论文题目 计算机语言在线考试系统的设计与实现

**姓 名**

**学 院**

**专 业**

**指导教师**

**备 注**

**2017 年 X 月 X 日**

计算机语言在线考试系统的设计与实现

**Design and Implementation of Online Examination Monitoring System**

计算机语言在线考试系统的设计与实现

摘 要

随着家庭装修行业的竞争日益激烈，客户对装修质量的要求越来越高，装修公司如何保证自己的服务和装修质量是能否提高竞争力的关键因素，这便对装修工程的过程监控提出了更高的要求。在工程监控方面，目前比较常见的方式是监理到现场查看，或者在工程现场安装一定数目的摄像头，但是大量的人力成本与监控地点的不灵活性等因素均造成了很多不便。在这种情况下，伴随着智能手机的逐渐普及，基于移动终端的工程监控系统的开发便显现出了重要意义。

本文首先介绍了基于iOS的工程监控系统的开发背景与目的，然后介绍了系统开发过程中所使用到的关键技术，包括Objective-C语言、Xcode、SQLite、JSON、AFNetworking、ASIHttpRequest以及SDWebImage等。接着本文描述了系统的需求分析，然后详细地论述了系统的设计和实现过程。在系统的设计和实现部分，本文详细地描述了用户的登录退出管理、日报记录管理、地理位置信息管理、账务记录管理、消息与公告管理、绩效考评、审核批示、项目权限管理、用户角色管理和客户专用版子系统等功能模块的开发过程。最后本文描述了系统的测试过程，通过对相关的测试用例所产生的结果进行分析，对系统作出了整体的评价。

本文描述的系统经过测试评估，运行状态稳定，功能基本达到客户要求，目前已经通过App Store成功发布了两个版本，并已应用于多家装修公司，公司内部的团队成员可以通过系统进行协调配合，相关负责人也能够通过它全程监控项目的进展情况，并随时对重要的汇报记录作出审核批示，真正达到了移动办公与移动管理的目标。

**关键词：**Objective-C，Xcode，iOS，工程监控系统

**Design and Implementation of Online Examination Monitoring System**

**Abstract**

With the competition of home decoration industry becomes increasingly fierce and the demands on the quality of the decoration increasingly comprehensive, how to ensure the quality of their services and decoration can improve is the key factors for decoration company to sharp their edge in the race, which puts forward higher requirements for process monitoring of decorating. Concerning project monitoring, the more common methods at present is to view at the scene or to install a certain number of cameras in the engineering field, but a lot of factors such as labor costs and the inflexibility of monitoring sites have caused a lot of inconvenience. In this case, with the growing popularity of smartphones, the development of engineering monitoring system based on mobile terminal shows the important meaning.

This article firstly introduces the development background and purpose of engineering monitoring system based on iOS, and then introduces the key technology, including Objective - C, Xcode, SQLite, JSON, AFNetworking, ASIHttpRequest and SDWebImage, etc. Then this article describes the system requirements analysis and the process of system design and implementation in detail. In the section of system design and implementation, the paper describes the user login and exit management, daily record, geographical position information management, accounting records management, news and announcement management, performance evaluation, audit instructions, project permissions management, users’ role management and customer special edition subsystem, etc. And introducing their development process. At last, the paper describes the test process of the system and makes overall evaluation on system through analysis of results related test cases caused.

**Key words:** Objective-C, Xcode, iOS, Project Monitoring System

目 录

[摘 要 I](#_Toc495246250)

[**Abstract** II](#_Toc495246251)

[第1章 绪 论 1](#_Toc495246252)

[1.1 选题背景及意义 1](#_Toc495246253)

[1.2 开发目的及意义 1](#_Toc495246254)

[1.3 国内外发展现状 2](#_Toc495246255)

[1.4 论文研究内容 2](#_Toc495246256)

[1.5 论文组织结构 3](#_Toc495246257)

[第2章 关键技术 4](#_Toc495246258)

[2.1 MVC设计模式 4](#_Toc495246259)

[2.1.1 MVC组成部分介绍 4](#_Toc495246260)

[2.1.2 MVC交互流程 4](#_Toc495246261)

[2.1.3 MVC模式的优点 5](#_Toc495246262)

[2.2 SSH集成框架 5](#_Toc495246263)

[2.2.1 Struts2框架 6](#_Toc495246264)

[2.2.2 Spring框架 7](#_Toc495246265)

[2.2.3 Hibernate框架 8](#_Toc495246266)

[2.3 MySql数据库 9](#_Toc495246267)

[2.4 B/S体系结构 10](#_Toc495246268)

[2.5 本章小结 10](#_Toc495246269)

[第3章 需求分析 12](#_Toc495246270)

[2.6 3.1 系统总体需求 12](#_Toc495246271)

[3.1.1 系统业务需求 12](#_Toc495246272)

[3.1.2 系统用户需求 12](#_Toc495246273)

[2.7 3.2 系统功能需求 12](#_Toc495246274)

[3.2.1 系统的登陆退出与用户的角色管理模块 13](#_Toc495246275)

[3.2.2 本地与网络记录管理模块 13](#_Toc495246276)

[3.2.3 日常记录管理与日报记录查找模块 13](#_Toc495246277)

[3.2.4 日报记录详细信息查看与修改模块 14](#_Toc495246278)

[3.2.5 地理定位模块 14](#_Toc495246279)

[3.2.6 审核通知与公告中心管理模块 14](#_Toc495246280)

[3.2.7 财务记录管理模块 15](#_Toc495246281)

[3.2.8 签到与签退记录管理模块 15](#_Toc495246282)

[3.2.9 权限控制与安全处理模块 15](#_Toc495246283)

[3.2.10 更多设置功能模块 16](#_Toc495246284)

[3.3 非功能性需求 16](#_Toc495246285)

[2.8 3.4 本章小结 17](#_Toc495246286)

[第4章 系统设计 18](#_Toc495246287)

[2.9 4.1 系统概要设计 18](#_Toc495246288)

[4.1.1 系统主体模块 18](#_Toc495246289)

[4.1.2 系统架构分析与总体设计 18](#_Toc495246290)

[2.10 4.2 功能模块详细设计 18](#_Toc495246291)

[4.2.1 系统的登录退出与用户的角色管理模块详细设计 18](#_Toc495246292)

[4.2.2 本地与网络记录管理模块详细设计 19](#_Toc495246293)

[4.2.3 日常记录管理与日报记录查找模块详细设计 19](#_Toc495246294)

[4.2.4 日报记录详细信息查看与修改模块详细设计 20](#_Toc495246295)

[4.2.5 地理定位模块详细设计 20](#_Toc495246296)

[4.2.6 审核消息与公告中心管理模块详细设计 20](#_Toc495246297)

[4.2.7 财务记录管理模块详细设计 20](#_Toc495246298)

[4.2.8 签到与签退模块详细设计 21](#_Toc495246299)

[4.2.9 权限控制与安全处理模块详细设计 21](#_Toc495246300)

[4.2.10 更多设置功能模块详细设计 22](#_Toc495246301)

[2.11 4.3 系统部分关键内容详细设计 22](#_Toc495246302)

[4.3.1 客户端界面设计 22](#_Toc495246303)

[4.3.2 主要工具类的设计 24](#_Toc495246304)

[4.3.3 主体控制器类功能设计 25](#_Toc495246305)

[4.3.4 本地数据库设计 30](#_Toc495246306)

[4.3.5 网络接口设计 31](#_Toc495246307)

[2.12 4.4 本章小结 32](#_Toc495246308)

[第5章 系统实现 33](#_Toc495246309)

[2.13 5.1 网络请求与二次封装模块实现 33](#_Toc495246310)

[5.1.1 URL的统一编码处理 33](#_Toc495246311)

[5.1.2 不同请求方式的消息封装 33](#_Toc495246312)

[2.14 5.2 登录与退出登录的内存管理 35](#_Toc495246313)

[2.15 5.3 系统的详细功能模块实现 36](#_Toc495246314)

[5.3.1 系统的登录退出与用户的角色管理模块功能实现 36](#_Toc495246315)

[5.3.2 日报记录管理模块功能实现 37](#_Toc495246316)

[5.3.3 日报记录详细信息查看与修改模块功能实现 38](#_Toc495246317)

[5.3.4 地理定位模块功能实现 39](#_Toc495246318)

[5.3.5 审核消息与公告中心管理模块功能实现 40](#_Toc495246319)

[5.3.6 财务记录管理模块功能实现 41](#_Toc495246320)

[5.3.7 签到与签退模块功能实现 42](#_Toc495246321)

[5.3.8 用户权限管理模块功能实现 44](#_Toc495246322)

[5.3.9 更多设置模块功能实现 45](#_Toc495246323)

[5.3.10 用户记录上下拉刷新功能实现 46](#_Toc495246324)

[2.16 5.4 iOS端本地数据库功能实现 48](#_Toc495246325)

[2.17 5.5 本章小结 49](#_Toc495246326)

[第6章 系统测试 50](#_Toc495246327)

[2.18 6.1 测试环境 50](#_Toc495246328)

[2.19 6.2 测试用例及结果 50](#_Toc495246329)

[2.20 6.3 非功能性测试 54](#_Toc495246330)

[2.21 6.4 测试结论 54](#_Toc495246331)

[2.22 6.5 本章小结 54](#_Toc495246332)

[第7章 总结与展望 56](#_Toc495246333)

[参考文献 57](#_Toc495246334)

[致 谢 58](#_Toc495246335)

# 绪 论

## 选题背景及意义

随着计算机技术和互联网技术的迅猛发展，各种相关的应用如雨后春笋纷纷涌现，颠覆了传统的方式，大大影响了人们的生活和工作方式。在这种新的发展形势下，社会对教育提出了更高的要求，计算机基础课程已经成为中国高校学生的必须课程，不仅涉及到计算机相关专业的学生，也是非计算机专业学生的公共基础课程，旨在现有技术背景和社会需要下，培养学生应用计算机来认知和解决问题的能力，为学生熟悉信息化社会中的各项基本应用，并适应社会的需要以及未来的发展奠定良好的基础。现代教育存在的一个普遍的问题是学生数量掌握了理论知识，而用已掌握的理论知识应用于实际问题的能力还很薄弱。因此如何有效地考察学生对知识的掌握程度和运用能力是高校和计算机基础课程教师关注的重点问题。相比于其他课程，计算机基础课程更加侧重于学生的动手实践能力以及举一反三的能力，因此采用在线考试的方式是一种必然选择。传统的计算机课程考试方式存在很多弊端，主要体现在以下几点：

一、采用纸质试卷的形式，仅仅能够考察学生对理论知识的掌握程度，不能考察学生的编程和实践能力，无法真实衡量学生对课程掌握的真实程度。

二、大量考试前的准备工作。教师需要事先准备准备考题并进行试卷印刷，浪费了大量的资源。同时需要安排考场、监考人员以及考试时间。

三、需要人工进行阅卷，耗费大量的人力。对于主观题，不同教师可能有不同的评分标准。为了了解学生对不同知识点的统计程度，教师需要统计不同试题的得分以及评判整个试卷的难易程度以便调节后续考试的难易程度。

四、考试不公平性。由于不同考场的纪律以及管理程度不同，考试存在不公平的现象。

因此我们需要采用在线考试的方式来取代传统的考试方式，将信息化技术应用于教育领域，改良传统教育模式，提高教学效率、教学效果以及科研管理水平。在线考试能够有效地解决传统考试的问题，大大简化了考试的过程，考生能够随时随地参与考试，还可以保证考试的公平性，同时方便了教师管理考试内容和评估考生成绩。

## 开发目的及意义

本论文的主要研究内容是通过借助计算机技术来开发一种在线考试系统来代替传统纸笔的考试模式。目前已有的在线开发系统都是为了方便考核学生的学习效果而开发的，但是这些平台大多针对以客观题为主的课程的考核。对于计算机课程的考核更加侧重于学生的编程实践能力而不仅仅局限于理论知识的考察，而本文所涉及到的在线考试系统的设计与实现目的，正是为了适应计算机课程的需要，不仅提供理论知识的考察，还有对学生的编程能力进行考察。

## 国内外发展现状

随着时代的发展，考核方式也发生了一些变化。过去高校的学生规模相对较少并且计算机基础课程的考察比较简单，传统的考试方式的局限性并不明显。随着高校招生规模的逐渐扩张以及计算机课程成为公共必修课程之后，参与计算机基础课程的学生人数日益增多，传统考试在计算机基础考试上弊端日益明显。为了适应新的发展需要，在线考试逐渐取代传统考试方式。相比传统考试方式，在线考试方式能够大幅度减少出题、监考、评阅、统计等考试工作[1] ，大大提高了学校的工作效率。早在上世纪70年代，美国考试委员会率先从事计算机模式考试的研究工作，并于80年代研发了第一个在线考试系统。目前专业提供在线教育以及测评的机构越来越多，其中美国著名的考试机构思而文学习系统有限公司专门为各大机构提供在线教育以及考试的服务，许多授证机构以及专业人员的认证考试都依托于该机构。相比之下，国内的在线考试起步相对较晚，从2004年开始，大学英语四六级以及计算机等级考试开始从人工考试方式向计算机考试方式转型。直至2006年，在线考试系统已经发展为考试的主流形式。截止到2016年，全世界141个国家，有超过25中不同语言通过在线考试的方式提供了2400多种考试，涉及到专业考试、学校教学效果检测以及企业考核等多个领域，考试能够随时随地举行，受时间地域的限制越来越小，逐渐成为一种学习方式，方便查漏补缺[2] 。

计算机基础课程考试主要包含两大部分，第一部分是理论知识考察，主要通过选择题、判断题等客观题。第二部分是编程部分，主要是通过写代码的方式进行考察。目前比较成熟的在线考试系统都是由专门的机构开发，虽然能够适合很多课程的考试需要，但是对于学校而言，采用这类的考试系统会存在很多问题。第一，学校考试相对比较频繁，考试费用或者软件开发费用过高。第二，这类系统大部分是基于标准化考试形式，能够很好的完成客观题的考核，难以对主观题进行判定，尤其是对计算机基础类课程的考核存在很大的限制。

## 论文研究内容

对于计算机基础课程，不仅要考核学生对理论知识的掌握情况还要考核学生编程实践能力。而目前已有的在线考试主要是针对客观题的考核，在客观题的评判上存在很大的问题，因此本文主要研究如何借助计算机以及互联网技术来设计和开发一种针对计算机基础课程的在线考试系统。为了更好的了解以及实现该系统，首先需要分析计算机基础课程的考核内容以及考察形式，确定系统的实际功能需求并进行业务建模。然后通过用例图以及数据流图对系统需求进行进一步的分析。最后针对相关的需求进行整体和详细的设计并确定相关功能实现所需的技术以及相关的算法。根据实际调研的需求，确定整个系统的架构，并完成学生、教师以及管理员的权限与角色管理、题目录入、考生考试验证登陆、智能组卷、防作弊、考试计时、试卷评分、试后统计分析等功能模块。

## 论文组织结构

本文主要是介绍计算机基础的在线考试系统的设计与实现，并从需求分析、系统设计、实现等方便依次展开。论文整体结构如下：

第1章，绪论。通过对比在线考试与传统考试的差异，分析在线考试的有点以及在线考试的意义。接下来分析了在线考试在国内外的研究现状，并确定本文的研究内容以及相关的功能模块。

第2章，相关技术。介绍在线考试系统的理论基础，以及该系统所涉及的技术。主要是C/S和B/S体系结构的相关介绍，以及本系统涉及到的集中关键技术，最后是本系统的开发环境、采用的开发语言以及相关的技术、数据库等内容。

第3章，需求分析。通过用例图以及数据流程图对系统进行需求分析，包括功能性需求以及非功能性需求分析，最后对系统的可行性进行分析。

第4章，系统设计与实现。首先是系统功能的分析，整个系统的设计框架以及设计原则，其次数据库的设计，最后是系统关键模块的实现。

第5章，总结与展望。对本文的工作进行总结，主要是本文所涉及的系统的优缺点，以及有待改进的方面以及后续的工作。

# 关键技术

在上一章中，主要叙述了在线考试系统相比传统的纸笔考试方式的优越性，并分析了现有的在线考试系统的局限性，从而提出了针对计算机课程的在线考试系统。本文将会利用javaEE的技术来开发针对计算机课程的在线考试系统，因此需要对该系统使用到的相关技术进行说明，了解为什么使用MVC设计模式，B/S架构，以及SSH（Struct、Spring、Hibernate）框架。

## MVC设计模式

MVC模式[3] 为model-view-controller的缩写，与80年代提出的一种软件设计典范，包含模式、视图和控制器三个逻辑组件，将业务逻辑、数据、界面显示进行分离的方式来组织代码，这三个逻辑组件以松耦合的方式协同工作，极大提高了应用程序的可扩展性和可维护性，并受到广泛的应用。后来被Sun公司推荐为JavaEE平台的设计模式，成为面向对象程序设计语言遵守的规范。

### MVC组成部分介绍

MVC模式按照用户输入、外部世界建模以及可视化反馈进行划分，并分别使用模型、视图和控制器来进行处理。下面依次对三个部分进行描述。

模型(Model)用于处理数据，视图部分会发出一些数据请求，模型会根据请求作出回应。当控制器发出请求该别状态时，模型部分就会作出相应的处理。整体来看，模型部分是MVC的核心。

视图(View)是与用户与应用程序交互的界面。负责展示模型返回的数据同时也能够接受用户的输入，但并不能够处理独立处理数据。

控制器(Controller)是模型与视图之间的连接器，负责从不同视图接受请求，并转发给对应的模型，并根据模型返回的结果选择视图来相应用户的请求。

### MVC交互流程

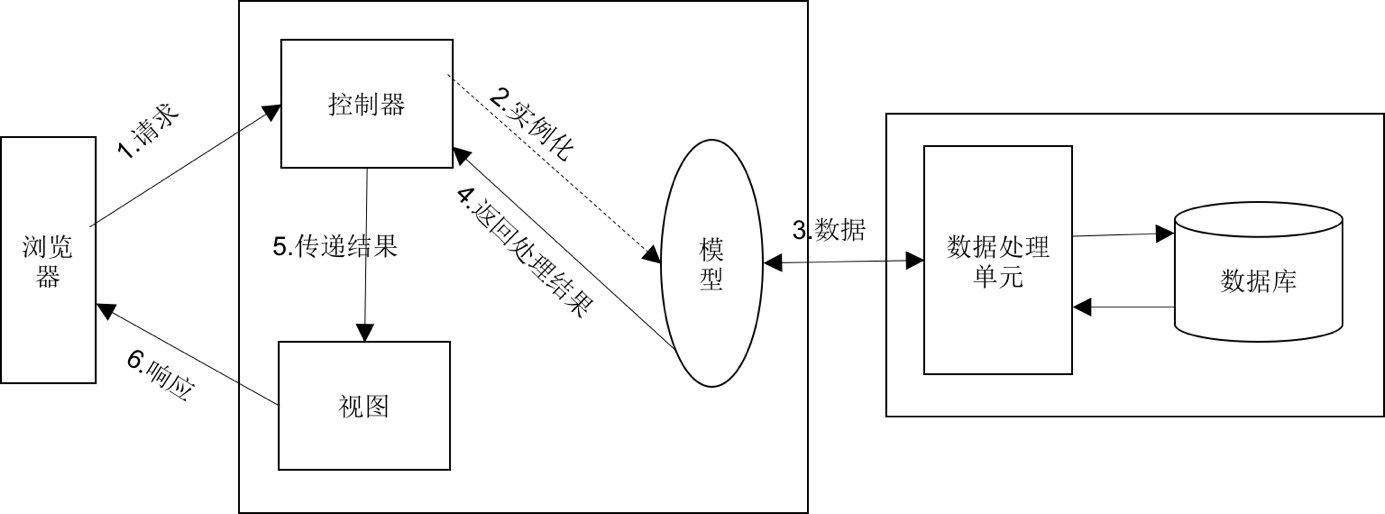
MVC中的各个组成部分之间的交互流程如图 1所示，用户发出请求，控制器根据请求来选择合适的模型进行处理，然后模型根据用户的请求进行相应的业务逻辑处理，并返回用户请求的数据，最后控制器将返回的相应数据传递给对应的视图，并由视图部分将数据最终呈现给用户。

图 1 MVC标准化流程

### MVC模式的优点

MVC模式能够将视图与逻辑处理单元分离，能够非常容易的改变视图而不必改变后端的逻辑处理，并且能够实现视图的动态添加和删除，同时不同模块的分离和单独开发，增加了程序的可扩展性和可维护性[4] 。控制器能够连接不同的视图和模型，以适应用户的需求，增加了程序的可重用性。目前大部分web应用程序是基于过程化语言进行开发，将页面与数据库访问的代码混杂在一起，大大增加了维护的难度。

## SSH集成框架

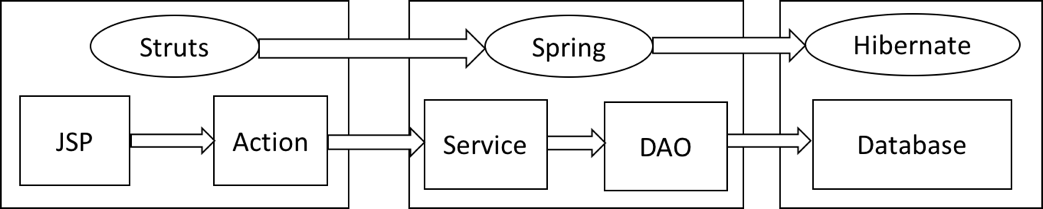
SSH是一种集成框架，包含Struts+Spring+Hibernate三种框架，是目前比较流程的一种Web应用程序开源框架。整个框架结构分为表现层、业务逻辑层、数据持久层以及域模块层四层结构，使得整个框架结构清晰、职责分明且易于开发和维护。SSH集成框架中Struts用于搭建这个应用的基础架构，负责MVC中各个部分的分离。Struts负责业务跳转，Hibernate负责数据持久化部分，而Spring则是负责管理Struts和Hibernate，Struts，Spring，Hibernate之间的交互流程如下图 2所示。在表示层中，使用Jsp页面来与用户进行交互，负责接受用户请求以及接受响应，然后Struts根据配置文件Struts-config.xml选择合适的Action来处理请求。在业务逻辑中，Spring的loC容器负责向Action提供一些业务支持，同时采用一定的技术来提高系统的性能。在持久层中，Hibernate负责对象与数据库之间的映射，完成数据库处理操作。

图 2 SSH集成框架结构

### Struts2框架

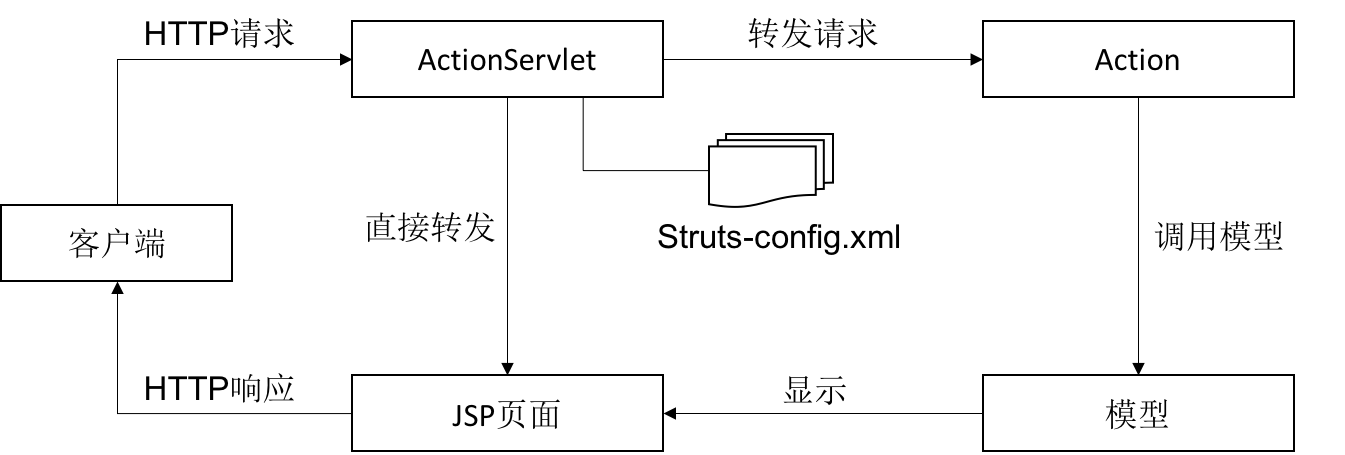
 Apache Struts是一个可扩展的免费开源框架，用于搭建企业级的java web应用[4] ，图 3是Struts1的框架结构。Struts1框架是基于MVC的视图层部分的框架，利用Java Servlet 和JSP构建Web应用。Struts结合了JSP、JavaBean等技术，有效地实现了视图与业务逻辑代码的分离，大大降低了系统的复杂度以及不同业务之间的耦合程度，增强了代码的可维护性以及可扩展性，同时提高了开发效率，Struts组成部分以及各部分之间的交互过程。

图 3 Struts1 框架结构

Struts2框架[5] 是在Struts1和WebWork框架基础上发展而来的，Struts2继承了WebWork的体系结构和运行流程，同时继承了Struts1的优势，并在此基础上提供了很多新的功能，比如智能默认配置、annonation等，大大减少了Struts1的XML配置。

Strut2框架包含三个部分：核心控制器、业务控制器以及业务逻辑组件，详细结构如图 4所示。

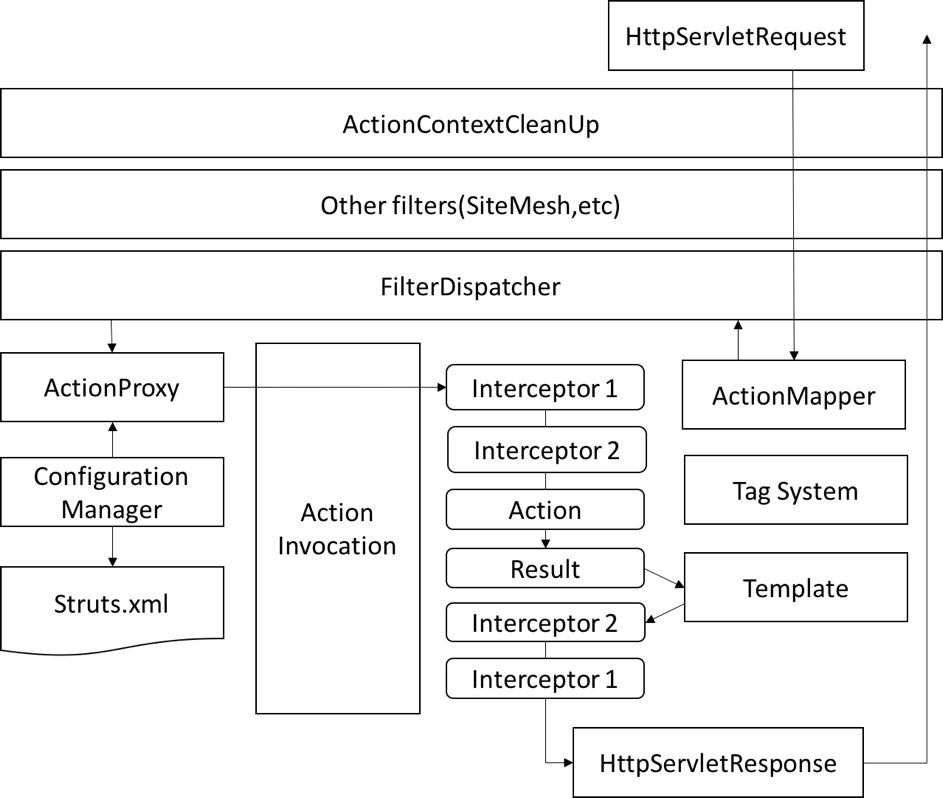


图 4 Strut2 框架结构

首先客户端提交一个HttpServletRequest请求，接下来这个请求要经过一系列的过滤器，顺序依次是ActionContextCleanUp，Other filters，然后交由FilterDispatcher处理，并通过ActionMapper确定是否需要调用某个Action来处理该请求，如果确定能够某个Action能够处理该请求，则FilterDispatcher将该请求交由ActionProxy，ActionProxy通过配置管理器来读取struts.xml配置文件来找到能够处理该请求的Action类。在找到对应的Action类之后，ActionProxy创建一个ActionInvocation对象，并通过该对象使用代理模式来调用对应Action类的实例，同时会根据配置文件为该Action类加载所需的拦截器。最后创建的Action实例会对客户端发出的请求作出相应，ActionInvocation根据struts.xml配置文件返回处理的结果。

### Spring框架

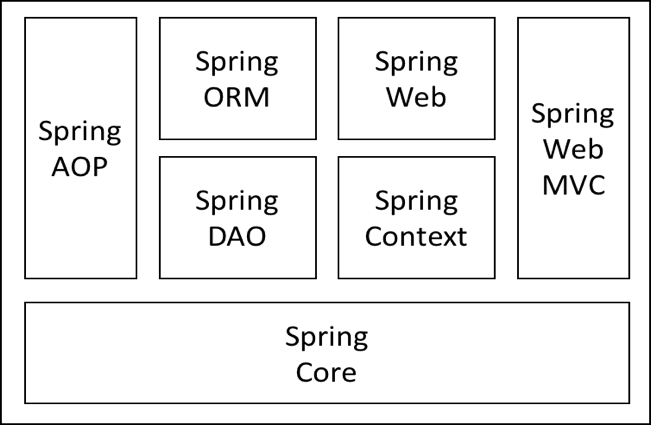
企业应用程序的业务逻辑比较繁杂，开发的复杂度比较高并且后期维护起来也比较困难。Spring是一个开源的框架，主要就是通过分层的概念来降低应用程序的复杂性，对JavaBean的生命周期进行管理[6] 。Spring构建在Spring Core之上，用于创建、配置和管理JavaBean。Spring的核心思想是IOC和AOP，其中IOC为依赖注入，避免了手动显式地创建对象，而是利用java反射机制，在程序运行过程通过Spring的配置文件动态地创建对象以及调用对象；AOP是面向方面编程，是一种编程技术，将影响多个类的行为封装到可重用的模块中。Spring框架包含7个模块，如图 5所示。

图 5 Spring框架结构

核心模块定义了创建、配置和管理对象的方式，提供了整个框架的基本功能；上下文模块是Spring的配置文件，向Spring提供上下文信息；AOP模块直接将面向方面编程功能集成到Spring框架中，从而使得Spring所管理的任何对象都能够很好的支持AOP，同时为这些对象提供事务管理服务。DAO模块提供了访问数据库的方式，免去了繁琐的数据库访问控制以及异常数据工作。ORM模块为各种数据库对象映射的框架和工具提供了支持，使得Hibernate、iBatis能够集成到Spring框架中。Web MVC模块提供了基于MVC模式的Web应用程序开发的支持。

### Hibernate框架

在面向对象软件开发过程中，关系数据库与对象之间的协同工作是非常耗时的，由于范型不匹配的原因，如何将对象中的数据与关系数据库之间建立映射关系是一个非常困难的问题，而Hibernate则是为解决这个问题而产生的对象关系映射框架。Hibernate框架将Java对象与数据库中的表以及Java数据结构与SQL数据格式进行对应，同时提供了数据库查询以及获取等操作。Hibernate框架的设计目标就是降低开发人员在持久化数据工作中的时间，减少通过SQL和JDBC处理数据的工作。

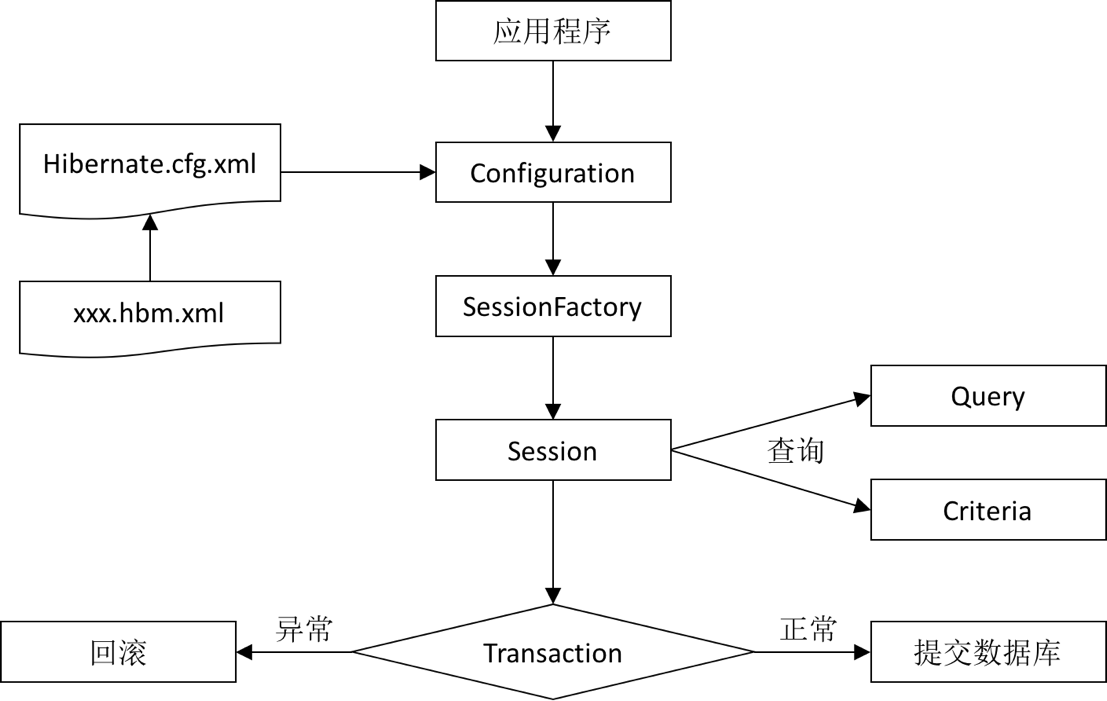
Hibernate将数据资源映射为一个或多个Java对象，将数据库中的各种操作都封装为Java对象中的函数，大大减少了JDBC以及SQL代码的编写，使得开发人员将更多的精力花费到业务逻辑单元部分[7] 。Hibernate将业务逻辑部分与数据库操作部分的代码分隔开，大大降低了代码之间的耦合程度，并且当数据库发生变化以及数据迁移时，仅需要修改配置文件，提高了应用的可扩展性以及迁移性。

图 6 Hibernate框架结构

图 6显示了Hibernate框架的整体结构以及各个部分之间的流程，包括六大核心结构以及两个主要的配置文件。其中Configuration接口负责读取Hibernate配置文件，并创建一个SessionFactory对象，SessionFactory为Session工厂，负责生产Session对象。Session对象用于完成对象的持久化操作，例如数据库的查询等操作，同时可以产生Transaction对象，该对象负责将操作的结果提交到数据库中。

## MySql数据库

MySQL数据库是一个开源的关系数据管理系统，由瑞典MySQL AB公司开发，目前已经属于Oracle公司[8] 。MySQL借助结构化的查询语言（SQL）来进行管理，是目前比较流行的Web应用的数据库。

MySQL以其高性能、成本低、开源等特点，能够帮助我们有效的进行数据处理，大大提高了工作效率。MySQL数据库的优势可以总结为一下几个方面。

第一、支持多用户、多线程。MySQL的内核采用多线程编程和C语言编程，能够灵活地为用户提供服务，同时避免了占用过多的系统资源，能够充分利用CPU资源。

第二、完全开源。用户完全可以根据自己的需要修改MySQL，并且提供了接口以适应不同语言的连接操作。

第三、用户友好，并且提供了灵活并且安全的权限和口令认证系统。当客户端与数据库建立连接时，两者之间传送的内容以及消息都是经过加密的，并且支持主机认证。同时提供多种客户端以适应不同用户的需要，包括界面、命令行等。

第四、适用于各种操作系统，包括Linux、Unix、Windows以及OS等平台，不仅可以用于服务器还可以用于小型电脑甚至个人电脑。

## B/S体系结构

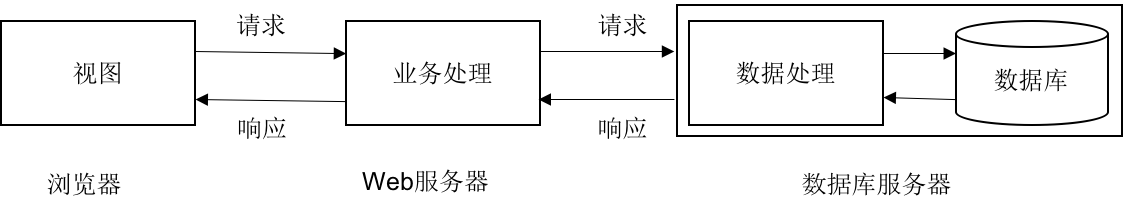
B/S体系结构的全称是浏览器/服务器结构，它是在C/S（客户端/服务器）体系结构的基础上发展而来的，并采用HTTP作为传输协议[8] 。用户通过浏览器向服务器端发送请求，服务器从数据库中获取数据并按照请求进行业务逻辑处理，然后服务器端将处理后的结果返回给服务器，最后浏览器解析返回的结果并将结果呈现给用户。相比C/S结构而言，B/S结构不需要额外开发客户端软件而是直接使用浏览器，大大提高了可扩展性以及可维护性，同时降低了开发和维护的成本。

图 7 B/S 体系结构

B/S体系结构包含三个部分，浏览器、web服务器和数据库服务器，如图 7所示。首先是浏览器部分，承担视图显示部分的工作。负责接受用户的输入并封装为Http协议的格式，同时也负责将服务端返回的结果显示给用户。然后是Web服务器部分，主要承担业务流程以及业务处理逻辑任务。浏览器通过Http协议将用户的请求发送给Web服务器端，Web服务器接受到请求之后，与数据库服务器建立连接，并通过SQL语句的方式提交数据处理请求。等数据库服务器端执行完数据库请求之后，将执行的结果再返回给Web服务器端，最后Web服务器端将结果传送给浏览器。最后是数据库服务器部分，主要完成数据处理任务，执行Web服务器发送的SQL语句的数据处理任务，从而完成对数据库中数据的增删改查等操作，并将执行的结果返回给Web服务器。

## 本章小结

本章节介绍了与项目相关的基础知识，包括MVC设计模式以及B/S体系结构。同时介绍了项目开发所要使用的集成框架，以及该框架下Struts2，Spring，Hibernate三个框架的结构以及工作流程，并简要介绍了项目所使用的数据库以及相比其他数据库的优势，也间接说明了选用该数据库的主要原因。从第三章开始将介绍本文的工作内容。

第3章 需求分析

## 3.1 系统总体需求

3.1.1 系统业务需求

3.1.2 系统用户需求

根据系统研制的用户需求，在开发过程中严格遵循以下原则：

1. 突出重点，注重实用
2. 软件架构合理，开发方便
3. 层次清楚，符合标准
4. 安全稳定，简便快捷
5. 完全按照平台要求开发，尽力增强系统适应性与可拓展性

本管理系统的使用者一定是基于iOS操作系统的移动智能终端的用户，或者能够熟练的使用智能终端操作系统和各类软件，具备基本的终端操作常识。

## 3.2 系统功能需求

通过实际需求调研结果，本工程监控系统的基本功能需求可以大致概括为以下这些模块：用户所在地定位、个人日报记录、网络日报记录、正常或者按日期查看员工日报记录、设置我对谁可见，查看谁对我可见、查看对我可见的指定用户日报记录、后台用户注册、查看用户个人信息、配置网络服务器地址、修改网络及个人日报记录、尽量降低网络要求、公告与消息中心、项目权限设置、审核批示、财务管理与财务预警等，通过多次调研，完成了整个项目的需求分析与系统设计模块。结果显示，以上描述的功能模块比较符合实际的市场需求，功能模块以及系统的使用方法均可参见系统设计部分内容，十分全面且易于理解。

3.2.1 系统的登陆退出与用户的角色管理模块

用户正确配置完成服务器地址后，就可以进行系统的登录操作，输入用户名与密码并点击登录以进行网络验证，在验证的过程中将会识别用户的角色，判断当前账号是否为相应客户端的有效用户，因为客户不能登录企业版系统，而企业员工也无法登录客户版系统，以达到系统角色划分的目的。根据实际的市场调研结果，总结出的系统角色及其相应操作权限的用例分析如图3.1所示。



图3.1 工程监控系统角色用例图

当用户验证通过后，则会将该用户信息保存在KeyChain的专门区域内，如果用户下次登录则无论网络连接是否正常，均可以直接登录进入系统查看已存的本地记录。但是一旦用户退出当前登录则会将KeyChain内容清空，并且直接退出当前登录状态，回到系统的登录界面，用户需要重新输入账号密码，联网验证正确后才能够重新进入系统。

3.2.2 本地与网络记录管理模块

本地记录的形成需要拍照与定位功能相结合，在拍完照片之后将相应的记录基础信息填充完毕后，存储所拍照片于自建立的文件中，并将记录相关文本信息存于本地数据库中，然后形成本地记录后加载到界面展现，由用户填写其他信息，然后选择记录上传、保存还是删除等操作。

网络记录则是从服务器端取到用户已经上传的日报记录，用户可以在此页面中查看已上传的记录信息是否有差错，如果发现除基础信息之外的填充信息有误的话，可以随时在客户端修改相应内容，但是网络记录不允许用户删除，其已作为生效的日常记录保存在服务器端。

3.2.3 日常记录管理与日报记录查找模块

日常记录管理是用于查看对我可见员工已上传的日报记录的方便形式，会在页面上面呈现出按照时间由新到旧的顺序上传的记录项，可以点击进入详细界面，并进行审核批示，方便用户查看相应人员的工作情况以及业务进度，而这种方式对于使用者来说有时又不是十分方便，所以日报记录查找功能模块随之产生。记录查找可分为按人员查找与按日期查找两种方式。

1. 按人员查找

按人员查找功能名称为某人记录，用户可以在这一页面看到所有对自己可见人员的基本信息，比如用户头像和用户姓名等，然后轻触屏幕上的某一用户，就可以进入新界面来查看该用户最近的日报记录，便于了解其最近业务情况，方便使用者进行进度判断与决策。

1. 按日期查找

按日期查找功能名称为日期查找，实际可以划归为记录管理的子模块，用户可以选择性地查找某一特定日期的日报记录，方便于用户进行日报记录的针对性检查与相应工作的战略部署等，也是不可或缺的实用需求。

3.2.4 日报记录详细信息查看与修改模块

日报记录无论在本地界面、网络界面、日常记录界面还是查找后的记录显示界面，均只是日报记录的简略介绍，如果需要查看该记录的详细信息，则需要用户能够进入到该记录的详细信息界面，记录的详细信息应该包括记录的基本属性与附加内容，如日报的拍摄时间、拍摄地点、记录图片的放缩等，附加属性则是该记录是否有评论存在，如果有则需要展现评论列表。

3.2.5 地理定位模块

在苹果开发过程中，地理位置的确定是比较常见的功能，而且苹果的定位方式会由系统根据实际情况，在WIFI、基站与GPS卫星定位之间进行合理的选择，但是由于iOS所取到的经纬度坐标是基于地球坐标系（WGS-84），而国内由于信息安全的需要对于坐标系进行了偏移，也就形成了所谓的火星坐标系（GCJ-02），所以在使用高德的SDK进行定位的时候，经纬度的“纠偏”是一个难点，不过通过拟合出来的“纠偏”数据库已成功解决定位偏移问题，而且在开发过程中将地理位置获取的部分封装成了一个完整的类，只需要新建地理位置类的对象或者调用类方法即可实现定位相关功能，十分方便。

3.2.6 审核通知与公告中心管理模块

系统每隔一段时间需要访问服务器一次，以询问是否存在对用户可见却未查看的新公告或者新消息，当存在的时候会在系统上出现相应的提示，提醒用户及时查看与自己相关的最新动态，这是形成消息中心与公告中心的基础所在。因为在工程进行过程中，有些时候是需要在用户之间直接进行交流审批的，所以审核批示的功能将会允许一方对某些日报记录进行评论，当另一方查到有新内容的时候，可以及时提示其查看审核结果，以便做出相应决定。而公告则是针对全体系统用户的，方便团队之间进行一些重大事情的集体通知。

3.2.7 财务记录管理模块

在工程项目管理系统之中，财务管理部分必不可少，根据实际需求需要完成根据项目进行记账的基础功能，具体业务流程见图3.2所示。系统从服务器端获取到当前项目的信息之后，显示在手机客户端之内，用户可以选择项目与类别，在其中查看所有对自己可见的已有花费记录，以及项目的预算与余额情况，并且可以直接填写必要信息后添加新的财务记录，当某一项目的实际花费已经逼近预算，或者超过既定金额警戒线的时候，客户端界面上应标红信息，并进行相应提示，方便用户随时了解项目的花费程度。



图3.2 财务管理记账功能业务流程图

3.2.8 签到与签退记录管理模块

对于用户的考勤与绩效统计也需要由本文描述的系统进行实现，这里将会充分利用之前在地理位置模块提到的已封装完善的地理信息类，然后辅以安全控制部分的设备唯一标识作为用户标记，防止用户考勤作假，允许用户每天签到与签退各一次，并且附带当时的地理位置与备注信息后上传到后台服务器，以方便之后进行考勤统计与绩效管理，然后交由后台进行处理，并在WEB端进行绩效的统计与查看。

3.2.9 权限控制与安全处理模块

关于权限控制，在系统中用户可以设置自己的日报记录对谁可见，以及自己想要接收哪些项目的新消息提醒，对于整体角色的分工合作有显著效果，该业务的具体流程如图3.3所示。



图3.3 系统用户的可见性权限相关示意图

手机端系统需要绑定设备信息，唯一标识用户，但是iOS开发将手机端获取设备唯一标示码，即等价于在安卓平台进行移动开发时获取IMEI码的接口禁用了，所以只能采取另一种处理方式，使用基于苹果自带的KeyChain（钥匙串）功能，也是关于记住密码的主要实现技术，将设备的IDFV，即identifierForVendor的值经过Base64编码后，形成一串不可读标识码，存在KeyChain之中作为该设备的唯一标识，这样在传输过程中也不会因为数据被偷走而导致直接被对方破译这种标识，属于一种基础编码加密技术，但是效果上能够大大提升系统的安全可用性。

3.2.10 更多设置功能模块

更多设置里面应该包括前面介绍的权限设置与签到签退等部分功能模块，但由于该部分功能使用相对比较频繁，在实际开发过程中会根据实际情况将部分功能显示在主体界面之中，所以在前面进行了详细的需求分析介绍，下面将介绍一些基础的功能模块。

(1) 个人与开发者信息模块

需要建立个人信息模型，从服务器端获取当前登陆用户的基本信息，包括用户名、账号、性别、联系方式等，在客户版子系统中，还有项目经理的角色，客户可以直接点击电话按钮进行电话自动拨号，方便联系项目相关负责人。开发者信息模块则列出了开发人员的基本信息以及开发公司的基本信息与联系方式，起到宣传与介绍的作用。

(2) 网络设置与版本更新提醒模块

用户在登陆之前需要配置一下服务器地址，以便于用户登陆到自己所在单位的服务器之中，进行相应的记录上传与查看，该服务器地址将被保存记录下来，不用二次配置，但是网络设置模块在登陆后依旧可见可操作，用户可以随时切换服务器地址注销账号重新登陆。当用户登陆之后，第一件事情是进行系统版本的判断，如果该系统有更新版本则弹出提示，提醒用户到App Store上面进行新版本的下载更新。

## 非功能性需求

1. 开发硬件环境描述

机型：MacBook Pro 版本：MBP55.00AB.B03

内存：4GB 1333 MHZ DDR3 处理器：Intel Core i5 2.3GHZ

1. 开发软件环境描述

操作系统：Mac OS X 10.10.3；

开发环境：Xcod6.3.1(iOS 8.3 SDK)

1. 适用机型以及设计约束

适合iPhone 4s以上，并装有7.0以上版本iOS的所有iPhone设备，本系统基于iOS手机开发，一切的设计与实现全部遵守iOS手机应用开发的有关规定。

1. 性能需求

本系统是基于网络的日报信息化管理平台，所以网络的稳定性是系统运行的关键。建议使用WIFI或者3G移动网络，保证网络速度的前提下提高用户体验。另外服务器的处理速度要符合用户习惯，在网络带宽足够的情况下，避免因为服务器延迟造成浏览延迟。保证稳定提供服务的前提下，尽量提高速度。

1. 安全性需求

客户端的操作安全性完全建立在保护用户基本信息的基础上。本文在设计之初为避免客户端预留过多不安全接口，以防止其他程序读取本系统中存储的私密记录，或者通过接口篡改进行恶意破坏，并窃取用户个人信息等非法行为。

1. 错误处理

实时捕捉到用户的错误操作，并与服务器交互，服务器将返回客户端反馈的错误信息，并及时在用户的手机客户端系统上作出有效反馈，不会出现未知错误而导致应用异常闪退或者系统崩溃。

## 3.4 本章小结

本章从系统的业务需求分析、用户需求分析、系统功能需求分析进行了详细的介绍，并按照系统的功能性与非功能两方面描述了系统的各项需求，通过相应的需求分析模型，详细的描述了系统最终所需要实现的所有功能，为后续即将开展的系统设计阶段奠定了良好的基础。

第4章 系统设计

## 4.1 系统概要设计

4.1.1 系统主体模块

基于iOS的工程监控系统，在实现过程中将使用Cocoa框架，以及MVC的设计模式，将视图层、控制层与模型层相互分离，使之具有更好的性能与可拓展性，根据实际需求与需要实现的目标相结合进行分析，可以概括的将本系统的主体功能模块大致分为九大部分，如模块图4.1所示，在系统的功能实现的过程中，图4.1所展示的内容将会全部进行详细地介绍，但是在主功能模块之下，还存在很多的子功能模块，在本文中对于一部分会进行简单介绍，但是一些简单而基础的部分功能的设计与实现过程将不会提及过多。



图4.1 系统总体功能模块图

4.1.2 系统架构分析与总体设计

结合系统的主体功能模块设计图，综合考虑后得到本系统的设计目标与需要实现的具体功能的逻辑关系如设计图4.2所示。客户专用版子系统的功能大多数是以下功能的组合与二次开发，该子系统的设计开发中更为重要的是对一些细节问题考虑得是否全面，所以在此对于子系统的功能结构等内容不再进行过多介绍。另外，用户记录查看的上下拉刷新将会是查看新旧记录的通用方式，之后在系统实现部分会进行介绍



图4.2 系统总体功能架构与设计图

## 4.2 功能模块详细设计

4.2.1 系统的登录退出与用户的角色管理模块详细设计

系统的启动登录与退出登录模块详细设计如图4.3所示，按照程序的启动过程以及界面展现顺序的方式来呈现，关于各级界面所代表的模块在下面会进行更为详细的设计介绍，这里只是用以辅助说明本模块的设计思想。



图4.3 启动登录与退出登录模块示意图

在NavigationController之中以PUSH与POP的方式实现系统的进入与退出，实际上是一个压栈与出栈的过程，这样的设计会使得用户在通过验证登录到系统中之后，在进行退出操作之后将系统当前所占用的内存空间全部释放，避免出现内存溢出与内存泄露等严重问题。

用户角色管理将会严格按照需求分析中的用例所展示内容进行设计区分，系统创建用户的时候，可以选择自己的角色，产生相应的角色标识，附带不同的操作权限。

4.2.2 本地与网络记录管理模块详细设计

1. 本地个人记录管理

本地个人记录的形成依靠拍照与地理位置信息的获取，当用户打开本系统所提供的相机接口后，地理位置信息就将进行即时更新，用户可以选择拍照或者放弃，如果放弃还可以选择是退出拍照状态，还是不保存已拍照片（此时还未形成日报记录），如果选择采用已拍照片，则系统会将拍照时间、当前地理位置信息以及拍照者姓名等基本信息与照片名称相关联一起保存在本地数据库，同时将刚拍摄照片按照压缩率为0.1的比率进行压缩后保存在本地自生成文件中，在应用外并不可访问，有效防止用户数据造假。之后用户可以在事先通过网络请求得到的可用项目列表中选择自己当前所在项目组，填写完成基本备注之后，即可选择上传还是先保存在本地，上传后则形成一条有效的日报记录，并存在于团队服务器中，未上传与已上传记录均可以根据自己意愿在本地进行删除，并且本地支持多账号登录，即不同用户在同一手机登录均可查看到自己在本机的本地记录，但是由于设备唯一性的原因，用户之间的绩效考勤记录并不能够通过替代操作来完成，增加了系统的安全性与可辨别性。

1. 网络个人记录管理

网络个人记录的与本地个人记录最大的区别在于网络记录并不可由本人在客户端系统随意删除，因为其已经是一条有效记录，将会用于多方面考量与审核。不过用户如果发现某一条网络记录的备注等自己可添加的信息有误，也可以直接进行网络修改，方便用户纠正并避免再次上传的不便。而且网络记录是可以被可见自己的用户进行审核批示或者评论的，自己也可添加新的回复，至于新消息的提示将会由新消息中心功能模块进行处理并显示。

4.2.3 日常记录管理与日报记录查找模块详细设计

日常记录是对自己可见人的网络记录的显示，默认显示的界面是按照时间由新到旧的排序记录，即团队效率的首次显示页面，不过用户可以根据自己的需求，在对自己可见用户列表中选取指定用户，查看其最近的所有相关记录，对应其中的某人记录功能模块，使用者还可以按照时间进行某一天的记录查询，即日期查找功能模块，查看方式灵活多变，对于使用者来说也十分方便。不过团队效率记录并非使用者本人生成，所以功能设计上面无论是删除、更改、增加均不可操作，只有查看与对相应记录进行审核批示的功能可用，符合系统逻辑与真实需求。

4.2.4 日报记录详细信息查看与修改模块详细设计

日报记录在TableView上面的每个Cell中显示的应均为简略信息，用户想要进行详细信息的查看需要点击进入二级界面，在这里可以按照本地、网络与日常三种情况进行所开放的相应操作，操作权限设计为个人网络记录可本人修改记录内容，但是他人网络记录用户只可以查看与评论，照片记录上面添加了手势识别，用户可以点击图片进行图片放大操作，便于用户更为详细地查看日报主要信息。

4.2.5 地理定位模块详细设计

地理定位模块是本系统中极为重要的一部分设计内容，由于iOS本身提供的定位机制是按照自己的既定规则在基于基站、基于WIFI与基于GPS卫星三种方式间进行选择，其所使用的是标准的地球坐标系（WGS-84），而国内因为信息安全的需要对于坐标系进行了偏移，也就形成了所谓的火星坐标系（GCJ-02），所以纠偏也是一项十分重要的工作，使用经纬度转换的拟合数据库进行相应的纠偏，然后将纠偏后的经纬度坐标传递给已封装进系统内的高德SDK进行处理，通过逆地理解析编码得到用户当前所在的较为精确的位置，同时将地理位置的获取整体封装成两个类AddrInfo与AddrObj，两者组合来快速方便地使用定位相关功能，两个类的具体设计会在之后详细说明，其实现过程的时序图可参看图4.4所示内容。



图4.4 地理定位模块交互过程时序图

4.2.6 审核消息与公告中心管理模块详细设计

系统为了能够及时获取到新信息与新评论，采用客户端系统每隔一定时间向服务器询问一次的单一轮询方式进行交互，每次服务器会检查下针对当前用户是否有未读消息或者未读公告，并将对应结果返回到客户端系统，如果二者均无，则客户端系统无任何变化，照常工作。如果具有新内容未查看，则系统会根据返回结果对新消息或者新公告进行相应的提醒，均为图标变化与振动相结合的方式，提醒频率符合用户需求。

4.2.7 财务记录管理模块详细设计

财务记录类将会包含项目名、项目类别、处理者、录入者、描述、时间、单价、数目以及总价等基本属性，这些属性是用户在查看与添加记录的时候所必须展现与填写的内容，按照服务器与客户端的最初协商结果，在两种备选实现方式之间进行了实际测试抉择，第一种是服务器将财务相关的全部信息返回以响应客户端的请求，然后由客户端进行本地分类处理后显示在界面上面，并由用户根据项目名称与花费类别组合在本地已分类信息中进行查找并显示，同时发送网络请求获得当前项目组的预算与余额，并根据是否超过财务预警值来决定是否进行警告提示，超警戒线则标红，否则正常显示；第二种方案是用户根据项目与花费类别组合同样发送网络请求，然后将返回内容呈现在界面之上，预算余额实现方式选用一样方案。这样做前者拥有的优势在于数据加载速度很快，并且不会有延迟，但是劣势也很明显，系统首次处理的速度相对较慢，如果在数据量巨大的情况下，性能应该会有所降低，第二种方案的优势则是相对应的数据量大时会比前者处理速度快，因为不涉及到其他项目的分类，但是问题也随之产生，网络请求会更加频繁，对于服务器与系统的稳定性都有比较大的负担。所以，最后的折衷方案定为采用第一种方式，但是对数据量进行限制，当超过时间范围的时候需要用户使用WEB端查看，这样对于手机用户来说也是十分合适的解决方案。用户可以点击添加按钮进入添加财务记录页面，填写相应属性，金额会自动算出，然后点击完成后如上传成功则会返回财务主界面，同时更新数据源，如上传失败则停在本界面，并对用户进行相应的提示。

4.2.8 签到与签退模块详细设计

签到与签退模块的显示位置设计在主页面的我的设置之中的子项里，当用户点击进入该页面的时候，会调用已经封装完成的地理位置类来进行当前位置的最新定位，并从服务器获取到系统时间显示在本页面，方便用户知道自己签到或是签退的具体时间，无论是签到还是签退均设置各自独立的地点更新按钮，并均可以填写附带信息，以说明相应情况，并且由服务器与本地共同完成当前签到签退数目的限制，一天一个账号只允许签到与签退各一次，多次签到或者签退并不会传送到后台进记录存储，以方便绩效考评。并且为了保证签到的真实性，系统会将设备的唯一标识同时传输到后台服务器中进行存储，考虑到信息安全因素，手机唯一标识数据的保密性将经过Base64编码与MD5的组合使用来实现提升，不会轻易泄露用户设备的隐私，同时也极大程度上保证了用户的确定性与唯一性，对于系统的可靠性也是进一步的提高与增强。

4.2.9 权限控制与安全处理模块详细设计

1. 用户角色的功能权限划分

按照用户的不同角色，根据其工作性质设计划分出每种用户角色应该具有的权限功能，达到权限分配合理，切实可控的效果，针对不同用户登录的时候会在系统上面进行角色判断，如果用户是团队注册用户，即企业相关人员，则不可登录到之后开发的客户版子系统之中，同理具有较弱权限的客户也只能登录到客户版系统中，而不能登录到企业版的系统中，达到用户权限的全面隔离与工作日报选择性隔离的目标，此部分均按照需求分析中的用例描述情况进行设计。

1. 新消息可见性设置

根据用户角色的不同，有时项目负责人或者项目经理的角色会面对比较多的项目记录，所以一旦有大量新消息出现的时候可能会导致该类角色不得不一一查看相关消息，但是项目经理是要对自己的项目负责的，所以其查看所有记录也是很正常的工作内容之一。不过如果是针对老板或者高级领导等角色的时候，显然大量并不想看的新信息提醒会成为一个严重的问题，所以项目提醒的权限控制由此而生，此类角色可以根据自己的关注点设置项目新消息的可见性，只需要接收那些自己目前比较关注的项目新消息即可，当想要查看其他项目的时候，将该项目的消息接收“开关”重新打开即可。

1. 用户可见性设置

当用户登录企业版客户端系统的时候，根据自己的角色情况需要去设置可见自己的人，这样自己上传的日报记录就能够被对方查看到，并由对方作出评论或审核批示。不同角色所拥有的权限不同，可见的人也不同，对于工程监控系统来说是一个不可缺少的重要功能设置。

4.2.10 更多设置功能模块详细设计

更多功能模块在设计中将会显示系统的其他基础功能，并且为后期开发预留接口，该功能应由主界面内的一个平行视图控制器来管理，界面风格按照即将介绍的风格进行设计实现，界面上面加载TableView控件，每个Cell将会显示对应的功能名称，当用户点击之后应该进入该功能界面，控制权转到相应的视图控制器中。

## 4.3 系统部分关键内容详细设计

4.3.1 客户端界面设计

本文所论述的基于iOS的工程监控系统使用了目前最为主流的Storyboard界面管理开发方式，主列表页面与详细信息页面的初期设计如图4.5所示。

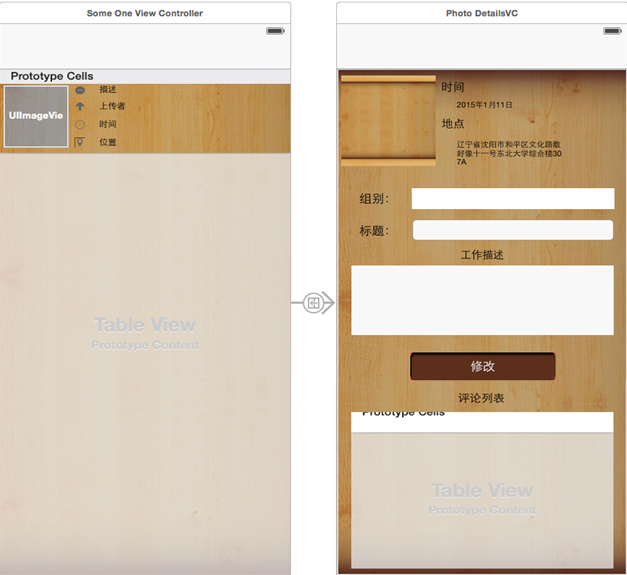


图4.5 主记录列表与详细信息初期未渲染界面设计图

由于显示界面与对应的跳转关系原始设计图内容过多，所以在此只选择部分关键界面的设计图进行展示说明。另外，本系统在一些功能界面上还辅以纯代码设计，这样可以大幅降低维护的难度，而且代码也更具逻辑性，但是整体的掌控与设计难度将会大幅增加，因为相对于纯代码与纯界面设计，这种混合的开发设计方式将对逻辑思维与系统整体的掌控提出更高的要求。

绩效考勤的签到与签退界面，以及账务管理初期界面设计如图4.6所示，相比主列表与详细信息页面，两者之间并不存在界面的层级与跳转关系。



图4.6 签到签退与账务管理初期未渲染界面设计图

4.3.2 主要工具类的设计

在介绍完成主体功能滞后，还有一些在开发阶段常用到的方法与函数，也封装成了一些类，这里面最为重要的工具类便是CommonFuncVC类与NetworkTool类，NetworkTool类是对网络请求类进行的二次封装，在系统实现的部分将会有详细介绍，本次主要介绍的是常用方法集合的工具类CommonFuncVC，其中所有方法均被设计成为类方法，这样可以在导入工具类头文件后，在不创建对象的情况下使用类名直接调用相关方法，简单高效，以下是在该工具类内设计实现的部分类方法：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*页面设置类的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//设置页面的tableview的样式与主页面一样

+(UITableView \*)setTableBgClear:(UITableView \*)tableView andBgImg:(NSString \*)imgName;

+(ImageObj \*)saveTakedPicture:(UIImage \*)image; //保存拍照后图片放在自创建的数据库中

+(BOOL) netWorkIsOK; //判断网络情况

+(NSString \*)getRecordTime:(NSString \*)timeStr; //返回解析后的正常时间格式

+(NSMutableDictionary \*)translateEntityListJson:(NSMutableArray \*)tempArray; //得到cell属性

+(UIImage \*) reSize: (UIImage \*) img newSize: (CGSize) viewsize; //按照指定尺寸剪裁图片

+(NSString\*)getSavePahtInDocumentDirectoryWithFileName:(NSString\*)filename; //获取文件路径

+(void)setCornerRadios:(UIView\*)view; //给相应的视图设置圆角

+(void)isHaveNewVersion; //判断是否有新版本更新

+(NSString \*)getTheDevicesType:(UIView \*)view; //判断设备的类型

+(NSString \*)getIDFVToIdentifyUser; //获取设备的IDFV码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*关于用户信息类的方法\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

+(NSString \*)getCurrentUserId; //返回用户的ID

+(NSString \*)getCurrentUserPassword; //返回用户的密码 无参数

+(NSString \*)getCurrentUserName; //返回用户的密码 无参数

4.3.3 主体控制器类功能设计

iOS程序开发相对于Android开发具有一定的特殊性，很多类在程序运行时用来保存有关数据，但是其大多数被隐藏在各个页面之中，而每个页面虽然由相应的控制器来管理，但是每个控制器也都分别是一个独立的类，可被其他的类来使用，这样就导致整个程序的类与方法并不适合用于完整讲述系统整体的逻辑设计说明，因此本文将会从中选取相对比较重要的类与比较常用到的自定义方法来分类介绍。本系统中所有的自创建类如图4.7所示。

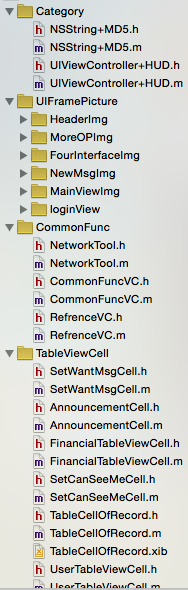
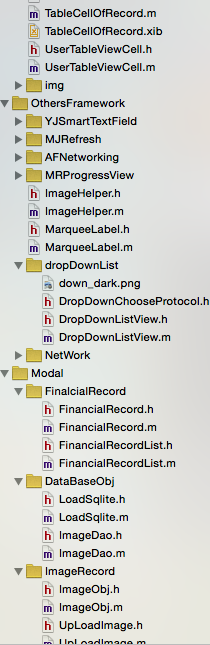
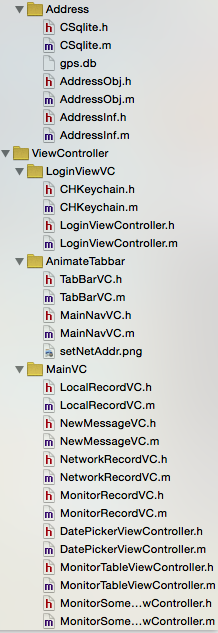
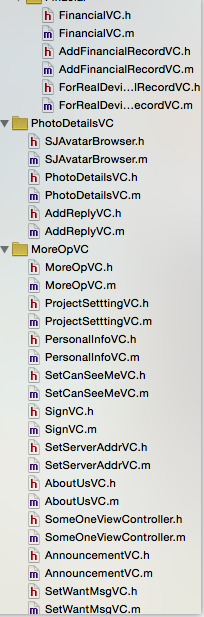
   

图4.7 系统中主要自创建类展示图

对于界面来说，每个页面之间或者存在跳转关系，或者存在相互调用关系，但在iOS的工程设计之中，每一个页面都将单独由一个ViewController，及视图控制器类来管理其内容，每个控制器类都拥有一对.h与.m文件。由于内容较多。所以在此只对最为重要的ViewController类，即视图控制器类进行详述，现在本文将对它们的作用及其显示内容进行一下介绍。

(1) 第一个介绍的类是AppDelegate，它虽然是程序开发默认生成的关键类文件，但是它为程序在启动之时首先会进入的类，也是整个应用程序最为重要的单例类。在这里将会对于整个系统做一些基础配置，以保证之后的功能能够正常使用。这里将会对KeyChain（iOS中的钥匙串）进行属性配置，USERNAME、USERID、PASSWORD、USERNAME、SERVERIP和USERROLE等成员属性的初始化，以保证系统可以方便取到当前用户信息进行请求，并且在这里声明主要的NavigationController，将之后出现视图控制器以入栈的形式包含进来，保证之后退出某界面的时候能够正常出栈，有效的避免内存泄露或者溢出等重大错误。

(2) 系统登录页面的LoginViewController类，是程序启动后，启动图标加载显示完成后，第一个出现的页面，用于用户输入账号密码，进行登录、验证以及记住用户信息等操作，但是提及这个界面之前，需要介绍下配置服务器的视图控制器SetServerAddrVC类，在这里可以由用户选择所属服务器地址，也可以默认恢复服务器地址为新功能测试服务器，并将所配置服务器地址存于本地数据库中，之后的所有网络请求都将基于本服务器地址进行拼接后，再与服务器交互。配置完成该网络后将进入登录阶段，在用户输入账号密码之后，将与服务器交互判断该用户是否属于本服务器内存在的有效用户，如果不是则提示用户名密码错误，如果验证通过则会将返回的用户基本信息存于KeyChain中，此时可以保证本用户的有效性，在下次不联网登录的时候用户一样可以进入系统查看本地记录，当用户退出登录时则会将本地用户信息删除，用户需要重新联网验证登录。

(3) 个人记录页面的LocalRecordVC类，当用户验证成功后由PUSH方式推入的系统首页面，也是整个系统整体UITabbarController的第一个TabbarItem内容，进入后将会启动程序的定位功能，定位功能将会根据设备所处环境的实际情况由系统在GPS、基站和WIFI三者之中根据内部机制进行合理选择。在这里面将会显示出的内容包括地理位置信息（跑马灯效果）、刷新位置信息按钮、分段器（两段分别为本地记录与网络记录）、右边 UIBarButtonItem照相按钮，左边消息中心图标，关于消息中心将会在之后进行单独介绍。由于个人与网络两种记录的显示类型是一样的，故在这里设计的时候采用一个控制器对两种记录显示进行切换，设置两种数据源，并在点击Segment的选项时，根据情况切换数据源。不过数据源的来源并不一样，本地数据来源于系统存储的数据库，而网络记录的数据源是对服务器返回数据的解析后进行处理后显示。基础界面均是展示记录时间、拍摄地点、上传者或上传状态和描述简介，以及通过代码控制的图片显示。同时在控制器中还涉及到右上角拍照的UIBarButtonItem生成本地记录的功能，当用户点击相机的时候，系统会自动调用相机接口，同时开启地理位置防止地理定位造假，用户拍照完成后可以选择保存或者取消当前拍摄记录，当用户取消后可重新拍照生成记录，采用后则会将已配置完成基础信息的界面显示出来，然后由用户填写项目分组等附带信息，并选择保存在本地数据库还是直接上传到服务器，无论怎样该记录都会先保存于本地数据库中，用户可以在本地滑动某一记录的Cell，选择是删除还是保留。当用户点击网络的Segment段时，则用户可以查看自己已经上传到服务器的记录内容，并可以根据实际情况对于错误记录直接进行网络修改然后更新保存在服务器端的记录信息，方便简洁。地理位置信息的显示将会依靠之前所提到的地理位置信息类模块设计，只需要调用已封装好的相关接口即可，用户在联网状态下随时可以手动点击来更新自己的当前位置，在无网状态下则会显示相关提示。在本页面类配置完成并显示出来后，它还会开启新消息查询的循环访问线程，以及新版本更新查询线程的单次请求，以完成系统的登录配置过程。另外，网络记录部分的TableView具有上下拉刷新功能。

(4) 消息通知中心页面的NewMessageVC类，该视图控制器类是与LocalRecordVC类具有跳转关系的类，位于个人记录的左边的消息图标样式的按钮，同时在个人记录页面的LocalRecordVC类中会持续开启新消息查询线程，时间间隔可以自行改变，暂定为300秒一次询问。当有新公告或者新消息出现时，则会进行相应提示，在这里是针对新消息，所以消息图标会变成带有新消息红点的状态。当用户点击消息中心的时候会以Modal形式跳转入新消息界面，并根据用户信息与服务器进行交互，获取到与自己相关的未读消息，并按照本地记录与网络记录的一样形式加载到自己的TableView上面将它们显示出来，便于用户查看，当用户从本页面以Dismiss的形式退出的时候，会再次与服务器进行交互，将与之相关的已读新消息记录全部清除，这样在断网或者用户没有退出该界面之前，都不会出现未读消息丢失的情况，是最为合理的实现逻辑。

(5) 团队效率页面的MonitorRecordVC类，在本页面将会显示日常记录与某人记录的Segment分段器，同时NavigationController的RightBarButtonItem是日期查找功能，其属于另一个页面DatePickerViewController类，该控制器上安放了一个系统原生的DatePicker控件，当用户以PUSH形式进入该界面时候，根据自己搜索日期选择完成后会自动POP出该页面，回到TbaleView界面并显示该日期的所有可见记录，如果没有记录项则会弹出提示。在本界面的Segment分段器采用另一种方式控制页面，每次点击将会直接跳转到新的页面控制器类，两个控制器分别是MonitorTableViewController类与MonitorSomeOneTableViewController类，这两者分别控制日常记录与某人记录，并且各自具有自己的TableView，完全分离控制显示，日常记录显示的是对我可见的所有人按时间逆序排列的记录项，而某人记录则是获取到所有对我可见的人列表，然后用户可以按照自己意愿来选取可见用户进行查看，这里附带一层跳转关系，到新的系统界面，由页面控制器类SomeOneViewController来进行控制显示，同样是记录列表形式显示，且本页面的所有TableView均设置由上下拉刷新功能，方便用户使用。

(6) 以上所介绍的视图控制器类内部都会有至少一个TableView来显示相关日报记录，而点击这些TableView的每个Cell后都将会进入到详细信息界面，而这个页面由PhotoDetailsVC类来进行控制显示，当由不同页面跳转到这一个界面的时候需要根据实际情况显示不同内容，所以相对而言，这个页面的设置逻辑最为复杂，所以在此会尽可能的简单描述该页面的逻辑处理过程。该页面会显示的内容包括照片记录（可点击放缩）、记录产生时间、记录产生地点、记录组别、记录标题、记录的工作描述，以及该记录具有的评论信息列表，同时还具有右上角的审核批示按钮，这个功能点将在之后介绍。当用户是从本地界面进入的时候，页面会按照本地记录详细信息形式显示，隐藏审核批示功能，因为本地记录不需要被评论，若该记录处于已上传状态的话，还需要隐藏上传按钮uploadOrChange功能，否则显示该按钮功能，使得用户可以上传保存在本地的日报记录。当用户是从网络界面进入的时候，需要去服务器请求评论列表，如果存在则加载到页面上面显示，其余的功能均可显示，附带信息等可进行重新编辑，然后修改上传。当用户从其他界面，如新消息记录、日常记录、某人记录或者日期查找等页面进入的时候，因为是面对别的用户的记录，所以该用户没有权限更改，故需要将所有附带信息内容设置为不可编辑，同时隐藏修改或上传按钮，显示评论列表。

(7) 审核批示页面的AddReplyVC类，本页面按照逻辑关系需要附带在由控制器类PhotoDetailsVC来管理的详细信息界面，当用户点击之后进入，添加完成评论并与服务器交互成功后会自动以POP方式退出该页面，同时更新详细信息页面的评论列表，评论列表默认按照时间逆序显示，即最新的评论记录将会显示在评论列表的最顶端。

(8) 财务管理的FinancialVC类，本页面将会按照项目与财务类型对财务记录进行显示，不过只显示对用户可见的财务记录，并且财务记录会限制客户端可显示数目，为了保证系统的性能，所有过早的财务记录可以在WEB端进行访问查看，本功能可以在WEB端进行记录数设置。用户在选择项目名称与财务类型的时候，同时系统会与服务器进行交互，显示出相应的财务预算与余额，并且根据与财务花费警戒线数值作比较，超过警戒线将会对用户做出提示，并将余额一项标红显示。当用户点击财务记录的简介后可以进入到财务的详细信息页面，由控制器类FinancialRecord来管理，并显示该条财务记录的详细信息，当然所有显示项均不可编辑。财务列表同样具有上下拉刷新功能，方便用户查看更多内容。

(9) 在财务记录界面中，右上角的UIBarButtonItem是添加财务记录的功能按钮，当用户想要添加财务记录的时候，需要在跳转之前进行判断，如果当前项目已完结则提示用户不可添加新的财务记录，否则可以让用户跳转到由页面控制器类AddFinancialRecordVC管理的添加财务信息界面，并可以编辑，用户必须添加全部必要信息后才可以与服务器交互，成功后会跳转回到财务界面，同时刷新财务界面，显示新纪录项。

(10) 我的设置页面的MoreOpVC类，里面由一个TableView显示所有的可用子功能，提供给用户点击后跳转到相应功能界面的操作，其中包含最底部的一个退出登录功能按钮，当用户点击的时候，将会POP到处于内存栈顶的登录界面，这样就会将所有已入栈的内容全部弹栈，避免出现内存问题。

(11) 公告中心页面的AnnouncementVC类，MoreOpVC类的跳转页面，用户点击进入后将会与服务器产生交互，并获取到对该用户开发的最新的50条公告信息，用户可以进行查看，公告列表的上下拉刷新功能未被设置，由服务器限制显示的记录项数目。

(12) 我的信息页面的PersonalInfoVC类，MoreOpVC类的跳转页面，用户进入查看自己的相关基础信息，如姓名、性别与头像等。

(13) 项目权限页面的ProjectSetttingVC类，MoreOpVC类的跳转页面，目前只有两个功能，谁可见我与消息提醒，分别跳转到控制器类SetCanSeeMeVC和控制器类SetWantMsgVC，在谁可见我页面可以设置可见我的用户。在消息提醒页面可以在设置想接受项目的新消息提醒，二者均是视图控制器类ProjectSetttingVC的跳转页面。

(14) 签到打卡界面的SignVC类，MoreOpVC类的跳转页面，用户进入后可以获取到服务器时间，并调用地理位置获取模块对地理位置进行更新，用户可以输入附带信息进行签到或者签退，系统将会对设备的唯一标识码进行Base64编码后再传输，从信息安全角度考虑，有效地提升防破解能力。

(15) 关于我们界面的AboutUsVC类，MoreOpVC类的跳转页面，介绍系统的基本信息，同时网络设置页面在内部仍旧可以被使用。

这些视图控制器类支撑起了系统的主体界面与功能，相当于是整个系统的“骨架”。

4.3.4 本地数据库设计

本地信息的存储使用到了KeyChain技术与SQLite数据库相关技术，KeyChain主要用于记录用户信息等基础属性，但是数据库将会保存大量的本地记录信息，所以对于数据库的结构设计等就显得尤为重要，如何确定属性、数据库表和查询方式均有可能对系统的性能造成很大影响，下面将介绍一下整个系统的本地数据库设计内容。

1. 图片信息表Image保存了日报记录的全部基础信息，具体结构见表4.1所示。

表4.1 Image记录信息表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **字段名称** | **数据类型** | **字段说明** |
| 1 | userId | Integer | 用户ID |
| 2 | imageName | Text | 存储图片名（主键） |
| 3 | isPublic | Integer | 是否公开 |
| 4 | uploadTime | Text | 上传时间 |
| 5 | takeTime | Text | 拍摄时间 |
| 6 | description | Text | 记录描述 |
| 7 | imgLatitude | Float | 拍摄地址纬度 |
| 8 | imgLongitude | Float | 拍摄地址经度 |
| 9 | popularity | Integer | 受欢迎程度（预留） |
| 10 | isUp | Integer | 是否上传 |
| 11 | imgWidth | Integer | 图片宽度 |
| 12 | imgHeight | Integer | 图片高度 |
| 13 | takeAddress | Text | 拍摄地址 |
| 14 | title | Text | 日报标题 |
| 15 | workGroup | Text | 日报分组 |

(2) 关于日报记录中的图片的保存方式并未存入到数据库中，图片表中只存储了每 个图片记录的相关信息以及图片名称，而为了节省之后上传图片所需要用到的流量，图片则是按照压缩系数0.1进行图片压缩，再经过Base64编码后保存于系统的沙盒路径下，并且单独成为一个文件，不支持用户从手机相册内点击选择形成假冒记录项，同时重新编码也使得该记录被用户破解后重新生成新记录的造假行为难度大幅提升，有效的提高了信息安全与记录唯一的程度。

(3) 由于图片信息表是最为关键的本地记录表，所以对其相关的SQL语句以及查询方法会做出最为全面的介绍，具体设计内容如图4.8所示。

图4.8 本地数据库及接口函数说明设计图

(4) 关于本系统的本地数据库表还有用户表，用于保存用户的基础信息，该表的每个字段是用户角色所拥有的所有属性，还有地理位置信息表，该表不仅存有经纬度信息，同时会将经过逆地理位置解析后地址信息更新在表中的相应字段中，这样可以存下该用户的日常轨迹，对于工程组研究某种职务的员工的活动规律具有很大的参考价值，这两张表在最初的时候是作为使用数据表来设计的，但是在实际开发过程中使用了新的技术手段，所以在本次开发中将这两张数据库表用作预留接口，方便之后的拓展开发使用。

4.3.5 网络接口设计

整体项目组在一起深入研究了市场需求调研结果，并确定了总体需求分析之后，全员一起分析讨论后最终确定了对于接口API的设计方案，在iOS移动端上面的系统开发只需要在测试相关API可以正常使用，并无重大错误的情况下，按照后台人员的接口设计文档能够向后台服务器请求或者传输数据即可。其中涉及到的隐私保护等信息安全方面的内容将会在手机终端本地进行处理与实现，具体介绍参见表4.2。

表4.2 服务器部分接口描述与说明表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **接口访问示例** | **方式** | **接口说明** |
| 1 | webservice/user/login.do | post | 验证用户登录 |
| 2 | webservice/record/photodetail/2369.do | get | 查看一条图片记录 |
| 3 | webservice/user/listUp/2369.do | get | 取得能查看我的所有用户 |
| 4 | webservice/user/listDown/2369.do | get | 取得所有对我可见的用户 |
| 5 | webservice/user/getAllUser.do | get | 取得系统中的所有用户 |
| 6 | webservice/user/detail/2369.do | get | 取得该用户详细信息 |
| 7 | webservice/user/addUserCanSeeMe.do | post | 增加对自己可见的用户 |
| 8 | webservice/user//deleteUserCanSeeMe.do | post | 删除对自己可见的用户 |
| 9 | /webservice/record/addPhoto.do | post | 用户上传一条图片记录 |
| 10 | /webservice/record/listPhoto/2369.do | post | 用户下载一条图片记录 |

由于项目中实际需要的接口内容过多，且其中大部分工作属于后台服务器实现，本地客户端系统承担的任务更多是测试接口的相关工作，所以对于其他的接口在此不再进行赘述。

## 4.4 本章小结

本章从系统主体模块与系统架构分析与总体设计两方面出发，对系统概要设计进行了较为全面的介绍，并对系统的功能模块详细设计进行了全面的说明，包括日常记录管理模块详细设计等。随后又对系统的部分关键内容详细设计进行了详尽的解释，包括主体控制器类设计和本地数据库设计等，从而使得自己更为深入的了解了各部分模块之间的衔接与协调关系，为下一步进行编码开发，最终能够实现本管理系统奠定了良好的基础。

第5章 系统实现

## 5.1 网络请求与二次封装模块实现

本系统所使用的最为主要的网络框架是AFNetworking与ASIHttpRequest两个主流框架，在前面的关键技术一章内已经对二者进行了简单的介绍，并提到对一些常用的网络请求类进行了一定程度的二次封装，方便编码人员在开发与测试阶段进行使用，在本部分将会进行详细的说明。

5.1.1 URL的统一编码处理

对于所有的网络请求，其所发送的链接地址都需要由最基本的服务器地址来进行拼接，这其中包括GET和POST两种方法进行网络请求，同时需要对整个发送链接进行统一字符集编码后才可以将参数发送出去，然后交由服务器进行处理，并给出响应后再交由本地系统进行处理。相关的处理函数关键代码如下：

+(NSString\*)preprocessURLStr:(NSString\*)urlStr { //urlStr 未处理的 URL 字符

NSString \*serverIP = [[NSUserDefaults standardUserDefaults] objectForKey:@"serverIP"];

urlStr = [serverIP stringByAppendingPathComponent:urlStr];

urlStr = [urlStr stringByAddingPercentEscapesUsingEncoding:NSUTF8StringEncoding];

return urlStr; //处理后的 URL 字符

}

以上代码具体流程的简单解释是先对URL中的字符作预处理，然后将 URL 与 用户输入且已保存在系统的serverIP 进行连接，然后将 URL 进行 UTF-8字符编码处理后返回给调用对象。

5.1.2 不同请求方式的消息封装

1. 使用AFNetworking网络框架进行封装的基本GET方式请求[17]，关键代码如下所示：

+(void)baseGetRequsetWithURLStr:(NSString\*)urlSt parameter:(NSDictionary\*)parameterDict

success:(void (^)(AFHTTPRequestOperation \*operation, id responseObject))success

failure:(void (^)(AFHTTPRequestOperation \*operation, NSError \*error))failure{

// parametersDict 参数字典

urlStr = [self preprocessURLStr:urlStr]; //请求用的 URL

AFHTTPRequestOperationManager \*manager = [AFHTTPRequestOperationManager manager];

[manager GET:urlStr parameters:parameterDict success:success failure:failure];

}//success 请求成功回调函数 failure请求失败回调函数

1. 使用AFNetworking网络框架进行封装的特殊GET方式请求，在本系统中会以.do为结尾访问的URL，关键代码如下所示：

+(void)specialGetRequsetWithURLStr:(NSString\*)urlStr parameter:(NSString\*)parameterStr

success:(void (^)(AFHTTPRequestOperation \*operation, id responseObject))success

failure:(void (^)(AFHTTPRequestOperation \*operation, NSError \*error))failure{

urlStr = [[urlStr stringByAppendingPathComponent:parameterStr]stringByAppendingString:

@".do"];

urlStr = [self preprocessURLStr:urlStr]; //请求用URL的预处理

AFHTTPRequestOperationManager \*manager = [AFHTTPRequestOperationManager manager];

[manager GET:urlStr parameters:nil success:success failure:failure];

}//success 请求成功回调函数 failure请求失败回调函数

1. 使用AFNetworking网络框架进行封装的基本POST方式请求[17]，关键代码如下所示：

+(void)basePostRequsetWithURLStr:(NSString\*)urlStr parameters:(NSDictionary\*)parametersDict

success:(void (^)(AFHTTPRequestOperation \*operation, id responseObject))success

failure:(void (^)(AFHTTPRequestOperation \*operation, NSError \*error))failure{

urlStr = [self preprocessURLStr:urlStr];

AFHTTPRequestOperationManager \*manager = [AFHTTPRequestOperationManager manager];

[manager POST:urlStr parameters:parametersDict success:success failure:failure];

}//success 请求成功回调函数 failure请求失败回调函数

1. 使用AFNetworking网络框架进行封装的特殊的POST方式请求，用于那些虽然是 POST方式请求但还需要用 URL 进行传参的接口，关键代码如下所示：

+(void)specialPostRequestWithURLStr:(NSString\*)urlStr parameter:(NSString\*)parameterStr

success:(void (^)(AFHTTPRequestOperation \*operation, id responseObject))success

failure:(void (^)(AFHTTPRequestOperation \*operation, NSError \*error))failure{

urlStr = [[urlStr stringByAppendingPathComponent:parameterStr]stringByAppendingString:

@".do"];

urlStr = [self preprocessURLStr:urlStr];

AFHTTPRequestOperationManager \*manager = [AFHTTPRequestOperationManager manager];

[manager POST:urlStr parameters:nil constructingBodyWithBlock:nil success:success failure:

failure];

}//success 请求成功回调函数 failure请求失败回调函数

1. 使用ASIHTTPRequest网络框架进行简单使用的GET方式异步请求方法与POST异步请求方法，关键代码如下：

ASIHTTPRequest \*request = [ASIHTTPRequest requestWithURL:url]; //GET方式异步请求

[request setDelegate:self];

[request startAsynchronous];

NSURL \*url = [NSURL URLWithString:baseUrl]; //POST方式异步请求

ASIFormDataRequest \*request = [ ASIFormDataRequest requestWithURL :url];

[request setDelegate:self];

[request setRequestMethod:@"POST"];

[request addPostValue:@"1" forKey:@"page"];

[request startAsynchronous];

无论是POST方式还是GET方式，两者在ASIHTTPRequest框架下异步请求的回调函数均在可以在同一个函数内部进行处理，这样可以大大提升代码的重用性，提升代码质量，以下为所使用的异步请求回调函数：

- ( void )requestFinished:( ASIHTTPRequest \*)request {…} //成功的回调函数

- ( void )requestFailed:( ASIHTTPRequest \*)request {…} //失败的回调函数

由于实际的回调处理代码较多，所以在此只介绍处理函数名称，其他不再详述。

## 5.2 登录与退出登录的内存管理

在iOS开发中，存在ARC与非ARC两种开发模式，ARC即自动引用计数，开启ARC模式的时候开发者不用考虑内存管理问题，系统会自动处理不使用的资源，非常方便。于本系统的登录功能最初存在内存隐患，即退出登录时可能造成的内存泄露。

退出登录的最初实现是采用只释放当前页面资源然后以Modal形式跳转到登陆界面的方式，经过测试显示会发生内存泄露，一些资源占用并未被解除，但是却找不到清除方式，于是就会存在大量的已加载资源不会被自动释放，因为系统会认为你还在使用那些资源，多次使用退出登录功能再登录的话，会造成内存的暴增与极大浪费。

经过一番设计与编码的重新实现，最终作出前面所提到的最合理的系统设计图，使用栈管理方式以PUSH和POP方式实现登录与退出登录的操作。用户一旦退出登录，则会将之前入栈的所有资源全部释放，避免本处内存问题的出现。登录之前系统配置处理的具体实现代码如下所示：

//取出storyboard中的登陆界面的ViewController

UIStoryboard \*storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];

UIViewController \*loginNavVc = [storyboard instantiateViewControllerWithIdentifier:@"login"];

//把登陆界面放入作为root的NavigationController中

UINavigationController \*navigationController = [[UINavigationController alloc] initWithRoot

ViewController:loginNavVc];

[navigationController.navigationBar setHidden:YES]; //把顶部的NavigationBar隐藏

//把rootViewController设置为登陆界面所在的NavigationController

self.window.rootViewController = navigationController;

之后当用户进入系统，并且登录验证成功的时候将会自动进行跳转，调用代码 [self.navigationController pushViewController:mainTabBar animated:YES];进行压栈的PUSH操作后进入系统主界面，此处还涉及到用户角色的判断，之后会进行详细介绍。

当用户点击退出登录的时候，将会执行POP出栈操作，将已经入栈的界面全部清理出栈，保证内存的合理释放，关键代码如下：

AppDelegate \*appOfLogOff = (AppDelegate \*)[[UIApplication sharedApplication] delegate];

[CHKeychain delete:appOfLogOff.KEY\_USERNAME\_PASSWORD];

//pop回到rootViewController

[self.tabBarController.navigationController popToRootViewControllerAnimated:YES];

测试显示最初进入页面时系统的占用大约在15MB左右，将所有页面全加载出来后，内存的占用约为40MB，选择退出登录后则会再次恢复为15MB左右，此种实现方式使得本应用的健壮性得到了进一步提高。

## 5.3 系统的详细功能模块实现

5.3.1 系统的登录退出与用户的角色管理模块功能实现

当用户在由LoginViewController管理的登录界面输入账号与密码的时候，系统便会拼接好URL并与配置好的服务器地址进行交互，进行用户信息验证，同时将用户基本信息返回，以便本地进行保存以及为之后网络请求的发送进行基础信息配置，完成保存与角色判断的关键实现伪代码如下：

- (IBAction)logIn:(id)sender {

//获取用户输入的账号密码

if ([userPasswd isEqualToString:@""]||[userId isEqualToString:@""]){

// 提示输入用户名或密码

}

else{

[self showOverlayInView:self.view title:@"正在登陆中，请稍候"];

//声明并创建AppDelegate类型单例变量appDeleOfUserInfo

NSMutableDictionary \*mDictOfUserInfo = [[NSMutableDictionary alloc] init];

//发送网络请求

[NetworkTool loginWithLoginID:userId password:userPasswd success:^(AFHTTPRequ

estOperation \*operation, id responseObject) {

NSDictionary \*json = (NSDictionary \*)responseObject;

//在这个地方到时候进行角色判断，在客户版进行开发的时候

NSString \*flag = [json objectForKey:@"flag"];

if ([flag isEqualToString:@"true"]) {

NSString \*userName = [[json objectForKey:@"user"]objectForKey:@"name"];

NSString \*userRole = [[json objectForKey:@"user"] objectForKey:@"role"];

if (![userRole isEqualToString:@"ROLE\_CUSTOMER"]) {

//是企业员工角色，存储用户ID与密码以及返回的用户名

}

else{

//提示企业版本请使用企业账号登录

} }

else{

//提示用户名或密码有误，请重新输入

}

//去掉进度条

[self hideOverlayForTheView:self.view];

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

//提示服务器地址配置错误或网络状况不佳，请检查您的网络配置

//去掉进度条}];

}}

用户在登录成功后则会保存相关信息，一旦用户信息里面有值的话，一定是已经验证过的有效用户，退出登陆的时候将钥匙串中已保存的用户信息清空即可完成注销，在之前的登录与退出登录内存管理一部分已对关键代码进行了介绍，在此不再进行赘述。

5.3.2 日报记录管理模块功能实现

日报记录的管理分为个人记录管理与网络记录管理，而网络记录又分为个人网络记录与对我可见人的网络记录，个人记录包括生成、修改、上传以及本地管理等基本功能，其中最为重要的是生成记录的代码，需要使用UIImagePickerControllerDelegate代理来调用相机接口，关键代码如下所示：

/\*\*\* 调用相机进行拍照右上角相机按钮 \*\*\*/

- (IBAction)beginCamera:(id)sender {

//本地记录才让拍照

if (segmentConOfLocal.selectedSegmentIndex == 0) {

if ([UIImagePickerController isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera]){

currentPickerController=[[UIImagePickerController alloc] init];

currentPickerController.sourceType=UIImagePickerControllerSourceTypeCamera;

currentPickerController.delegate = self;

currentPickerController.modalTransitionStyle=UIModalTransitionStyleFlipHorizontal;

[self presentViewController:currentPickerController animated:YES completion:nil]; }

}else if (segmentConOfLocal.selectedSegmentIndex == 1) {

//网络记录部分

}

}

- (void)imagePickerController:(UIImagePickerController \*)picker //UIImagePickerControllerDelegate

didFinishPickingImage:(UIImage \*)image

editingInfo:(NSDictionary \*)editingInfo { //开启新的线程去存储照片 }

- (void)imagePickerControllerDidCancel:(UIImagePickerController \*)picker { NSLog(@"退出了相机");}

- (void)savePicture:(UIImage \*)image{ //开启新线程调用常用方法类中的照片保存方法 }

所有网络记录整体的实现思路均为先拼接好请求的URL，然后由系统向服务器发送请求，服务器将相应的记录项信息返回，然后系统对返回的信息进行解析，并封装成相应的记录模型，接着作为数据源，封装成Cell后加载到相应界面的TableView上面进行显示，当然本功能一定包括上下拉刷新的设置与实现，日报记录的模型显示与上下拉刷新的功能实现界面请参看图5.5，关键的伪代码如下面所示：

- (IBAction)segmentChanged:(id)sender {

switch ([sender selectedSegmentIndex]) {

case 0: //本地界面

case 1: //网络界面

[self.btnOfCamera setEnabled:NO]; //发送网络请求获取数据源 }

}

//网络请求得到数据源之后,配置相关的Cell代码

ImageObj \*image = [arrayOfNetWorkRecord objectAtIndex:indexPath.row];

static NSString \*cellId = @"cellOfNetWork";

TableCellOfRecord \*cell = (TableCellOfRecord \*)[tableView dequeueReusableCellWithIdentifier:

cellId];

if (cell == nil) { //新建一个cell }

[cell.imgOfRecord setImageWithURL:[NetworkTool downloadUrlOfPhotoByImageName:

//配置Cell的相关属性

return cell;

5.3.3 日报记录详细信息查看与修改模块功能实现

日报记录详细信息查看时会根据用户进入的页面入口进行相关配置，当用户是从本地页面进入的时候，如果该记录已经上传则只允许用户在本地编辑，但是上传按钮将会隐藏使得用户无法上传，若该记录未上传则上传显示，允许用户编辑并上传，但是本地会隐藏审核批示的评论功能。当用户是从自己的网络记录页面进入的时候，可以查看该记录的评论列表，也可以添加评论，并且可以对自己信息填写有错误的网络记录进行编辑，在联网情况下成功修改。如果用户是从查看其他用户记录的页面进入，则用户只能对该用户进行审核批示，并不能对他人记录内容进行编辑，也不会显示修改按钮。关于详细信息查看与修改最为关键的代码部分就是判断用户进入的界面，同时进行显示页面配置的相关内容，部分关键代码如下所示：

@synthesize flagOfController; //用户进入的界面判断标志

//flag的判断 判断是从哪个界面进来的，做相应的定制

//0 -- 从刚拍完照片的本地进入 1 -- 从本地界面点击cell进入

if (self.flagOfController == 0 || self.flagOfController == 1){

if ([UIImage imageWithContentsOfFile:[[NSHomeDirectory() stringByA

ppendingPathComponent:@"Documents"] stringByAppendingPathComponent:self.imgObjOfDetails.imageName]] != NULL){

[self.imageOfRecord setImage:[UIImage imageWithContentsOfFile:

[[NSHomeDirectory() stringByAppendingPathComponent:@"Documents"] stringByAppendingPathComponent:self.imgObjOfDetails.imageName]]];

}

//不显示评论的按钮，使其失效

if (self.imgObjOfDetails.isUp == 1){ //未上传的状态下显示上传按钮 }

else{ //已上传记录隐藏按钮 }}

else{ //非本地进入

if (self.flagOfController == 2) { //2 -- 从网络cell点击界面进入

//获取项目组列表 判断项目性质

if (isUnFinishedPro){ //说明是已经完成的项目 不允许修改记录 }}

else{ //正常显示修改按钮 }

else if (self.flagOfController == 3) { //3 -- 从监控界面进入 均不允许修改内容}

}

5.3.4 地理定位模块功能实现

对于地理位置的解析实际上是对于定位功能部分最为复杂的一个部分，考虑到用户的实际网络情况，只使用一种定位的方法是并不可靠的，而在真正应用时，关于谷歌、百度还是高德地图的选择就变得比较困难，为了尽可能得实现安全稳定的地理位置解析，设计之初选择了高德、百度与谷歌三种途径进行组合的方式完成这一项功能，这并不是一件简单的事情，因为三家公司关于定位的封装与解析方式均采用了不同的方式，而且各自拥有自己的坐标体系，相互之间的转换可以说是比较麻烦，所以必须对三种地理位置封装、解析以及定位均进行了解，然后使用相应的地理位置纠偏方法才能实现本功能。本文所设计系统最终采取的策略是先向封装进入系统的高德SDK进行请求，若请求不到则转向百度SDK进行请求，若非网络原因却仍旧请求不到，则转向尝试进行谷歌的网络请求，如果仍旧不可请求到，则说明系统现在的网络处于未连接的状态，提示用户检查网络等。由于解封部分函数代码过多，在此不做过多提及，下面是进行地理位置请求的关键代码：

/\*\*\*\*\*谷歌部分\*\*\*\*\*/

NSString \*tmp;

tmp=[NSStringstringWithFormat:@"http://maps.google.cn/maps/api/geocode/json?latlng=%f,%f&language=zh-CN&sensor=false",addr.latitude,addr.longitude];

//加载一个NSURL对象

NSURLRequest \*request = [NSURLRequest requestWithURL:[NSURL URLWithString:tmp]];

NSData \*response = [NSURLConnection sendSynchronousRequest:request returningResponse:nil error:nil]; //将请求的url数据放到NSData对象中

NSDictionary\*locDic = [NSJSONSerialization JSONObjectWithData:response options:

NSJSONReadingMutableLeaves error:&error];

NSMutableArray \*locArray = [locDic objectForKey:@"results"];

/\*\*\*\*\*百度部分\*\*\*\*\*/

tmp=[NSStringstringWithFormat:@"http://api.map.baidu.com/geocoder/v2/?ak=5bd4c42cbbcce4e82c35f0020322722e&callback=renderReverse&location=%f,%f&output=xml&pois=1",addr.latitude,addr.longitude];

NSDictionary \*locDic = [NSJSONSerialization JSONObjectWithData:response options:

NSJSONReadingMutableLeaves error:&error];

NSMutableArray \*locArray = [locDic objectForKey:@"results"];

self.locInfo = [[locArray objectAtIndex:0] objectForKey:@"formatted\_address"];

/\*\*\*\*\*高德SDK部分\*\*\*\*\*/

AmapReGeocodeSearchRequest \*regeoRequest=[[AMapReGeocodeSearchRequest alloc]init];

regeoRequest.searchType=AMapSearchType\_ReGeocode;

regeoRequest.location=[AmapGeoPoint locationWithLatitude:addr.latitude longitude:addr.longitude];

regeoRequest.radius=10;

regeoRequest.requireExtension=NO;

[self.search AMapReGoecodeSearch:regeoRequest];

在之后的性能测验中，目前整个系统对于高德SDK的支持最为快速准确，所以在之后的实现过程中主要对高德SDK的定位进行了大量的优化处理，将它作为了系统的主定位选择，这样不仅可以提高程序的效率，还可以精简系统的大小。地理定位功能的显示界面具体请参看图5.4呈现的内容。

5.3.5 审核消息与公告中心管理模块功能实现

工程监控系统每隔一定时间会与服务器进行一次交互，查询是否有新消息或者新公告未被查看，如果存在的话则对相应的内容进行提醒，否则不作任何处理，程序功能实现的总体流程如图5.1所示。

在本部分功能实现的时候使用到到了多线程的相关知识，需要开启定时器的新线程来进行新消息或者新公告是否存在的判断，是一种结合多线程操作的网络请求操作，具体的实现过程将会在后面进行相关介绍。



图5.1 新消息和新公告请求流程图

开启计时查询功能需要计时器NSTimer的使用，关于每次请求的时间间隔可以由代码来进行人工控制，实现本部分功能的关键代码如下：

[NSTimer scheduledTimerWithTimeInterval:300.0f target:self selector:@selector(requestIsUpdate)

userInfo:nil repeats:YES]; //在此设置询问的时间间隔 现在为300s 开启定时器

-(void)sendLocInfoToServer:(NSNotification \*)sendLoc{

if (sendLocFlag) {

[NetworkTool ifIHaveUnreadCommentSuccess:^(AFHTTPRequestOperation \*operation, id responseObject) {

NSString \*flag = [recvDict valueForKey:@"flag"];

if ([flag isEqualToString:@"true"]) { //解析服务器返回内容，判断相应部分标志位 }

else{ NSLog(@"没有新消息"); }

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

NSLog(@"获取评论部分出错 啥也不做 300s之后会再次请求");

}];}

}

目前系统确定的时间间隔是300秒，即5分钟与服务器进行一次信息交互，最终实现的界面显示效果如图5.2所示。



图5.2 新消息和新公告提醒功能实现效果图

5.3.6 财务记录管理模块功能实现

用户财务记录全部由网络请求得来，为保证记录与服务器的一致性，所以本地对于这类信息不作任何存储。具体流程是先从服务器请求到有关的财务记录，经过处理后作为主数据源，封装成财务记录的Cell后加载到本页面的TableView上面，同时发送请求获取当前项目分类的预算与余额，并按照系统设计部分进行财务预警的显示处理，然后呈现在界面上，用户每点击一次项目分类就会向服务器请求一次最新的财务余额信息，当用户添加一项财务记录的时候，具体的操作便是按照封装好的网络请求类，一般选取POST方式与服务器交互，由服务器将传送过去的新纪录项插入后台数据库，同时本地数据源进行更新显示即可，财务记录所使用的关键函数及伪代码说明如下：

-(void)configUI{ //对界面视觉显示进行相关配置 }

-(void)configData{ //对界面的数据显示进行相关配置处理 如获取项目列表与相关财务记录 }

-(void)getNewerRecords{ //获取第一条记录的时间,看看之前是否有财务记录 获取最新记录 }

-(void)getOlderRecords{ //获取最后一条记录的时间,看看之前是否有财务记录 获取以前记录}

-(void)addFinancialRecord{ //跳转到添加财务记录的界面 另一控制器管理添加记录信息操作 }

//设置两个下拉菜单

self.viewOfDropDown.mSuperView = self.view;

[self.viewOfDropDown setDropDownDataSource:self];

[self.viewOfDropDown setDropDownDelegate:self];

//添加完记录之后返回时同时更新余额

NSString \*project = [self.viewOfDropDown currentTitleInSection:0];

NSString \*type = [self.viewOfDropDown currentTitleInSection:1];

[NetworkTool getBudgetWithProject:project costType:type success:^(AFHTTPRequestOperation \*operation, id responseObject) {

NSString \*titleOfLabel = [listOfRecords getBalaceAndBudgetStrUsingNetworkResponse:

responseObject];

if (titleOfLabel != nil) { //对财务预警显示做处理 }

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

NSLog(@"更新预算与余额超时 "); }];

}

5.3.7 签到与签退模块功能实现

用户在进入签到打卡界面的时候，会调用一次地理位置信息进行一次刷新，这样在联网状态下就会对用户当前位置进行重新确定，同时在用户签到签退的时候会对保证设备唯一性的标识符进行Base64编码，然后再发送到服务器，最后实现的系统界面效果如图5.3所示。



图5.3 签到签退界面实现效果图

在进入签到界面则会调用[self getServerTime]语句来获取服务器的当前时间显示在界面上面，同时调用[addrOfSign refreshAddress]语句来重置地理位置信息，关于签到与签退功能的实现代码类似，所以在此结合后进行一次性说明，关键代码如下所示：

- (IBAction)signIn:(id)sender { //签到 0 签退 1

NSString \*deviceId = [self getIDFVToIdentifyUser]; //获取设备IDFV码，构造唯一标示

NSString \*signInLoc = [self.lbOfSignInAddr text];

NSString \*signInComm = [self.textViewOfSignInComm text];

[NetworkTool registerOutInWithLocation:signInLoc type:NO remark:signInComm device\_id:

deviceId success:^(AFHTTPRequestOperation \*operation, id responseObject) {

NSDictionary \*recvDict = (NSDictionary \*)responseObject;

NSString \*flag = [recvDict objectForKey:@"flag"];

//判断签到状态 未签到则提示签到成功

if ([flag isEqualToString:@"true"]) {

//成功签到，开始上班 }

else if([flag isEqualToString:@"checked"]){

//已签到则提示签过到了 若为签退则提示签退}

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

//提示考勤失败，重试或检查网络}];

}

对于设备的唯一标示码是由手机的IDFV码经过本地处理后进行存储的，并对其采用了Base64的编码操作，实现Base64编码关键代码如下所示：

NSString \*identifierStr = [[[UIDevice currentDevice] identifierForVendor] UUIDString];

//使用base64加解密的代码段

NSData \*originData = [identifierStr dataUsingEncoding:NSASCIIStringEncoding];

NSString \*encodeResult = [originData base64EncodedStringWithOptions:

NSDataBase64EncodingEndLineWithLineFeed];

5.3.8 用户权限管理模块功能实现

(1) 客户与用户的权限区分

客户可以查看与其相关联的项目记录，并且具有评论权限，用户的功能则按照系统设计里面介绍的一样，具体实现的时候只需要对其当前登录的用户角色是客户还是非客户进行判定即可。

(2) 新消息可见性与用户可见性设置

按照系统设计介绍，用户可以根据自己的实际情况进行新消息可见性与用户可见性的设置，但是本部分内容在实现过程中遇到的问题是当用户点击对应开关之后，网络请求发出修改了相关可见性，但是滑动页面后再次回来则发现该开关状态恢复原样，原因在于没有对数据进行及时的更新，导致数据显示的时候仍旧是旧数据，所以会出现设置“无效”的情况，经过仔细修正数据源的更新已经完美地解决了该问题。

关于本功能的具体实现过程完全参照系统设计进行，最后实现的界面效果如图5.4所示。



图5.4 权限设置相关界面实现效果图

用户可以在上图所示的界面中显示的开关键对相应条目的可见性进行设置，由于部分内容属于真实信息，考虑隐私问题在此进行了一定的处理，界面仅供了解。相应条目的开关键打开则可见，关闭则取消可见，由于部分代理方法之前已经提到过多次，且代理方法一般比较繁多，所以在此只介绍实现的核心代码如下所示：

-(void)changeSwitchInDataSource:(NSNotification \*) noti {//可见性列表的设置 关键的显示部分

[[self visableList] setObject:[[noti object] objectForKey:@"value"] atIndexedSubscript:[[[noti object] objectForKey:@"tag"] intValue]];

}

-(void) loadUsersICanSee { //获取全部对我可见的用户

[NetworkTool getUsersWhoCanSeeMeSuccess:^(AFHTTPRequestOperation \*operation, id responseObject) {

NSDictionary \*recvDict = (NSDictionary \*)responseObject;

NSMutableArray \* allUsers = [recvDict objectForKey:@"entityList"];

for (NSMutableDictionary \* userDic in allUsers) {

[self.canSeeUserList addObject:userDic];

}

[self loadAllUsers];

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

//提示用户检查网络

}];

}

-(void) loadAllUsers{ //获取全部用户

[NetworkTool getAllTheUsersSuccess:^(AFHTTPRequestOperation \*operation, id responseObject) {

//解析全部用户，并将对我可见的用户进行标识，加入到可见列表里面

[self.fullUserList addObject:userDic];}

[self.tableOfSetCanMe reloadData];

} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) { //提示用户检查网络 }];}

5.3.9 更多设置模块功能实现

关于更多设置的功能模块实现完全按照系统设计中所描述方式进行实现，在系统主界面的平行视图控制器中在该部分添加TableView，在每个Cell的部分添加对应功能，实现用户点击后的跳转，实现的关键伪代码如下：

//主Tableview上面的每个Cell所对应的功能名称

-(UITableViewCell \*) tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath {

switch (indexPath.row) {

case 0: cell.textLabel.text = @"公告中心"; break;

case 1: cell.textLabel.text = @"个人信息"; break;

case 2: cell.textLabel.text = @"项目权限"; break;

case 3: cell.textLabel.text = @"签到打卡"; break;

case 4: cell.textLabel.text = @"关于我们"; break;

case 5: cell.textLabel.text = @"网络设置"; break;

default: break;}

return cell;

}

//主Tableview上面的每个Cell所对应功能的跳转界面

-(void)tableView:(UITableView \*)tableView didSelectRowAtIndexPath:(NSIndexPath \*)indexPath{

NSInteger chose = [indexPath row];

switch (chose){

case 0: [self performSegueWithIdentifier:@"toAnnouncementSegue" sender:self]; break;

case 1: [self performSegueWithIdentifier:@"toPersonalInfoSegue" sender:self]; break;

case 2: [self performSegueWithIdentifier:@"toProjectSettingSegue" sender:self]; break;

case 3: [self performSegueWithIdentifier:@"toSignSegue" sender:self]; break;

case 4: [self performSegueWithIdentifier:@"toAboutInfoSegue" sender:self]; break;

case 5: [self performSegueWithIdentifier:@"toNetConfigSegue" sender:self]; break;

default:break;}

}

5.3.10 用户记录上下拉刷新功能实现

用户的日报记录可以进行上下拉刷新以查看以前记录与最新记录，每次会发送相应的时间，下拉则将当前时间之后的记录请求回来，如果没有则提示用户已达到最新记录，上拉刷新则请求当前时间点之前的记录，如果没有则提示用户已加载完所有记录，界面的最终实现效果如图5.5所示。

用户采用上拉动作，即手指向上滑动的时候，当前界面上的TableView将会根据用户滑动的范围是否超过手机当前显示范围来判断是否需要显示更多的Cell内容，当需要的时候则会刷新数据源，然后重新加载数据到界面上，实际上是将新数据进行补充加载的过程，这样一次上拉刷新就会被完成，下拉刷新也是同样的原理。用户始终不松开滑动手势，则手机不会响应刷新功能，于是用户在不想刷新的时候只需要将滑动区域重新放回屏幕内，即手指下移或者上移即可。



图5.5 上下拉刷新界面实现效果图

上下拉刷新功能是整个系统查看日报记录环节使用最为频繁的功能模块，在此使用到了第三方开源框架MJRefresh进行相应的设置，其具体实现的关键代码是首先在ViewDidLoad函数中对上下拉刷新进行前期设置，关键代码如下：

// 1.下拉刷新显示最新记录(进入刷新状态就会调用self的headerRereshing)

[self.tableOfLocalRecord addHeaderWithTarget:self action:@selector(headerRereshing)]

// 2.上拉加载更多(进入刷新状态就会调用self的footerRereshing)

[self.tableOfLocalRecord addFooterWithTarget:self action:@selector(footerRereshing)];

配置完成之后，用户在进行上下拉刷新的时候将会触发上下拉的处理函数，来进行相应的操作，由于上下拉的处理函数在实现形式上面基本相同，所以在此介绍的时候会将其结合在一起描述，实现的关键伪代码如下：

#pragma mark 开始进入刷新状态

- (void)headerRereshing{

if (segmentConOfLocal.selectedSegmentIndex == 1){ //网络记录才存在上下拉刷新，本地无

[NetworkTool getServerPhotoRecordAfterTime:self.firstTimeStr success:^

(AFHTTPRequestOperation \*operation, id responseObject) {

NSDictionary \*tempDict = (NSDictionary \*)responseObject;

//调用类方法 将解析后的数组返回来

NSMutableArray \*tempArray = [tempDict valueForKey:@"entityList"];

NSMutableDictionary \*dictForNetRecord = [CommonFuncVC translateEntityListJson:

tempArray];

NSString \*timestr = [dictForNetRecord valueForKey:@"firstTime"];

if (dictForNetRecord == nil || [self.firstTimeStr isEqualToString:timestr]) {

/\*提示已是最新记录 如果是上拉刷新则与记录的lastTime作比较，不等则

将该记录插入到本数据源的数组底部，然后重新加载记录列表\*/

}

else{

//记录当前时间 方便上下拉刷新

NSMutableArray \*mArrayOfInsert = [[NSMutableArray alloc] init];

mArrayOfInsert = [dictForNetRecord valueForKey:@"recordList"];

int countOfInsertArray = (int)[mArrayOfInsert count];

for (int i= countOfInsertArray-1;i>=0; i--) {

[arrayOfNetWorkRecord insertObject:[mArrayOfInsert objectAtIndex:i] atIndex:0];

}

//记录当前主数据源中的最新记录的拍摄时间

self.firstTimeStr = timestr; //如果是上拉刷新则是self.lastTimeStr = lastTime;

[self.tableOfLocalRecord reloadData]; //重新加载数据

}} failure:^(AFHTTPRequestOperation \*operation, NSError \*error) {

//第一个异步调用的失败处理，提示网络状况不佳，请检查网络

}];}

[self.tableOfLocalRecord headerEndRefreshing];

//结束下拉刷新，上拉则调用[self.tableOfLocalRecord footerEndRefreshing];}

## 5.4 iOS端本地数据库功能实现

在iOS客户端系统关于本地数据库设计的内容，在上一章的系统设计部分已经介绍了主体的设计方案，按照该设计方案进行系统实现的时候，数据库在系统中的创建以及实现流程如图5.6所示。



图5.6 本地数据库及接口函数说明设计图

本地所涉及到的数据库及数据库表功能实现的关键代码如下：

//创建数据库文件和表文件

-(void)createDatabase{

...

//创建数据库文件

NSArray\*paths=NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);

NSString \*thePath = [paths objectAtIndex:0];

NSString \*file = [thePath stringByAppendingPathComponent:KSqliteName];

//创建用户表

NSString \*createUserTableSQL = @"create table if not exists user(loginId integer,userName text,email text,gender integer, birthday text, telephone integer, isDel integer,userPassword integer);";

//创建用户图片日报记录表

NSString \*createImageTableSQL = @"create table if not exists image(userId integer,imageName text,isPublic integer,takeTime text,uploadTime text,description text,imgLatitude float,imgLongitude float, popularity integer,isUp integer,imgWidth integer,imgHeight integer,takeAddress text,title text,workGroup text, takePicUser text,primary key(imageName));";

... }

## 5.5 本章小结

本章主要的内容是详细地描述了整个系统的实现，它可以说是在系统开发过程中最核心也是最为耗时的一个部分，虽然在前面进行了整个系统的实际需求分析，完整的系统设计，还有更为详细的功能模块设计，但是真正实现的部分才是能够将之前的设计与方案的价值完全发挥出来的阶段，所以这部分是整个系统开发的重点。在本章着重介绍了在系统设计中所提到的系统登录与退出、日常管理、审核公告、地理定位、财务管理、权限管理、签到签退，以及上下拉刷新等实用功能的具体代码实现与界面的最终呈现效果，而且在一开始便对系统所采用网络框架进行封装后的请求类进行了全面而详细的介绍，这些细节均体现了系统在编码实现阶段是完全按照系统设计的模式进行，并且严格遵守合理的编码流程与一致的编码规范进行的。本项目在实现过程中一共包括了自创建实现的49对.h与.m文件，另加1个WholeHeadersFiles.h文件以便于管理项目内的所有头文件。使用到iOS自带或支持的原生以及第三方框架.framework文件23个，第三方开源方法类与功能类.h与.m文件12对，配合设计过程的23个页面来表现和完成所需要的系统主体功能，去除第三方库的开源代码数目，项目累计编写代码接近两万行，最终将整套系统按照既定的方案实现并呈现在了所有的用户面前。

第6章 系统测试

软件测试是所有软件系统开发过程中的一项重要组成部分，测试的第一个目的是用来确认该软件系统所实现的内容是否达到了预期的要求，其质量与性能是否符合实际的使用标准。测试的方式可以分为很多种，比如选择系统在某一个时间点的版本作为测试版本，或者是直接对最终的完整版系统进行测试等等，这些方式各有利弊，根据相关项目的实际情况可以选择不同的测试方案，以此来识别系统的完整度并判断软件的质量。测试的第二个目的是提供软件相关的信息，能够为开发人员或者项目经理回馈有效的信息，不仅仅是局限于找出系统之中所存在的错误，更为重要的是能够分析出这种错误出现的根本原因，还有某些不可控制的错误背后所隐藏的安全隐患等，能够为之后进行软件的风险评估等相关测评提供真实可靠的基础信息材料，并且可以设计出对于可能出现的软件隐患进行修正与完善的方案。针对本系统的实际开发特点，经过精心的考虑后，对本系统采去增量式开发的开发模式，所以功能测试将会伴随着整个系统的各个阶段，并在最后再次集中进行单元测试、集成测试、场景测试以及黑白盒的基础测试等等，以此来保证系统能够正确高效的运行使用，下面本文将从系统在测试阶段所做的功能性和非功能性等方面的工作进行说明。

## 6.1 测试环境

(1) 测试机：MacBook Pro 版本：MBP55.00AB.B03

内存：4GB 1333 MHZ DDR3 处理器：2.3GHZ Intel Core i5

(2) 服务器地址： http://182.92.152.53:8080/NEW\_FindME

(3) 操作系统：Mac OS X 10.10.3

(4) 软件开发工具及版本：Xcode6.3.1

(6) 网络环境：联通宽带网络与模拟多种手机网络环境

## 6.2 测试用例及结果

根据增量式开发的特点，有效的测试工作应该广泛地存在于软件开发周期的每一个阶段当中，通常开发人员在编码阶段就会进行相对严格的单元测试与极限测试，也就是通常所说的白盒测试的一种常用方式。而在整个系统完成之后，测试工作将会把重点扩大到复杂的集成以及系统测试的范围内，这种测试方式将会针对整套系统的全部流程，对每一个流程都会进行特定功能测试，并根据功能特点形成较为全面的测试用例体系，这样使用该体系与常用测试方式相结合的方式来设计整个系统的测试方案。在本文之中，测试工作将会以下面这些方面与测试用例结合进行相关测试。

1. 用户与权限管理模块用例测试

登录、权限及用户管理模块，测试正常登录与不同版本的限制登录，还有用户可见性设置，本模块的功能点测试用例如表6.1所示。

表6.1 用户与权限管理模块测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **用例描述** | **测试方式** | **预期结果** | **测试结论** |
| 1 | 验证企业用户分别使用正确与错误的账号密码是否可以成功登录企业版系统 | 配置服务器地址后，分别输入正确与错误的企业用户的账号与密码，点击登录 | 正确的用户账号密码经过验证后可以正常进入系统，错误的账号密码不可以进入，并提示用户名密码错误 | 通过 |
| 2 | 验证客户分别使用正确与错误的账号密码是否可以成功登录客户版系统 | 配置服务器地址后，分别输入正确与错误的客户的账号与密码，点击登录 | 正确的客户账号密码经过验证后可以正常进入系统，错误的账号密码不可以进入，并提示错误信息 | 通过 |
| 3 | 验证客户的账号密码是否可以成功登录企业版系统，企业账号是否可以成功登录客户版系统 | 配置服务器地址后，在企业版系统中输入正确的客户账号密码，点击登录，在客户版系统中输入正确的企业账号密码，登录 | 客户的账号密码经过验证后不可以正常进入系统，企业的账号密码经过验证后也不可以正常进入系统，并分别提示错误信息 | 通过 |
| 4 | 验证登录后是否成功记住密码，下次本地登录时断网也可以自动登录 | 首次联网登录成功后，退出系统，然后联网登录系统一次，断网登录系统一次 | 在联网与断网条件下均可以进入到系统之中，不过断网情况下会有断网提示 | 通过 |
| 5 | 验证设置某人的记录可见性功能点是否可以正常使用 | 点击我的设置中的项目权限，谁可见我，将对某人可见性开关打开，生成记录上传，可见账号进行查看，开关关闭后重试一次 | 可见性开关打开时，我新上传的纪录对于可见账号是可见的；可见性开关关闭，我新上传的纪录对于可见账号是不可见的 | 通过 |

1. 日报记录与地理位置管理模块用例测试

日报记录管理用例测试，针对日报记录的本地生成、查看、修改与删除，上传到服务器后的网络记录记录的查看与修改，从服务器获取的可见用户记录的查看等功能点进行测评。地理位置管理模块用例测试针对用户进入系统后使用到有关地理位置信息的获取是否准确进行测试，分为进入系统定位、手动重新定位、手机息屏后重进系统定位与生成日报定位四大部分，签到签退功能的测试结果会在后续介绍，在此不作赘述，具体测试过程如表6.2所示。

表6.2 日报记录与地理位置管理模块用例测试

| **序号** | **用例描述** | **测试方式** | **预期结果** | **测试结论** |
| --- | --- | --- | --- | --- |
| 1 | 验证本地记录的增加、删除、上传、修改与查看功能是否可以正常使用 | 拍照生成本地记录，选择项目分组，进行信息添加后上传记录，在本地界面找到该记录进行查看与修改，之后将本记录删除 | 本地记录正常生成，在联网情况下可以上传成功，在本地进行的查看、修改与删除功能均可正常执行 | 通过 |
| 2 | 验证网络记录与日常记录是否可以查看，是否可以对已上传的网络记录进行修改 | 点击网络记录，查看上传内容，并对其内容进行手动修改，点击确定修改后回到网络界面，查看效率里的可以正常显示的对我可见日报记录 | 网络记录可以查看并可以修改，记录项可以上下拉刷新显示新旧记录，对我可见的日常记录可以按照时间逆序、某人记录、日期查找等方式正常显示 | 通过 |
| 3 | 验证进入系统后是否可以准确定位到当前位置，拍照后的记录地理位置信息是否准确，手动更新与息屏后定位 | 进入系统后查看当前位置信息，拍照生成本地记录后查看地理位置信息准确性，手动点击更新地理位置查看显示内容，息屏后重进系统查看显示的地理位置信息内容 | 进入系统后、拍照生成本地记录后、手动更新后以及息屏重进后均可以看到当前准确的地理位置信息 | 通过 |

1. 绩效考勤与财务记录管理模块用例测试

绩效考勤管理模块用例测试，针对用户进行的签到签退次数进行统计，查看其是否满足正常的上班时间，是否发生迟到早退等违规行为。财务记录管理模块用例测试，针对用户对于每个项目的财务记录的查看与添加进行测试，正在进行的项目是可以添加财务记录的，并需要填写相关信息，对于已完成的项目只允许查看财务记录，不再允许增加新的记录项，当某一项目的实际花费数额超出预警值时，需要将金额标红显示来提醒用户，并且系统查看的财务记录数目有限，过多的项目记录项需要在WEB网页端进行查看，以保持系统的稳定性与高效性。

表6.3 绩效考勤与财务记录管理模块用例测试

| **序号** | **用例描述** | **测试方式** | **预期结果** | **测试结论** |
| --- | --- | --- | --- | --- |
| 1 | 验证签到与签退功能是否可以正常使用，地理位置与附带信息是否能够成功传输到服务器 | 进入系统后，点击我的设置，进入签到签退界面，查看时间显示信息，地理位置信息，并签到一次，签退一次，之后多次进行签到签退操作 | 服务器时间正常显示，地理位置信息显示准确，签到签退后分别弹出成功或者失败提示，重复签到签退会提示用户今日已经签过到/退，Web端查看一切信息正常 | 通过 |
| 2 | 验证查看、添加财务记录以及财务预警功能是否可以正常使用，对于已完成项目是否可以添加新的财务记录 | 点击财务管理，查看财务记录信息，查看显示预算信息，点击添加，填写必要信息增加财务记录，查找已完成项目，点击添加查看提示 | 针对未完成项目财务信息的查看与添加均可已正常实现，已完成项目信息可以查看，点击添加后会提示该项目已结束，不可继续添加财务记录等信息 | 通过 |

1. 公告中心、消息中心与审核批示管理模块用例测试

公告中心管理模块用例测试，当后台管理者在后台发布了一条新的公告的时候，需要客户端系统能够及时查询到有新公告存在的情况，并对使用者进行提示，用户可以点击公告中心进行查看，默认只显示未查看的最新50条公告，查看后将该公告状态置为已查。

消息中心管理模块用例测试，当与用户相关的日报记录有了新的评论或者审核批示，客户端系统能够及时查询到有新消息存在的情况，并对使用者进行提示，用户可以点击消息中心进入查看，当用户退出消息中心之后，将对所有新消息进行状态改变，将其所有置为已读状态，从用户新消息队列中清除，直到下一次有新消息记录存在时，再次对用户进行提示，提醒方式是图标改变并且手机震动。

审核批示管理模块用例测试，用户可以在自己上传的网络记录日报上面进行评论，可见我的人也可以对于我的某张网络记录进行审核批示或者评论，随之将会有新评论显示在日报记录详细信息界面，并且将本消息加入到与之相关用户的新消息通知队列中去。

表6.4 公告中心、消息中心与审核批示管理模块用例测试

| **序号** | **用例描述** | **测试方式** | **预期结果** | **测试结论** |
| --- | --- | --- | --- | --- |
| 1 | 验证公告中心的查看功能与新公告提醒功能是否可以正常使用 | 点击我的设置，进入公告中心页面，查看已有公告记录，在WEB端发布一条新的公告，客户端系统保持开启，等待提示出现 | 公告中心的所有公告信息可以正常查看，新公告发布后，很快手机客户端系统将会提醒，右下角出现新公告字样，并伴有手机震动提醒，成功进入查看后一切提示消失 | 通过 |
| 2 | 验证消息中心的查看与新消息提醒功能是否可以正常使用，同时可以检验审核批示功能的可用性 | 使用可见我的账号对我的某条记录进行审核批示，然后登录我的账号进入系统，查看新消息中心，点击进入查看评论后退出 | 消息中心的所有新消息可以正常查看，新消息出现后，很快手机客户端系统将会提醒，系统消息图标出现变化，并伴有手机震动提醒，成功进入查看后一切提示消失，并且可以在记录详细信息界面看到刚才的审核批示记录 | 通过 |

主要功能模块的测试结果完全满足预期结果，其他相应的子模块测试工作均已完成，并且同样满足预期结果，因为实际测试用例数量非常多，所以以上主体模块的测试用例描述将代表其它未被提及的模块及其子模块的测试过程，不再进行赘述。

## 6.3 非功能性测试

本系统的非功能性需求重点在于系统的稳定性与可用性。在实际测试过程中主要是采用有项目相关人员添加数据的方式来检验系统的性能，并使用高并发数的高频率访问服务器的方式检验服务器的承载能力以及响应能力，并一起检测本系统的反应速度与安全性的可靠程度。

## 6.4 测试结论

经过系统所有模块及其子模块的测试结果显示，基于iOS的工程监控系统的所有业务功能齐全，其实际运行结果、稳定性与可靠性均已达到需求分析与系统设计中设定的预期目标，符合相关要求。经过不断的测试与修正，系统的部分逻辑错误与缺陷得到有效改进，整体运行更为稳定，数据保存精确，安全性有所提升，可以满足绝大多数用户对于本系统的要求。

## 6.5 本章小结

作为系统开发的最终也是最关键的一个环节，测试阶段在上述完整而严谨的设计与实现过程中完成。本章针对基于iOS的工程监控系统的测试过程进行了完整而详细的描述，分模块的从用例描述、测试方式、预期结果以及测试结论四个方面进行了详述，并对系统出现的一些严重错误进行了有效修正，同时也对之前设计过程中没有考虑到的因素与细节进行了补充和完善。

第7章 总结与展望

本文所介绍的工程监控系统是基于iOS的应用程序，根据实际市场调研后总结的需求分析进行了相关功能的开发，在指导教师的耐心指导与团队成员的相互配合下，完成了整个系统的设计与实现。系统不仅完成了工程现场相关记录的增加、删除、修改、查询功能及其相关的权限控制，还提供了财务记录、绩效考勤和审核批示等与日常业务息息相关的处理功能，使得整个工程的业务操作流程更为简便，信息也更加准确可靠。

近年来，对于智能移动终端的高度需求与高度依赖的设计开发，使得智能终端的系统应用开发的市场占有率越来越大，各种移动平台也在进行激烈的竞争。本文首先阐述了工程监控系统的开发背景与市场前景，然后介绍了国内外现状以及系统开发所涉及到的关键技术，并根据实际的市场调研结果，从系统的总体需求、功能性及非功能性需求等角度进行了较为全面的需求分析。接着详尽地介绍了系统的总体设计、详细设计以及功能模块设计，并根据相应的设计方案给出具体模块的实现方法，随后进行实际的编码实现，最后按照公司级别的系统测试要求对整个系统进行了较为全面的测试与完善。

在整个系统的开发过程之中，主要使用的编码语言是Objective-C语言，开始时期利用Xcode6.1进行开发与测试，之后使用最新版的Xcode6.3.1正式版进行开发与测试，保证本系统的稳定性。主要使用的网络请求框架是AFNetworking，以及图片请求框架SDWebImage来进行与服务器的数据交互，并使用轻量级数据传输格式JSON来封装数据，进行数据传输标准格式，还有CoreLocation与MAMapKit高德地图SDK的组合使用实现地理位置的确定。系统主要实现了用户管理，准确定位，上传与编辑日报记录，新消息通知中心，团队成员工作记录的查看与审核批示，财务管理，签到签退，安全处理与权限控制，全尺寸屏幕适配等功能。

本系统基本完成了根据实际需求所确定的全部预期目标，不过由于时间方面的限制，以及对相关框架的熟练程度不足，对某些技术的了解与掌握仍旧有限，所以在实际使用过程中，对于一些实际问题的考虑可能还无法尽善尽美，但根据目前的测试结果分析，系统的稳定性与性能还是比较符合要求的。另外，由于工程管理的模式随着时间推进，可能还会出现许多不同的方式与理念，所以系统在今后的发展中还会遇到许多新的问题需要修正，这些都要求系统的可拓展性与鲁棒性达到较好的程度，并能够在今后的实际使用中得到进一步的蜕变。本人也将继续学习iOS的相关主流技术，逐步提高自身技术水平与架构思维，积累更多的开发与测试经验，并不断完善本工程监控系统。

参考文献

1. Qiao-fang Z, Yong-fei L. Research and development of online examination system[C]//Proceedings of the 2012 2nd International Conference on Computer and Information Application (ICCIA 2012). 2012: 936-938.
2. 奎晓燕, 杜华坤, 刘卫国. 大学计算机基础课程考试方式改革的探索与实践[J]. 中国教育信息化, 2014, 6: 39-41.
3. Krasner G E, Pope S T. A description of the model-view-controller user interface paradigm in the smalltalk-80 system[J]. Journal of object oriented programming, 1988, 1(3): 26-49.
4. 戴翔宇.基于MVC模式的Struts框架的研究与应用[D].武汉理工大学,2003.DOI:10.7666/d.y519533.
5. 余银山, 胡正华. 基于Struts2框架的Web应用开发研究[J]. 电脑知识与技术, 2009, 5(16):4224-4225.
6. 胡启敏, 薛锦云, 钟林辉. 基于Spring框架的轻量级J2EE架构与应用[J]. 计算机工程与应用, 2008, 44(5):115-118.
7. Wu Q, Hu Y, Wang Y. Research on data persistence layer based on hibernate framework[C]//Intelligent Systems and Applications (ISA), 2010 2nd International Workshop on. IEEE, 2010: 1-4.
8. MySQL A B. MySQL[J]. 2001:11-19.
9. Canglin W. The Analsis and Comparison Between Browser/Server Structure and Client/Server Structure [J][J]. COMPUTER STUDY, 1999, 5: 025.

致 谢

首先，我需要特别感谢我的导师朱志良老师给予我参与本套工程监控系统的开发机会。同时，在我进行毕业设计的每个阶段都得到了朱志良老师的耐心指导与悉心帮助，在需求调研期间，朱老师一直主动带领我们走进市场，与一些实际的市场潜在客户进行面对面的交流，给予我们进行真实需求调研的锻炼机会，并在之后与我们共同讨论调，提取最符合实际需求的功能点进行归纳总结，然后大家再次讨论确定需要实现的功能，完成需求分析与调研报告。在项目的初期设计阶段，我还需要特别感谢身为研究生的高宇学长，是朱老师在整体架构上面给予我们很多的方向性指导和阶段性帮助，同时高宇学长也以其丰富的开发经验在实际开发过程中多次帮助我们解决了十分关键的问题，大大提高了系统整体的开发效率。在系统实现的过程中，许多技术与架构方面的难题是在朱老师的方向性指导下得以顺利地避过了许多开发弯路，加上刘国奇老师部分问题的方向性指导与阶段性帮助，于海老师对于论文的修改意见，再加上像高宇等研究生学长的技术支持与帮助，系统与相应论文最后才得以顺利完成。在此，我对朱老师、刘老师、于老师以及高宇等研究生学长表示衷心的感谢，谢谢你们无私的帮助与支持。

其次，我要感谢本篇论文中所引用到的各种研究文献以及电子资料的诸位学者与技术人员，在项目的实际开发过程中，本人遇到的许多问题，很大一部分是在各位研究成果的启发下才得以实现，否则我很难完成此篇论文的写作。与此同时，我要感谢东北大学图书馆的全部，是其拥有的丰富资源让我找到了那么多优秀的期刊、杂志和论文等。

然后，我必须要感谢项目开发团队的所有成员。在团队的整体开发过程中，是大家的精诚合作以及相互包容，才使得整个项目的开发进程如此顺利，同时也让我更为深刻地感受到了团队合作的重要性与高效性。不同的团队成员之间的设计思路与解决问题的切入点都有所不同，但是最后都是殊途同归的能够解决问题，在这种问题的多次讨论与激烈争辩，往往很多时候都会带给我许多新的知识与对于某种事物的全新认知，这种感觉十分奇妙，也让我逐渐学会多角度的看待问题。

最后，我要感谢在日常生活中给予我帮助的同学与朋友，是他们让我感受到了大学生活的美妙与乐趣，还有学院的任课教师，是他们的培养与授课让我确立了自己的奋斗目标，现在的我即将完成本科四年的学业，在此向你们表示由衷的谢意。我还要感谢东北大学，感谢软件学院，感谢信息安全1102班，是你们让我遇到本科期间所能够遇到的一切，这对我来说是那么的幸运。