login，接受的请求方法为 POST，需要三个参数：

34 基于 springboot的视频实时监控系统的设计与实现username（用户名）、password（密码）和 code（图形验证码）。后端再对这着三个参数进行校验，如果通过则将用户信息和 token（唯一标识符）返回。完整的时序图如 4.13所示。

登录接口服务端 Session用户输入用户名、密码和图形验证码从 Sesion 中查询图形验证码数据库返回返回用户信息返回用户信息和 tokenRedis校验图形验证码根据用户名在数据库中查询校验密码生成 token存储 token 并设置过期时间返回

图 4.13 登陆接口时序图

4.3.3 列表查询接口设计与实现

列表页接口采用分页查询实现。由于Mybatis-generator自动生成的Mapper并不支持分页查询的功能，因此通过使用扩展插件的方法，在生成的代码中增加分页参数limit和 offset。该接口与前端的分页组件联合使用，可以实现分页查询列表的功能。

此接口的实现并不复杂，因此不再赘述。

4.3.4 监控视频保存的设计与实现

实时视频监控是通过在前端使用 flv.js组件播放设备的HTTP-FLV直播流实现的。本系统通过使用 Java-CV依赖对设备的 HTTP-FLV直播流进行抓流并保存为MP4文件来实现监控视频的保存。

其中每一段视频的长度定为 1分钟，同时为了使用户更方便的查找视频文件，

第四章系统的详细设计与实现 35

每一个视频的命名格式为“yyyy-dd-MM-HH-mm-ss.mp4”，并记录每个视频文件的录制开始时间和结束时间。本系统通过定时任务，每隔一分钟遍历全部的摄像头，为每个摄像头创建一个一分钟的录制任务，并通过线程池来执行。此外，在摄像头设备被添加到系统中时，也会同步启动一次视频录制。

监控视频保存的流程图如图 4.14所示。

设备管理视频录制服务用户数据库返回启动视频录制线程池查询全部设备创建录制任务保存录制结果添加设备提交执行循环每分钟定时执行

图 4.14 监控视频保存流程图

4.3.5 文件上传下载设计实现

文件上传文件上传接口用于个人中心修改头像功能。该接口利用 Spring Boot的 Multi-partFile实现，接口地址为/api/file/upload。MultipartFile的内容是文件的二进制数据和文件名。在接口收到数据之后，可以通过MultipartFile获取到文件的二进制数据和文件名，之后调用 Java访问文件系统的相关方法，将该文件保存在本地的文件系统中。最终返回一个文件的下载地址。

具体的流程图如图 4.15所示。

36 基于 springboot的视频实时监控系统的设计与实现开始前端调用文件上传接口后端接收MultipartFile获取二进制数据和文件名调用Java相关方法保存到本地返回下载地址结束

图 4.15 文件上传流程图

文件下载文件下载接口用于个人头像的现实和视频监控回放文件的播放和下载功能中。

其接口实现较为简单，首先前端根据参数传递需要下载的文件的文件名，后端通过 Java 访问文件系统打开对应文件获取到二进制的文件数据，然后将其写入HttpServletResponse中，并设置 HTTP返回值的 Content-Type、Character-Encoding，添加 Content-Disposition的响应头。

具体的流程图如图 4.16所示。

第四章系统的详细设计与实现 37

开始前端调用下载接口后端根据文件名获取二进制数据写入数据到HttpServletResponse中设置 Content-Type为 application/octet-stream设置 Character-Encoding为 utf-8设置 Content-Disposition结束

图 4.16 文件下载流程图

38 基于 springboot的视频实时监控系统的设计与实现

第五章系统测试与结果分析 39

第五章系统测试与结果分析

本章的主要任务是对系统进行完成的测试，目的是检验系统设计是否达到需求分析的目标，各个功能是否正常使用。

5.1 系统测试环境

本系统有摄像头、服务端和客户端三部分组成。

摄像头采用奥尼生产的 USB摄像头，分辨率为 1080P，如表 5.1所示。

表 5.1 测试摄像头配置表

品牌奥尼分辨率 1920*1080最大帧率 30 FPS编码格式 MJPG适应接口 USB2.0服务端和客户端部署在同一台设备上，设备的配置如表 5.2所示。

表 5.2 测试服务器配置表

操作系统 Windows 10处理器 Intel Core i5-7400内存 8GRedis 6.2.1MySQL 8.0.1340 基于 springboot的视频实时监控系统的设计与实现

5.2 测试结果及分析

测试方式采用黑盒测试，用于检验系统的功能是否正常。本节只介绍了关键功能的测试结果。

5.2.1 登陆功能测试

用户首次打开系统时，会进入登陆页面，要求输入用户名、密码和图形验证码。对于三个输入参数，只要存在一个不匹配，就会提示错误信息。表 3.1列出了部分测试用例以及测试结果。

表 5.3 登陆功能测试用例以及结果

序号用例描述测试输入预期结果结果

1 界面测试 1. 打开页面 1. 页面正常展示通过

2 正常登陆

1. 输入正确的用户名密码

通过

2. 输入正确的密码提示登陆成功

3. 输入正确的图形验证码并跳转到首页

4. 点击登陆按钮

3 用户名异常

1. 不输入用户名提示用户名

通过

2. 输入不存在的用户名或密码错误

4 密码异常

1. 不输入密码提示用户名

通过

2. 输入错误的密码或密码错误

5 验证码错误

1. 不输入验证码

提示验证码错误通过

2. 输入错误的验证码

6 记住密码

1. 输入正确的用户名、密码

通过

2. 输入正确的图形验证码自动填充用户名

3. 勾选记住密码和密码

4. 登陆后退出

第五章系统测试与结果分析 41

5.2.2 设备列表功能测试

用户可以在设备列表中进行以下操作：

1. 根据设备名称进行模糊搜索

2. 根据设备类型进行精确搜索

3. 添加设备

4. 修改设备信息

5. 更改设备删除状态

表 5.4列出了部分测试用例以及测试结果。

表 5.4 设备列表功能测试用例以及结果

序号用例描述测试输入预期结果结果

1 界面测试 1. 打开页面页面正常展示通过

2设备名称 1. 不输入设备名称列表展示符合通过搜索 2.输入设备名称条件的结果3设备类型 1. 选择全部列表展示符合通过搜索 2.选择除全部外的选项条件的结果

4 添加设备

1. 输入设备名称

提示添加成功通过

2. 选择设备类型

3. 输入设备地址

5 修改设备

1. 修改设备名称

提示修改成功通过

2. 修改设备类型

3. 修改设备地址

4. 点击保存按钮

6修改删除 1. 修改为删除设备删除状态变更通过状态 2.修改为不删除42 基于 springboot的视频实时监控系统的设计与实现

5.2.3 实时监控功能测试结果

用户可以在“设备中心 >实时监控”菜单下查看实时监控视频。对于已经被删除的设备，不支持实时监控查看。支持按照设备名称模糊搜索和按照设备类型精确搜索。

表 5.5列出了部分测试用例以及测试结果。

表 5.5 实时监控功能测试用例以及结果

序号用例描述测试输入预期结果结果

1 界面测试 1. 打开页面页面正常展示通过

2设备名称 1. 不输入设备名称列表展示符合通过搜索 2.输入设备名称条件的结果3设备类型 1. 选择全部列表展示符合通过搜索 2.选择除全部外的选项条件的结果4实时监

控

1. 点击查看实时监控视频按钮

播放实时监控通过播放

5.2.4 视频保存和检索功能测试结果

用户可以在“设备中心>监控回放>回放文件”菜单下查看实时历史的监控视频。对于已经被删除的设备，同样也支持查看历史的监控视频。用户可以在列表也下选中文件直接进行播放或者下载。支持按照视频的起止时间进行范围搜索。

表 5.6列出了部分测试用例以及测试结果。

表 5.6 视频保存和检索测试用例以及结果

序号用例描述测试输入预期结果结果

1 界面测试 1. 打开页面页面正常展示通过

2 搜索

1. 选择开始和结束时间列表展示符合

通过

2.不选择开始和结束时间条件的结果

3 播放文件 1. 点击播放按钮文件开始播放通过

4 下载文件 1. 点击下载按钮文件开始下载通过

致谢

参考文献

[1] 宋磊,黄祥林,沈兰荪. 视频监控系统概述[J]. 测控技术, 2003(05):33-35.

[2] 中国弱电网. 网络视频监控技术的未来发展趋势[EB/OL]. 2008. http://www.hx-avic.com/shichangyanjiu/200809/08-118.html.

[3] 万梅芬. 基于流媒体技术的数字化校园文化设计与实现[J]. 中国管理信息化, 2018, 021(020):152-153.

[4] 小岛上的黑桃六. 流媒体：RTMP协议完全解析[EB/OL]. 2020. https://zhuanlan.zhihu.com/p/191542130.

[5] 维基百科. Flash Video[EB/OL]. 2020. https://zh.wikipedia.org/wiki/Flash_Video.

[6] 魏雪飞,周祥. HLS流媒体技术在广播电视网络直播系统的应用[J]. 广播电视信息, 2020(9).

[7] 又拍云. RTMP、HTTP-FLV、HLS，你了解常见的三大直播协议吗[EB/OL]. 2018. https://www.upyun.com/tech/article/352/RTMP%E3%80%81HTTP-FLV%E3%80%81HLS%EF%BC%8C%E4%BD%A0%E4%BA%86%E8%A7%A3%E5%B8%B8%E8%A7%81%E7%9A%84%E4%B8%89%E5%A4%A7%E7%9B%B4%E6%92%AD%E5%8D%8F%E8%AE%AE%E5%90%97.html.

[8] 赖均等. 软件工程[M]. 清华大学出版社, 2016.

http://www.hx-avic.com/shichangyanjiu/200809/08-118.html

http://www.hx-avic.com/shichangyanjiu/200809/08-118.html

https://zhuanlan.zhihu.com/p/191542130

https://zhuanlan.zhihu.com/p/191542130

https://zh.wikipedia.org/wiki/Flash_Video

https://www.upyun.com/tech/article/352/RTMP%E3%80%81HTTP-FLV%E3%80%81HLS%EF%BC%8C%E4%BD%A0%E4%BA%86%E8%A7%A3%E5%B8%B8%E8%A7%81%E7%9A%84%E4%B8%89%E5%A4%A7%E7%9B%B4%E6%92%AD%E5%8D%8F%E8%AE%AE%E5%90%97.html

https://www.upyun.com/tech/article/352/RTMP%E3%80%81HTTP-FLV%E3%80%81HLS%EF%BC%8C%E4%BD%A0%E4%BA%86%E8%A7%A3%E5%B8%B8%E8%A7%81%E7%9A%84%E4%B8%89%E5%A4%A7%E7%9B%B4%E6%92%AD%E5%8D%8F%E8%AE%AE%E5%90%97.html

https://www.upyun.com/tech/article/352/RTMP%E3%80%81HTTP-FLV%E3%80%81HLS%EF%BC%8C%E4%BD%A0%E4%BA%86%E8%A7%A3%E5%B8%B8%E8%A7%81%E7%9A%84%E4%B8%89%E5%A4%A7%E7%9B%B4%E6%92%AD%E5%8D%8F%E8%AE%AE%E5%90%97.html

https://www.upyun.com/tech/article/352/RTMP%E3%80%81HTTP-

报告指标说明

- 原文总字符数：即送检文献的总字符数，包含文字字符、标点符号、阿拉伯数字（不计入空格）
- 检测字符数：送检文献经过系统程序处理，排除已识别的参考文献等不作为相似性比对内容的部分后，剩余全部参与相似性检测匹配的文本字符数
- 总相似比：送检文献与其他文献的相似文本内容在原文中所占比例
- 参考文献相似比：送检文献与其标明引用的参考文献的相似文本内容在原文中所占比例
- 可能自引相似比：送检文献与其作者本人的其他已公开或发表文献的相似文本内容在原文中所占比例
- 单篇最大相似比：送检文献的相似文献中贡献相似比最高一篇的相似比值
- 是否引用：该相似文献是否被送检文献标注为其参考文献引用，作者本人的可能自引文献也应标注为参考文献后方能认定为“引用”