

班 级 1703014
学 号 17039110002

西安电子科技大学

本科毕业设计论文



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

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。基于此,本文设计并实现了一个基于 **Spring Boot** 的视频实时监控系统。本文首先介绍了视频监控系统的相关概念、研究现状和发展趋势。然后介绍了流媒体传输技术的有关概念,分析比较了 **RTMP**、**HTTP-FLV** 和 **HLS** 这三个常用的流媒体传输协议,对三者的格式和原理进行了的阐述,并讨论了它们在实际使用过程中各自的利弊,最终决定使用 **HTTP-FLV** 进行系统开发。此外也介绍了基于 **Java** 的 **Spring Boot** 开发框架的使用方法,并对其特点、设计理念和功能做了细致的阐述。最后,本文详细的阐述了本系统的总体设计和实现细节。在此次毕业设计的任务中,运用到了 **Spring Boot**、**Redis**、**MapStruct** 和 **Vue.js** 等技术,逐步完成了视频实时监控系统的技术方案设计。最终实现了一个基于 **Spring Boot** 的视频实时监控系统,具备实时监控视频查看、监控视频保存和检索等功能。

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

Abstract

With the explosive growth in the number of various types of smart devices in recent years, video surveillance systems are developing in the direction of digitization, intelligence, networking, and humanization. Based on this, this paper designs and implements a real-time video surveillance system based on Spring Boot. This article first introduces the related concepts, research status and development trend of the video surveillance system. Then introduced the related concepts of streaming media transmission technology, analyzed and compared the three commonly used streaming media transmission protocols RTMP, HTTP-FLV and HLS, The format and principle of the three were elaborated, and their respective pros and cons in actual use were discussed. Finally, it was decided to use HTTP-FLV for system development. In addition, the use of the Java-based Spring Boot development framework is introduced, and its characteristics, design concepts and functions are elaborated. Finally, this paper elaborates on the overall design and implementation details of the system. In the task of this graduation project, technologies such as Spring Boot, Redis, MapStruct and Vue.js were used to gradually complete the technical scheme design of the real-time video surveillance system. Finally, a real-time video monitoring system based on Spring Boot is realized, with functions such as real-time monitoring video viewing, monitoring video storage and retrieval.

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

Abstract

目 录

第一章 引言	1
1.1 监控系统的发展趋势和研究现状	1
1.1.1 发展趋势	1
1.1.2 研究现状	2
1.2 本文主要创新性工作	3
1.3 本文结构	4
第二章 关键技术	5
2.1 流媒体传输技术	5
2.1.1 流媒体以及流式传输	5
2.1.2 基于 RTMP 的流媒体传输技术	7
2.1.3 基于 HTTP-FLV 的流媒体传输技术	8
2.1.4 基于 HLS 的流媒体传输技术	10
2.1.5 RTMP、HTTP-FLV、HLS 简单对比	11
2.2 Spring Boot 开发框架	11
2.2.1 Spring Boot 开发框架简介	11
2.2.2 控制反转和依赖注入特性简介	13
2.2.3 面向切面编程简介	14
2.3 本章小结	15
第三章 系统需求分析与总体设计	17
3.1 需求分析	17
3.1.1 功能性需求	17
3.1.2 非功能性需求	23
3.2 系统数据库设计	23
3.3 系统架构设计	24
3.3.1 公共模块	24

3.3.2	数据存储模块	25
3.3.3	业务逻辑模块	26
3.3.4	视频文件存取模块	26
3.3.5	前端接口交互模块	26
3.4	本章小结	27
第四章	系统实现和系统测试	29
4.1	前端界面实现	29
4.1.1	登陆界面	29
4.1.2	实时监控播放	30
4.1.3	监控回放播放	30
4.1.4	页面布局	31
4.1.5	二级菜单页面	33
4.2	后端服务实现	35
4.2.1	图形验证码接口	35
4.2.2	登陆接口	36
4.2.3	列表查询接口	36
4.2.4	监控视频保存	37
4.2.5	文件上传下载	38
4.3	系统测试	40
4.3.1	测试环境	40
4.3.2	测试用例及结果	41
4.4	本章小结	44
第五章	总结与展望	45
5.1	总结	45
5.2	展望	45
5.3	本章小结	45
致谢	47
参考文献	49

第一章 引言

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智能化方向不断发展^[1]。近些年来,视频监控在交通违法抓拍、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。

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统一管理,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。基于此,网络摄像头和网络视频监控系统应运而生。

1.1 监控系统的发展趋势和研究现状

1.1.1 发展趋势

上个世纪八十年代,监控系统出现首次出现在人们的生活中,这将近四十年发展历程可以分为四个主要阶段:模拟监控系统、半数字化监控系统、全数字化网络监控系统、智能监控系统。

模拟监控系统

模拟监控系统是第一代监控系统,主要由三个模块组成:采集模块、传输模块和存储模块。它们之间的数据交互都是采用的模拟信号,传输介质是同轴电缆。这一代监控系统往往能支持的监控范围不会很大,因为受到电缆信号传输距离的限制。此外,因为视频数据是以模拟信号的形式进行传输,存储介质是录像带,这就导致了视频数据的存储、分发和分析极为不便。

半数字化监控系统

半数字化监控系统是第二代监控系统，它提供了基“半数字-半模拟”化的监控系统解决方案，所以又被称为“模拟-数字”监控系统。半数字化监控系统一般由摄像机、数字硬盘录像机（Digital Video Recorder, DVR）、存储设备等组成。在这一代监控系统中，摄像机通过同轴电缆传输模拟信号给 DVR，DVR 将模拟信号转化为数据信号，编码压缩后存储，所以同时支持录像和回放。但是由于仍然使用同轴电缆传输信号，在摄像头数量增加时，布线的复杂程度也会上升。

全数字化网络监控系统

全数字化网络监控系统是第三代监控系统，它的诞生意味着监控系统进入了数字化的时代。第三代系统运用了更为先进的 D/A、A/D 转换设备视频服务器，或内置处理器的网络摄像机把图像处理（采集、压缩、协议转换、传输）设置在监控点^[2]，并通过互联网传输数字形式的视频数据。这一代监控系统使用的摄像头大多是指将图像信号编码为数字信号的 IP 摄像机，因此可以借助强大的互联网传输数据。

智能监控系统

智能监控系统是第四代监控系统，它集成了人工智能，可以对图像数据进行分析处理。得益于人工智能学科的产生和发展，监控系统也朝着智能化的方向发展。在这一代监控系统中，可以利用计算机对视频和图像数据进行分析，进而完成以往需要利用肉眼识别的工作，例如人流量计算、车牌识别、违章抓拍等等。因此，人工智能的出现，极大地扩展了监控系统的适用范围。

1.1.2 研究现状

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）。

视频监控支持兼容不同设备。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，每个厂商都有自己的协议和平台，这就会产生设备兼容性问题。这就需要制定一个统一的标准，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。

视频监控支持接入人工智能系统。现在市面上已经存在很多智能的监控摄像头和监控系统，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智能摄像头解决。

1.2 本文主要创新性工作

本文基于 Spring Boot 设计并实现了一个视频实时监控系統。本系統將傳統的視頻監控系統與網頁結合，擁有鏈接安防攝像頭、查看實時視頻、視頻保存和查看功能。

系統主要分為服務器端和網頁客戶端。服務器端主要使用 Spring Boot 編寫實現，採用 HTTP 協議與網頁客戶端交互，接口依照 RESTful 風格編寫，並使用 Java-CV 依賴對攝像頭視頻數據進行採集保存。網頁客戶端主要使用 Vue.js 框架編寫，主要實現了用戶登錄、實時監控查看、監控回放查看、增減設備、個人中心、權限管理等功能。總的來說，本文的主要創新性工作可以概括為以下幾點：

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。

1.3 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。

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，分析比较了 RTMP、HTTP-FLV 和 HLS 三个流媒体传输协议。紧接着对开发框架 Spring Boot 的功能和特性进行了介绍。

第三章从功能性需求和非功能性需求两方面对系统进行了需求分析，然后介绍了系统的数据库设计和系统架构设计。

第四章分别从前端页面和后端服务两个角度对系统关键功能的实现进行了介绍。之后介绍了系统关键功能的测试过程和结果，首先说明了系统的测试环境，然后以表格的形式介绍了系统关键功能的测试用例和结果。

第五章对本文完成的工作进行了总结和归纳，并对监控系统的设计和实现提出了自己的思考。

第二章 关键技术

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。如果不使用流媒体，那么就需要在播放音视频等多媒体文件之前下载整个文件。相反的，使用流媒体则不需要下载整个文件，数据会向流水一样依次被客户端加载，一边传输一边播放。

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”。想要使用流媒体就离不开流式传输。在流式传输中，音视频等多媒体数据通过流媒体服务器向客户端连续地、实时地传输，所以只需要经过短暂的加载时间，就可以进行观看。当音视频等多媒体在客户端上播放时，剩余的部分仍然会由流媒体服务器继续传输。流式传输极大的缩短了多媒体文件播放的加载时间。

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。

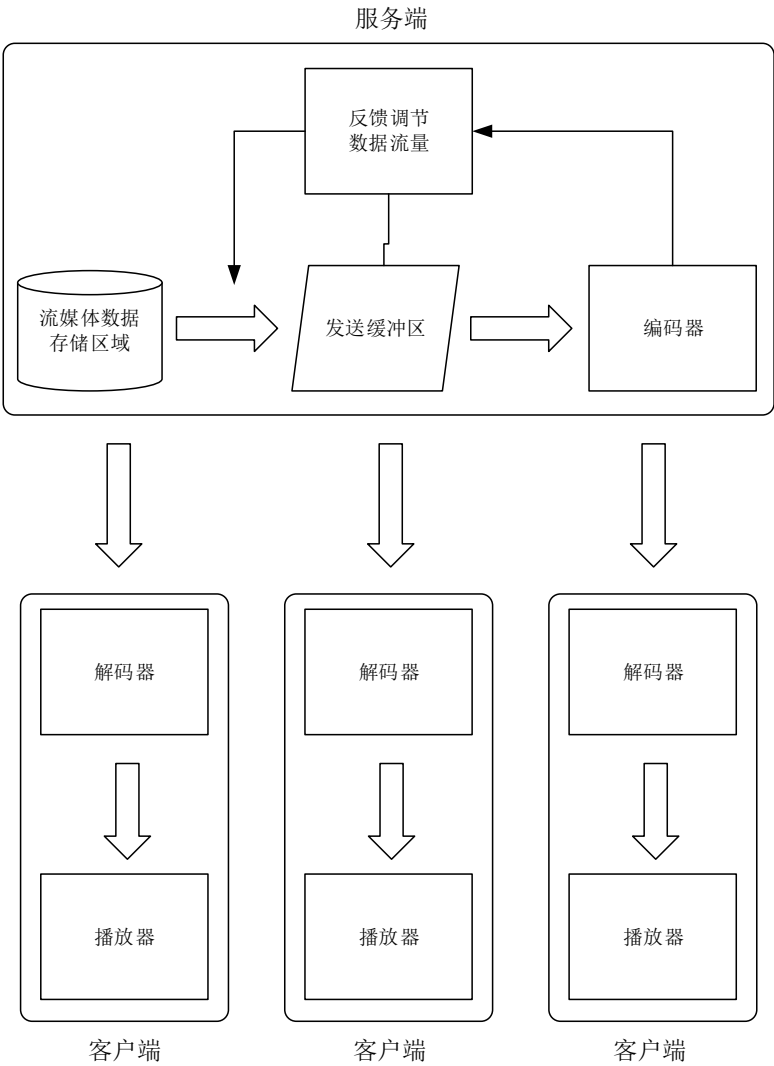


图 2.1 流式传输示意图

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 是基于 TCP 的明文协议，端口的缺省值为 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（**Extended Timestamp**）构成。

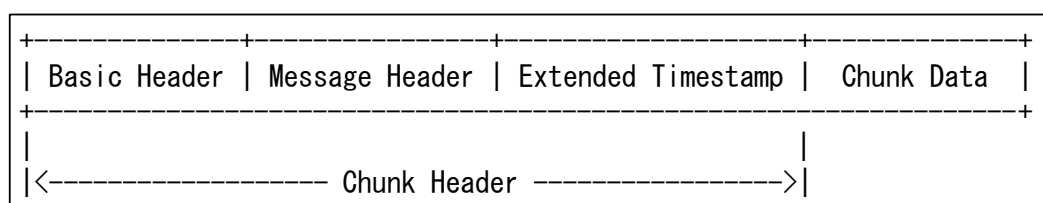


图 2.2 Chunk 格式示意图

消息（**Message**）是协议中的基本数据单元，消息块（**Chunk**）是比消息更小的数据单元。在数据发送时，消息会被分割为消息块，然后消息块会通过 TCP 发送出去。在数据接收时，消息块会被组装成消息，最终解析成原始的流媒体数据。

通常情况下，一个有效的消息，如果数据量超出 **Chunk Size** 的话，则会被拆分成多个消息块来分批传输。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。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。

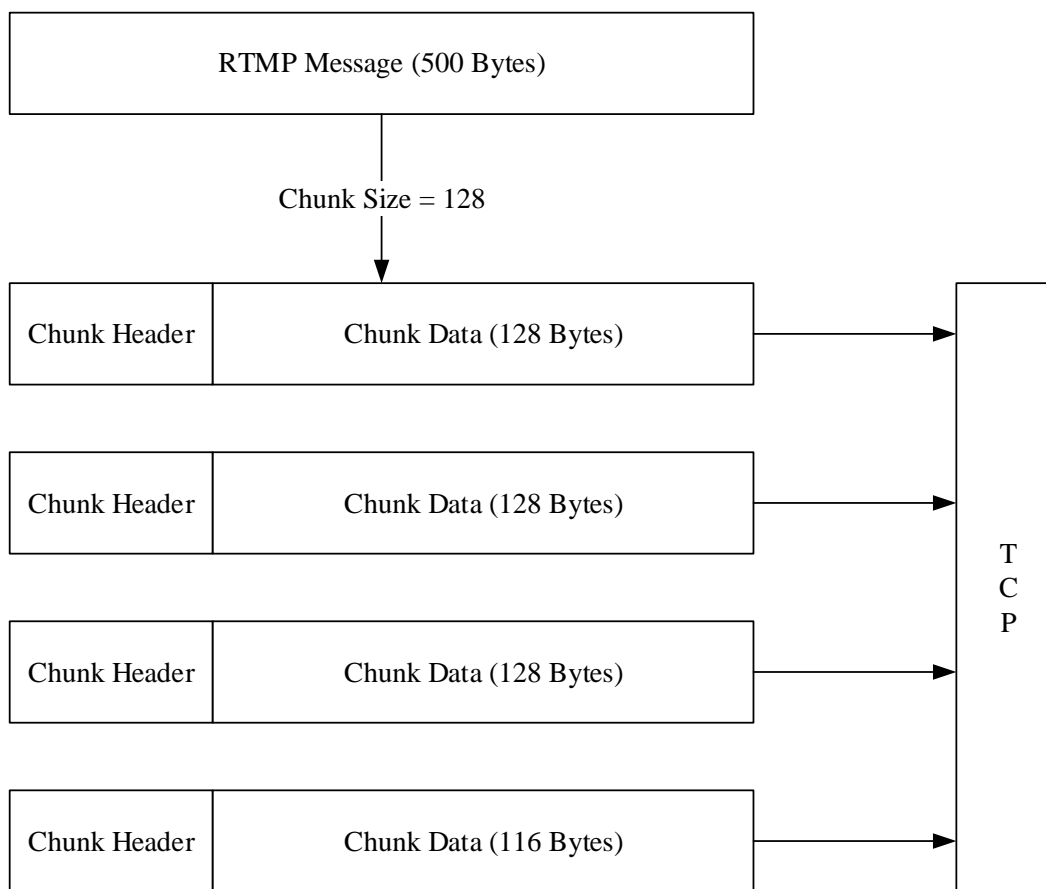


图 2.3 消息拆分示意图

2.1.3 基于 HTTP-FLV 的流媒体传输技术

HTTP-FLV 协议首先会将流媒体数据按照 FLV 格式进行封装，然后再使用 HTTP 传输封装之后的数据。

FLV 是一种适用于流式传输的网络视频格式，全称为 Flash Video。FLV 文件格式由 FLV Header 和 FIV 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。

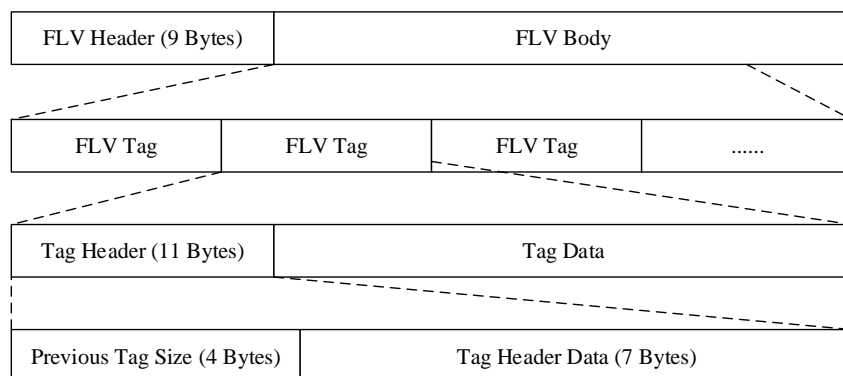


图 2.4 FLV 协议格式

FLV Tag 共有三种类型，分别是 Video Tag、Audio Tag 和 Script Tag，通过 Tag Header 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 Previous Tag 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。

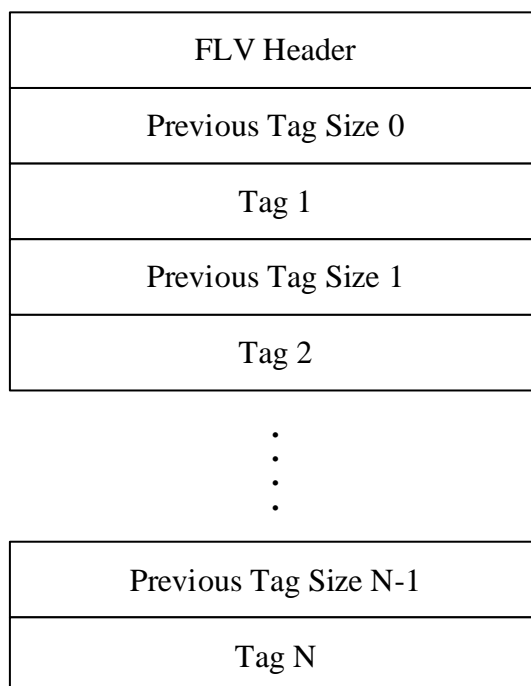


图 2.5 FLV 文件组成结构

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 iPhone3.0 中首次尝试了该技术^[5]。

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。

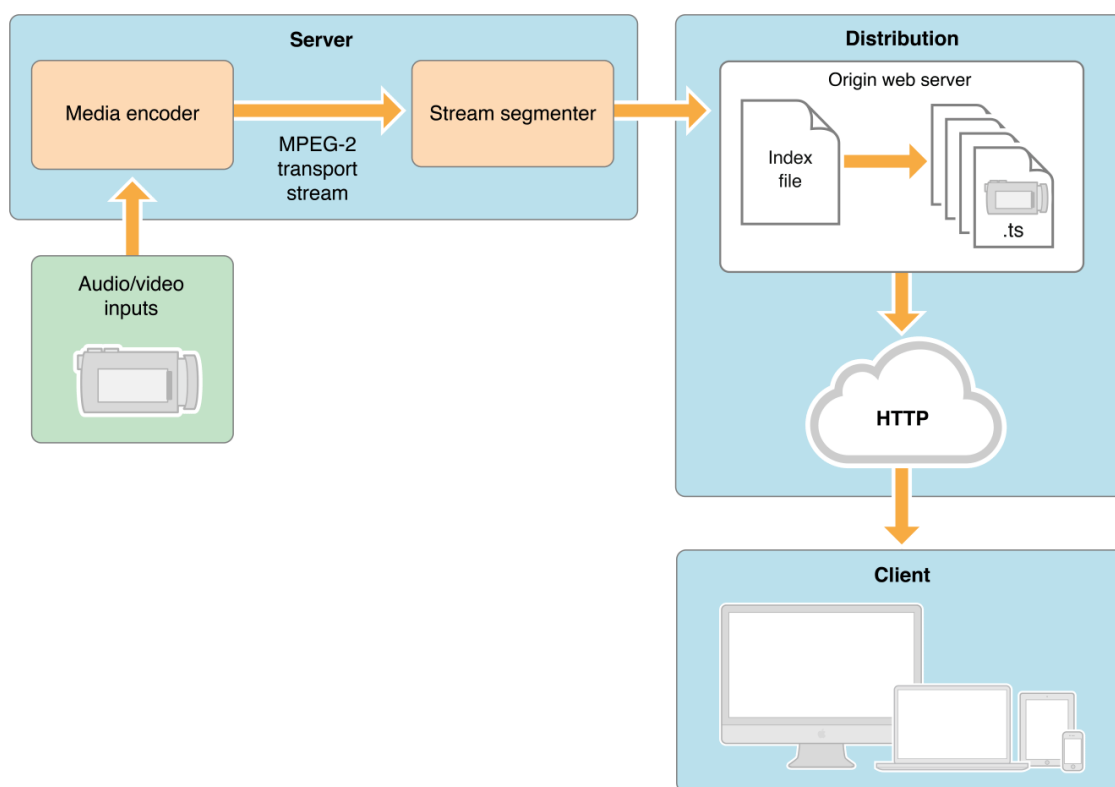


图 2.6 HLS 协议工作原理

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。

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^[6]

协议	底层传输协议	视频封装格式	延迟	数据分段	HTML5
RTMP	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。

综上所述，本系统将采用 HTTP-FLV 协议进行设计和实现。

2.2 Spring Boot 开发框架

2.2.1 Spring Boot 开发框架简介

Spring Boot 框架由 Pivotal 团队开发，它通过“自动装配”的方式来进行配置，可以省略 XML 文件的编写，因此极大程度的简化了基于 Spring 框架的应用的搭建和开发。

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。

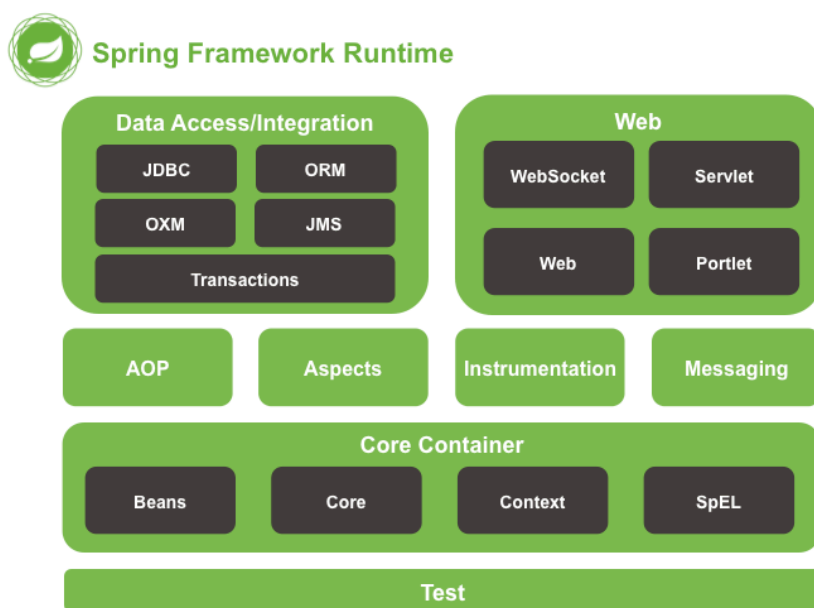


图 2.7 Spring 框架模块结构

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 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。

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。

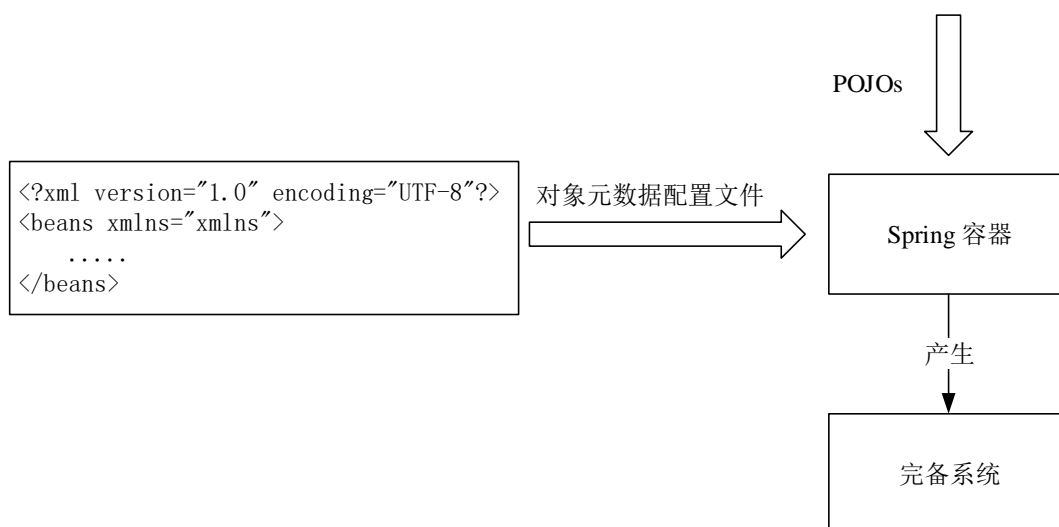


图 2.8 IoC 和 DI 工作过程

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, 有以下几个重要的概念:

1. 连接点 (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): 在某个特定的连接点上执行的一段额外逻辑。可以是在连接点之前、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连接点。对于方法执行的连接点, 切点可以是 “当参数为某一指定值”。
4. 切面 (Aspect): 切面是切点和增强的组合。

在程序执行的过程中, 有若干个连接点, 每个连接点都可以成为 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）对应的连接点（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**。

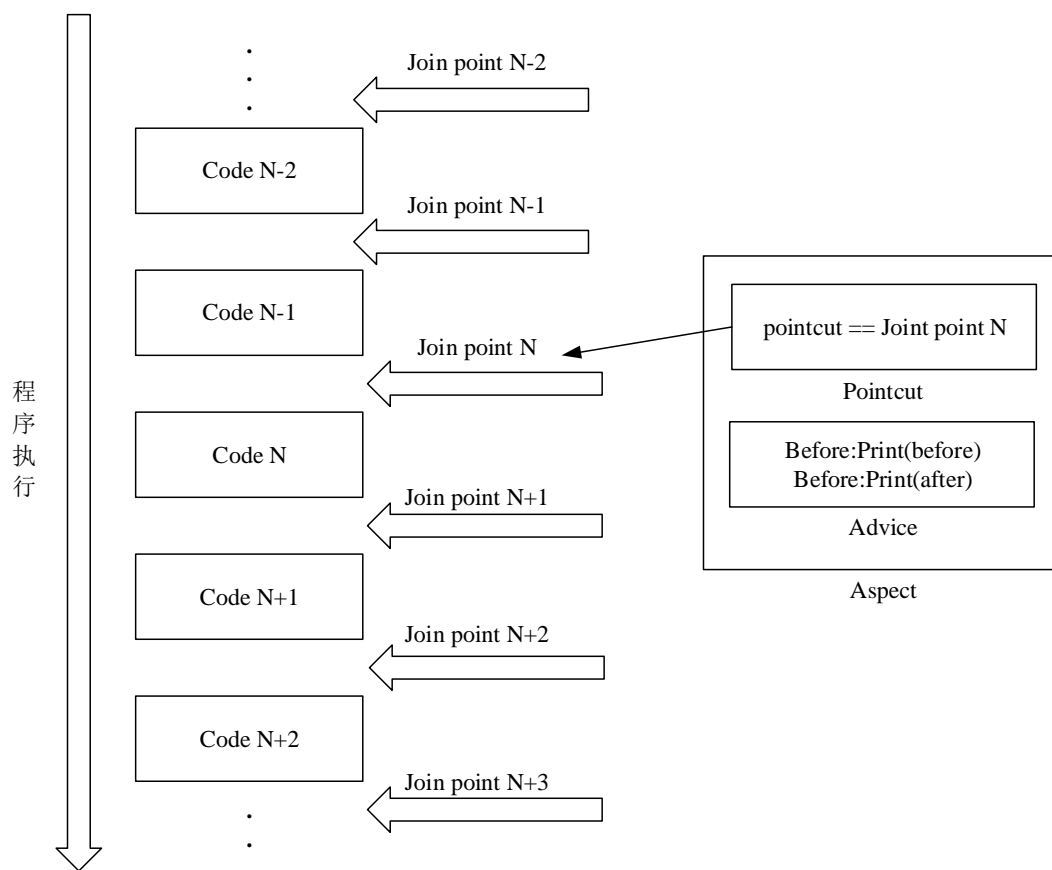


图 2.9 AOP 原理示意图

2.3 本章小结

本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，分析比较了 RTMP、HTTP-FLV 和 HLS 三个流媒体传输协议。紧接着对开发框架 Spring Boot 的功能和特性进行了介绍。

第三章 系统需求分析与总体设计

3.1 需求分析

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。

图 3.1 是系统用户用例图，体现了系统的大致功能。

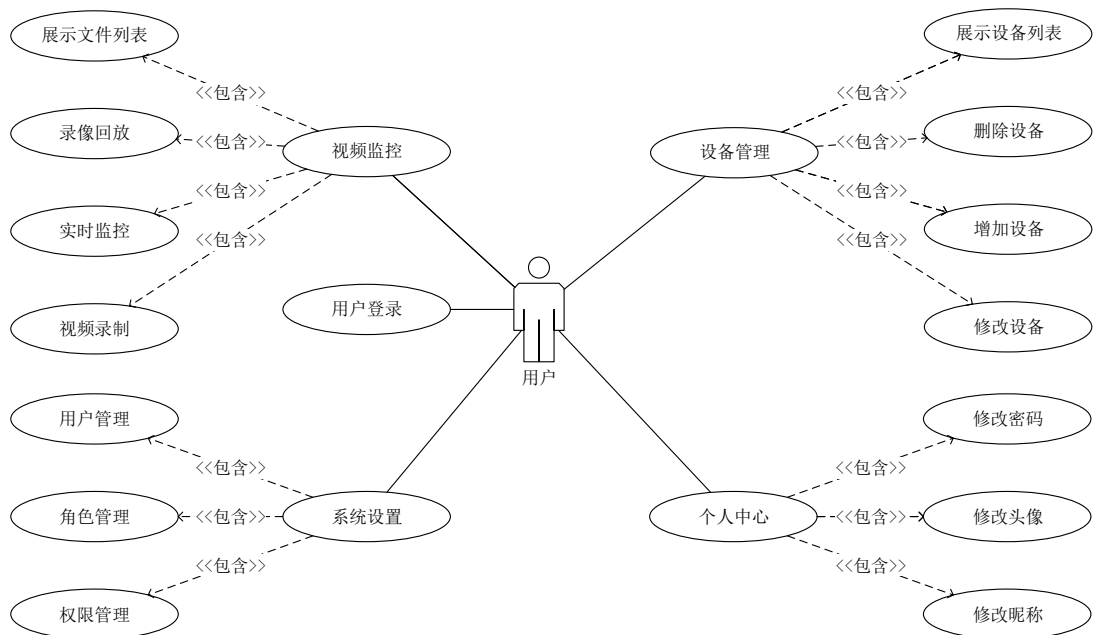


图 3.1 系统用户用例图

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 3.5 所示。

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。

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。然后再借助 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。

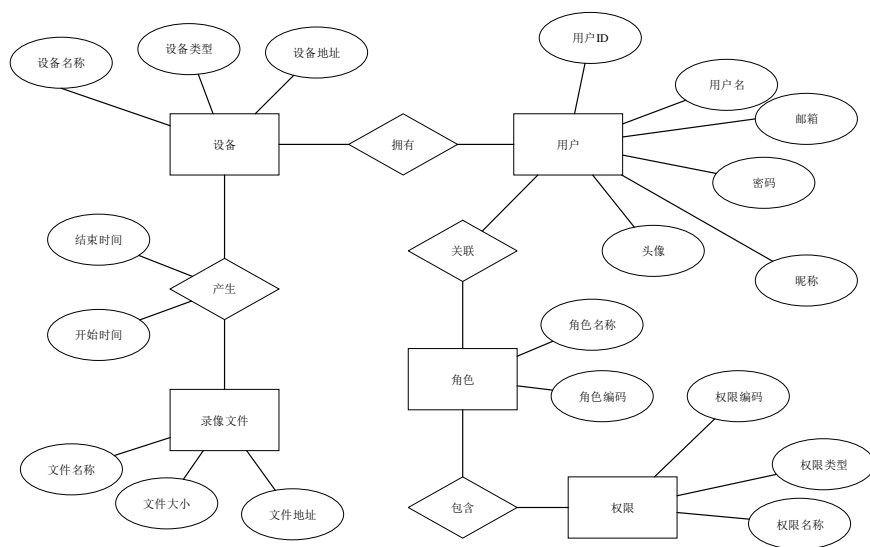


图 3.2 实体关系图

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

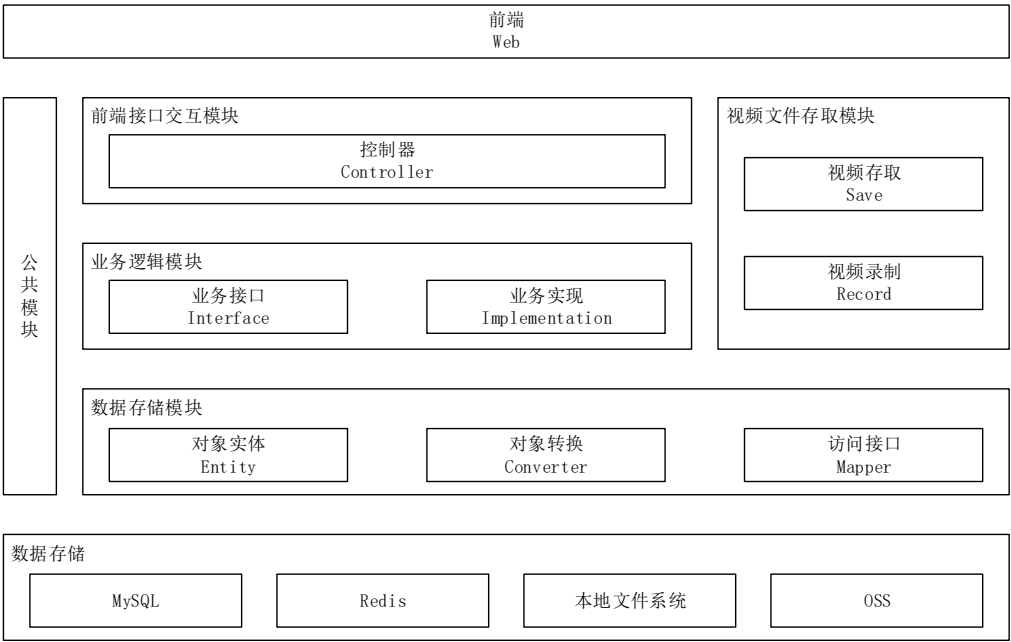


图 3.3 软件总体框架图

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，如图 3.4 所示。

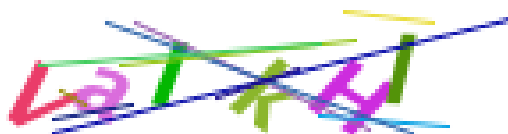


图 3.4 登陆验证码

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、Redis 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。

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。

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。

3.4 本章小结

本章从功能性需求和非功能性需求两方面对系统进行了需求分析，然后介绍了系统的数据库设计和系统架构设计。

第四章 系统实现和系统测试

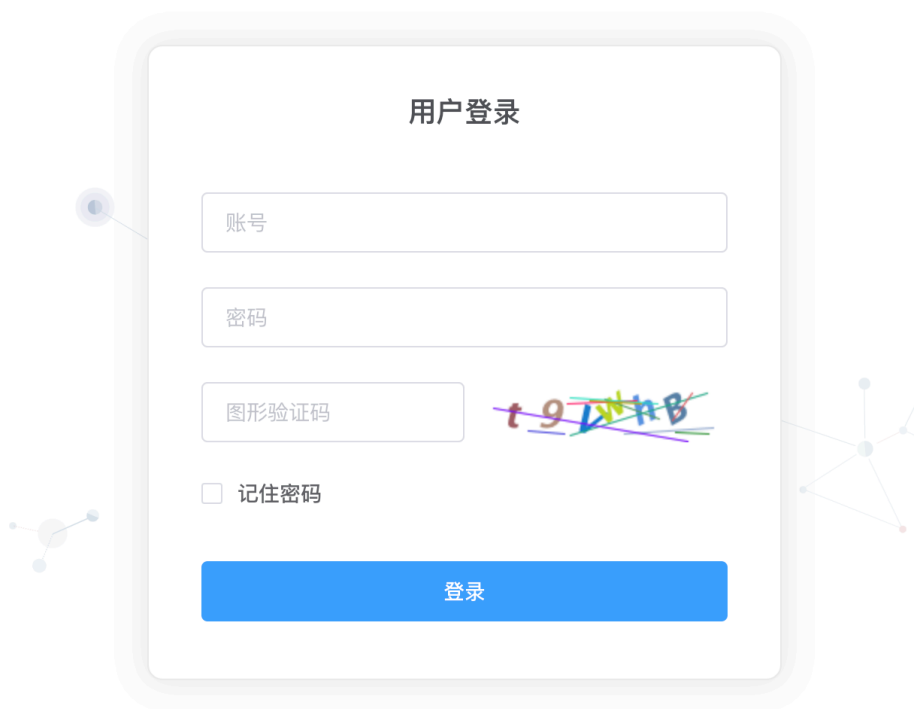
本章将介绍系统实现和系统测试两大部分。前端使用 **Vue.js** 框架进行开发，通过 **yarn** 和 **npm** 进行项目依赖管理。开发 IDE 使用 **Visual Code**。后端使用 **Spring Boot** 框架进行开发，通过 **Maven** 进行项目依赖管理。开发 IDE 使用 **IDEA**。

4.1 前端界面实现

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实现，主要采用 **Element-UI** 开源组件开发。

4.1.1 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。



The diagram illustrates a user login interface titled "用户登录" (User Login). It features three input fields: "账号" (Account), "密码" (Password), and "图形验证码" (Image CAPTCHA). To the right of the CAPTCHA field is a sample image showing a string of characters "t9LxhB" with colored lines. Below the input fields is a checkbox labeled "记住密码" (Remember Password). At the bottom is a blue button labeled "登录" (Login). The entire interface is enclosed in a light gray rounded rectangle, with decorative geometric shapes on the left and right sides.

图 4.1 登陆页面

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。

4.1.2 实时监控播放

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。



图 4.2 实时监控播放示例

4.1.3 监控回放播放

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。

文件名称	文件大小	开始时间	结束时间	操作
2021-04-23-19-06-17.mp4	9527321	2021-04-23 19:06:17	2021-04-23 19:07:24	<button>播放</button> <button>下载</button>
2021-04-23-19-07-28.mp4	15235622	2021-04-23 19:07:28	2021-04-23 19:08:28	<button>播放</button> <button>下载</button>
2021-04-23-19-08-00.mp4	12793256	2021-04-23 19:08:01	2021-04-23 19:09:01	<button>播放</button> <button>下载</button>
2021-04-23-19-20-50.mp4	22059788	2021-04-23 19:20:50	2021-04-23 19:22:10	<button>播放</button> <button>下载</button>
2021-04-23-19-21-50.mp4	16549683	2021-04-23 19:21:50	2021-04-23 19:23:10	<button>播放</button> <button>下载</button>
2021-04-23-19-22-50.mp4	16744658	2021-04-23 19:22:50	2021-04-23 19:24:10	<button>播放</button> <button>下载</button>
2021-04-23-19-23-50.mp4	16761208	2021-04-23 19:23:50	2021-04-23 19:25:10	<button>播放</button> <button>下载</button>

图 4.3 监控回放播放示例

4.1.4 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。

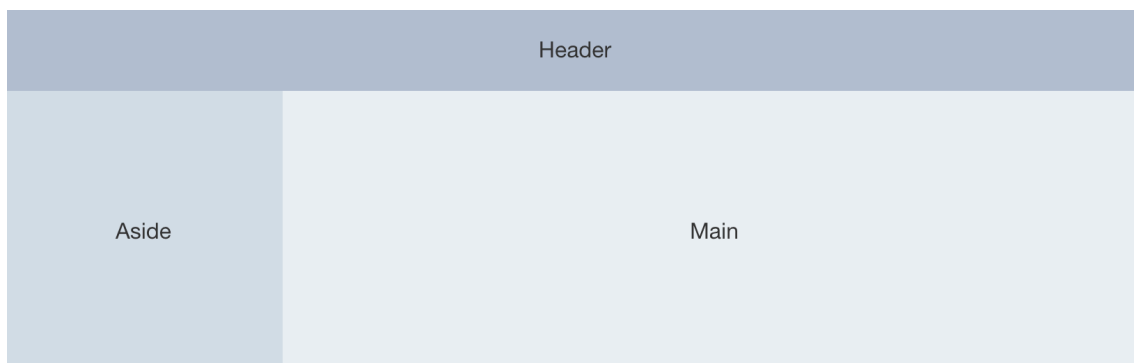


图 4.4 页面布局

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。



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。

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。

4.1.5 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

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。

首页 > 设备管理

搜索:

图 4.9 设备管理页面的搜索部分

首页 > 监控回放 > 回放文件

选择起止时间: 至

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。

4.2 后端服务实现

本节将介绍后端服务的实现，主要采用 Spring Boot 框架开发，通过 RESTful 形式的 HTTP 接口与前端交互。

4.2.1 图形验证码接口

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。

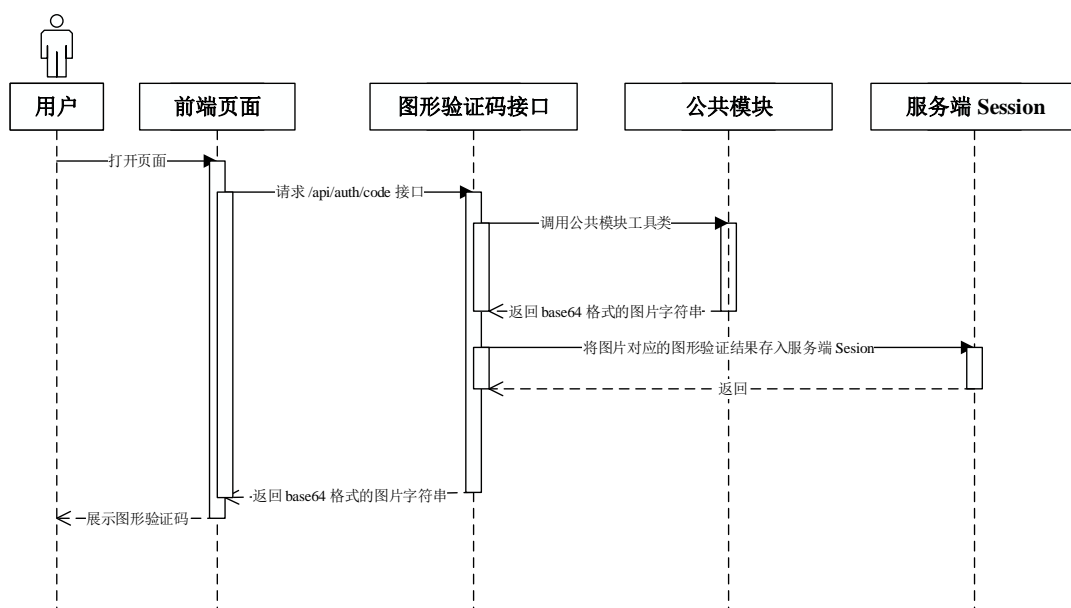


图 4.11 图形验证码接口时序图

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。

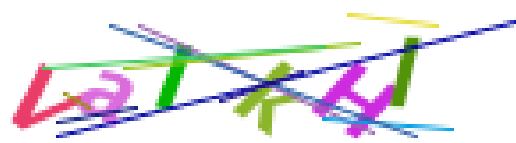


图 4.12 图形验证码

4.2.2 登陆接口

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：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。完整的时序图如图 4.13 所示。

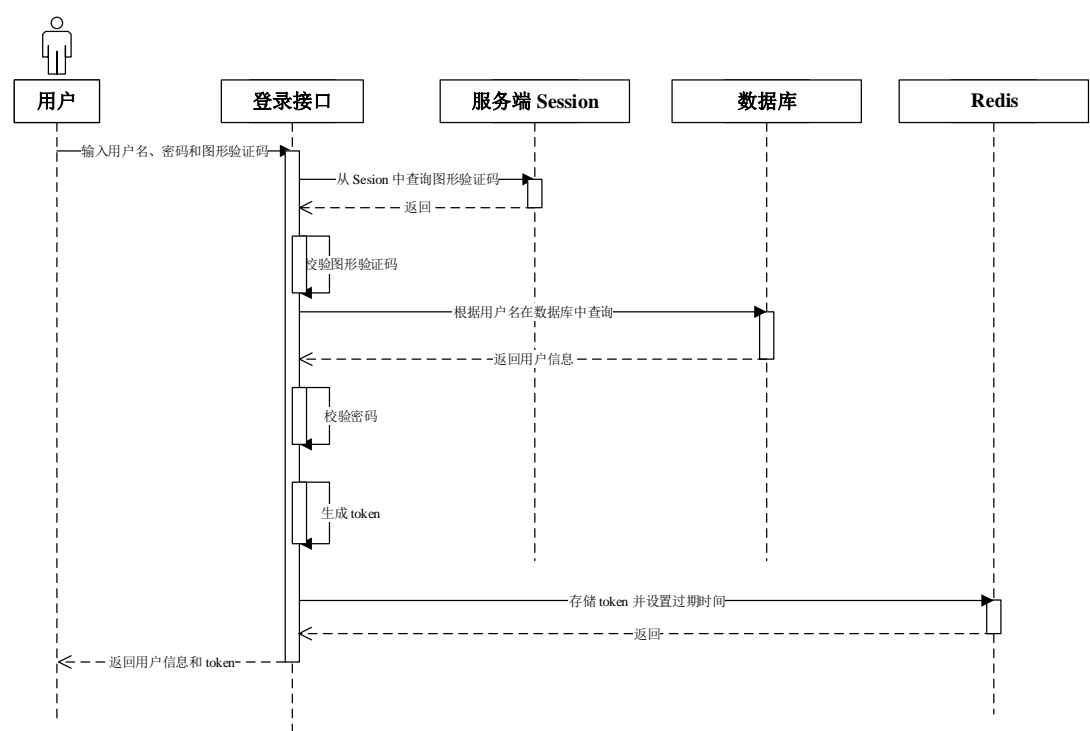


图 4.13 登陆接口时序图

4.2.3 列表查询接口

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。

4.2.4 监控视频保存

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，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。

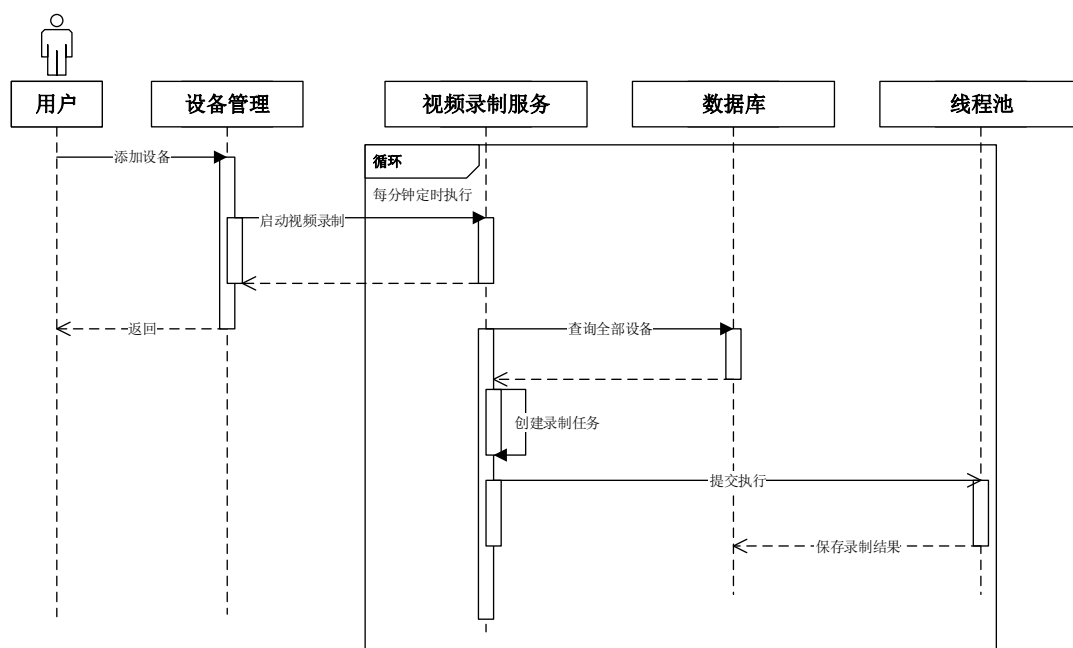


图 4.14 监控视频保存流程图

4.2.5 文件上传下载

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 `MultipartFile` 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。

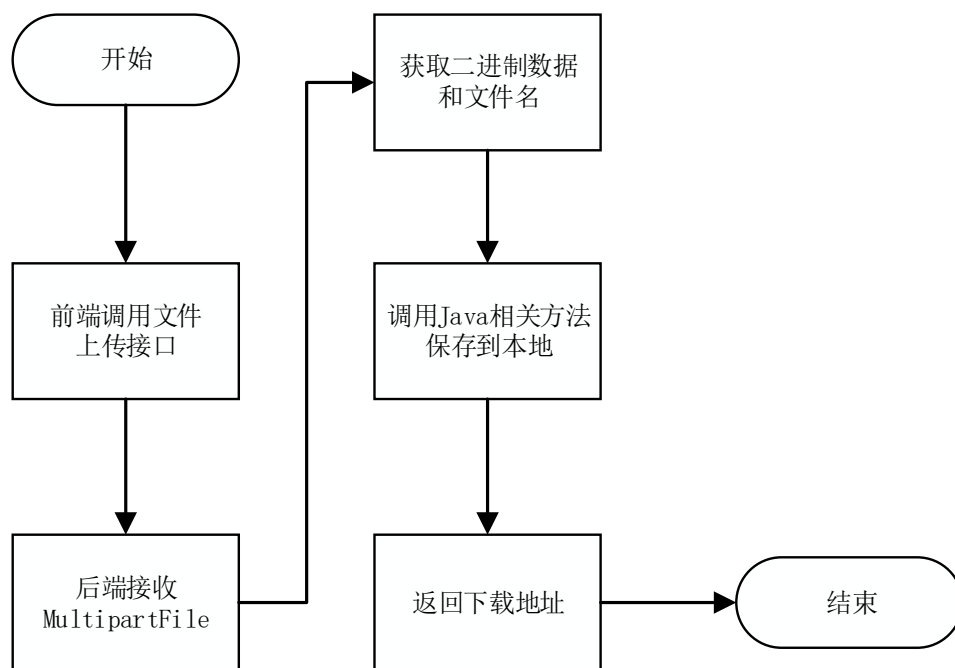


图 4.15 文件上传流程图

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。

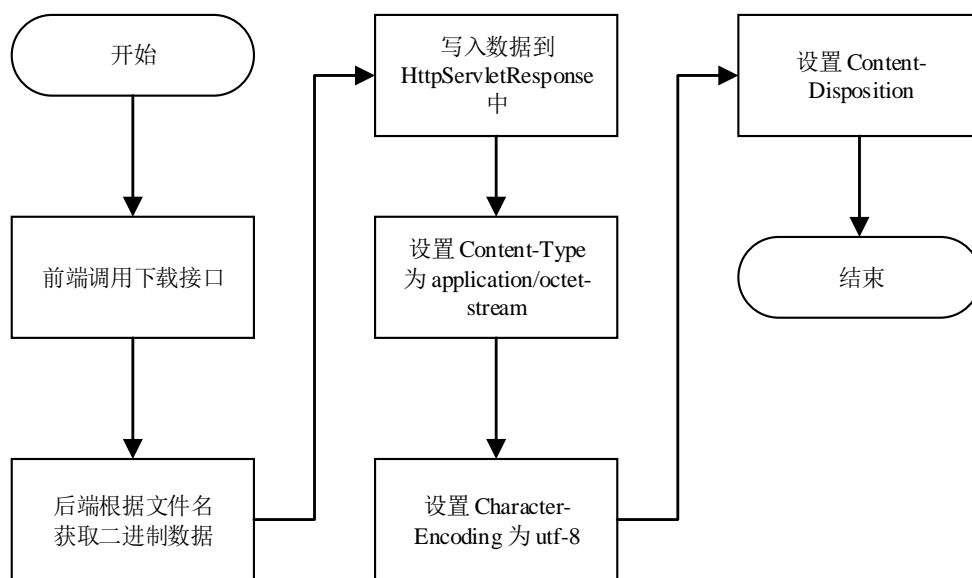


图 4.16 文件下载流程图

4.3 系统测试

本节的主要工作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。

4.3.1 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 4.1 所示。

表 4.1 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 4.2 所示。

表 4.2 测试服务器配置表

操作系统	Windows 10
处理器	Intel Core i5-7400
内存	8G
Redis	6.2.1
MySQL	8.0.13

4.3.2 测试用例及结果

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。

实时监控功能测试

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 4.3 列出了部分测试用例以及测试结果。

表 4.3 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 搜索	1. 不输入设备名称 2. 输入设备名称	列表展示符合 条件的结果	通过
3	设备类型 搜索	1. 选择全部 2. 选择除全部外的选项	列表展示符合 条件的结果	通过
4	实时监控 播放	1. 点击查看实时监控视频按钮	播放实时监控	通过

视频保存和检索功能测试

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 4.4 列出了部分测试用例以及测试结果。

表 4.4 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 2. 不选择开始和结束时间	列表展示符合条件的结果	通过
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

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 4.5 列出了部分测试用例以及测试结果。

表 4.5 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 搜索	1. 不输入设备名称 2. 输入设备名称	列表展示符合 条件的结果	通过
3	设备类型 搜索	1. 选择全部 2. 选择除全部外的选项	列表展示符合 条件的结果	通过
4	添加设备	1. 输入设备名称 2. 选择设备类型 3. 输入设备地址	提示添加成功	通过
5	修改设备	1. 修改设备名称 2. 修改设备类型 3. 修改设备地址 4. 点击保存按钮	提示修改成功	通过
6	修改删除 状态	1. 修改为删除 2. 修改为不删除	设备删除状态变更	通过

4.4 本章小结

本章分别从前端页面和后端服务两个角度对系统关键功能的实现进行了介绍。之后介绍了系统关键功能的测试过程和结果，首先说明了系统的测试环境，然后陈述了系统关键功能的测试用例和结果。

第五章 总结与展望

5.1 总结

本文介绍了一种基于 **Spring Boot** 的视频实时监控系统的设计思路和实现方法。实现了如下功能：

1. 链接安防摄像头，实时查看视频监控
2. 视频按时间顺序保存并支持检索
3. 用户登陆和权限管理

本系统将传统的安防摄像头与互联网结合，使用户可以通过网页查看实时监控视频和历史视频监控，符合互联网时代的需求和发展潮流。

5.2 展望

本系统实现了实时监控视频查看、监控视频回放查看、监控视频回放检索等功能，但还是有很多可以完善和改进的地方。首先，从功能方面，可以增加对监控摄像头操作移动的功能，现在市面上许多摄像头以及支持远程操作，只需要基于对方厂商提供的 **SDK** 进行二次开发即可。其次，从性能方面，可以对监控视频的录制做改进，本系统是通过遍历全部的摄像头来发起视频录制任务，但是对于自身带有存储的摄像头，可以直接通过厂商的 **SDK** 访问它的本地系统来进行回放文件的读取，无需再遍历。

5.3 本章小结

本章对本文完成的工作进行了总结和归纳，并对监控系统的设计和实现提出了自己的思考。

致 谢

四年一晃而过，期间有期待、兴奋、惊喜交集，也有焦虑、恐惧、忐忑不安，但唯一不变的是我对未来生活的憧憬和思考。完成毕业设计，离开生活了四年的学校，是一个终点，但也是另一个起点！

首先，我要感谢西电的所有老师，能成为他们的学生是我的荣幸。高等数学、线性代数和离散数学教会我如何的理性思考问题；马原、毛概和形势与政策，教会我如何用哲学的思维理解世界。毕业设计的指导老师和学长在我遇到困惑的时候，都会积极解答。

其次，我要感谢在公司实习中认识的导师、同事和小伙伴。刚踏入职场的我，对工作可以说是一窍不通，感谢导师和同事给了我足够的时间去适应和熟悉工作环境。在他们的帮助和指导下，我慢慢的了解如何工作，跟上了其他人的节奏。

然后，我要感谢母校西电的栽培，愿西电越办越好，不断提高国际影响力，在电子信息、计算机科学与技术等方向上越做越好。依稀记得我刚刚进入校园的时候，不论做什么事，我都畏手畏脚，不敢多问，不敢多说。而四年之后的我，变得更加的开朗，这离不开西电四年陪伴。

再次，我要感谢西安这座历史底蕴丰厚的城市，它见证了我从一个稚嫩的人，成长为能肩负责任的男子汉。多少次往返于杭州和西安，行走在承载历史的古城墙，徜徉于大明宫、大雁塔和秦始皇陵。我求学的足迹，永远的留在了这片土地上。

最后，我要感谢家人一如既往对我的关心和照顾，最坚实的精神支柱其实家人的关心。你们的健康和快乐是我今生最大的愿望。

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] 百度百科. 监控系统[EB/OL]. <https://baike.baidu.com/item/%E7%9B%91%E6%8E%A7%E7%B3%BB%E7%BB%9F/2354833>.
- [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] 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).
- [6] 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%E7%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>.