电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

互联网软件开发综合课程设计

设计报告

项目	工业监控预警系统
学生姓名	李逢君、张风啸、向尉、陈思远
指导教师	牛新征

2018 年 6 月 13 日

摘要

本文主要就如何实现工业监控预警系统课程设计项目做一份详细的报告。论文首先简单回顾项目课题的需求，之后，对需求进行简要分析，将项目面向的角色设定为用户、维修人员、管理员三种，并根据不同的角色开发相应的功能。

项目中用户和维修人员的核心功能紧紧围绕着仪器的告警与维修，通过 websocket 及邮箱的方式实现告警与维修进程的主动推送，以一种类似于物流的方式实现对仪器监控的流程化管理。此外，还实现了个人信息管理，仪器管理，维修工单管理，推送管理等必要的管理系统，使项目的设计尽量合理且人性化，项目功能达到闭环。项目中管理员角色的设定用于统筹整个系统的运作，并拥有更高的权限。

确定功能后，根据项目小组成员的能力特长分工为页面设计、后端开发、前端开发、移动端（ios）开发，制定项目开发工作流程图及时间表，开发完成后，每个小组成员将独立撰写分工部分的介绍、难点、不足点与收获，并进行汇总。

最后，论文对项目组实现功能进行截图展示，我们也提供了线上展示的 URL，并将设计图及代码进行开源。

目 录

第一章 项目概述

1.1 项目背景

本课题以构建“互联网+X”融合式综合平台为目标；以 Web 开发，移动端 Android 开发，后台服务器部署等为代表的互联网软件开发技术为主要学内容，并辅以讲解互联网软件技术在某些产业中的具体融合形式。启发学生在产业多元化背景下对“互联网+”扩展模式的思考，旨在激发学生创新能力，培养学生形成“互联网+”思维模式。

1.2 主要任务

基于 B/S 架构，实现一个对模拟工业仪器的监控预警系统

1.3 功能描述

假设工厂里有一些仪器，每个仪器具有一个数值，数值超过一定阈值代表存在异常，需要及时通知检修人员。本系统应实现对仪器的监控，功能如下：
（1）采用程序模拟一批仪器，按照一定策略产生数值；
（2）用户能查看仪器的信息；
（3）用户可以针对每个仪器设置一个告警阈值；
（4）系统能判断仪器的数值是否超过阈值，并自动向用户发送邮件

1.4 预期成果或目标

综合课程设计报告，可运行的程序和代码。

1.5 小组分工概况

原型设计	李逢君
页面设计	陈思远
后台开发	张风啸
前端开发	李逢君
移动端开发	向尉
API	张风啸、李逢君
数据策略	向尉

第二章 设计方案

在开发项目之前，我们先对课题简要的分析与评估。首先项目是面向用户的，设计要让用户满意，就要先确认用户是谁。下面我们首先对角色进行划分，然后再根据用户角色对功能进行更加详述的划分与设计。另外由于没有真实的仪器，因此我们需要对仪器的数据进行模拟，我们将系统的业务逻辑部分和仪器数据产生部分的开发进行分离，使项目的设计更加合理与真实。

2.1 角色划分

项目的主要目的是实现一个工业仪器预警系统并对仪器进行监控，而仪器的拥有者是用户，因此需要设置用户的的角色；因为仪器有可能产生损坏告警，需要系统分配维修人员对告警仪器进行检修，因此需要设置维修人员的角色；另外，除了普通用户和维修人员，还需要系统管理员对整个系统运行进行监管，所以又添加了管理员的角色设置。综上项目的角色分为三种：用户、维修人员和管理员，下面根据角色划分进行功能划分。

2.2 功能详述

2.2.1 用户

- 注册登录模块，实现简单的注册登录功能。
- 基本信息模块，该模块可以设置用户的基本信息，包括昵称、头像、用户密码、个人简介以及最重要的邮箱设置。
- 仪器管理模块，实现仪器的增删改查，仪器的属性包括有仪器名称、图片、类型、型号、参数、使用年限、历史阈值以及告警阈值。
- 维修工单模块，每次仪器产生预警的时候，系统会自动生成一个维修工单推送给相对空闲的维修人员，维修工单一共有四个状态：等待维修人员确认、维修人员已确认、维修人员正在维修、维修完成，用户可以查看历史产生的维修工单，也可以根据工单的状态对维修工单进行查询，获取维修人员的信息。
- 消息通知模块，每当用户的仪器产生告警，或者维修工单的状态发生变化，系统会对用户进行主动的消息提醒，并且发送邮箱进行通知，消息的状态有未读和已读两种状态，用户可以对消息设置已读。

2.2.2 维修人员

维修人员部分的功能在用户功能的基础上进行阉割：

- 登录模块，实现简单的登录功能，维修人员通过管理员进行注册。
- 基本信息模块，该模块可以设置维修人员的基本信息，内容同上。
- 维修工单模块，每次仪器产生预警的时候，系统会自动生成一个维修工单推送给相对空闲的维修人员，如果该维修人员分配到该维修工单，维修人员可以查看该仪器的信息，并根据维修进度更改的工单的状态。
- 消息通知模块，当维修人员得到系统分配的工单，系统会对维修人员进行主动的消息通知，并且发送邮箱进行通知。

2.2.3 管理员

- 登录模块，实现简单的登录功能，管理员通过修改数据库进行注册。
- 用户管理模块，可以查看已经注册的用户列表，可以根据用户 ID 查看用户创建的仪器列表，可以根据用户 ID 查看用户的工单列表。
- 维修人员管理模块，实现对维修人员的添加（注册），可以查看已经添加的维修人员列表，可以根据维修人员的 ID 查看维修人员参与的工单列表。
- 仪器管理模块，实现仪器分类的增删改查，实现仪器模板的增删改查（不同分类的仪器可以创建多种模板供用户创建时参考），根据仪器 ID 查看具体仪器的信息。
- 工单查询模块，根据工单的 ID 查看工单的基本信息，包括仪器信息，所有者信息，维修人员信息。

2.3 数据产生

本次仪器生成的策略构思了三种生成策略，即用三种函数生成数据，对于每一种仪器随机采用一种生成方式生成数据。

1. 使用 sin 周期函数，根据仪器的阈值来设定 sin 函数的幅度值，每次使用函数的生成数据时，加入随机突变因子，即每次生成随机数，当随机数的值小于一定值时就加上一定数值，逻辑如下：

假设

仪器的阈值为 A

随机数 R1 ($\frac{\pi}{4}$~$\frac{\pi}{2}$)

随机数 R2(0~1)

定义一个预定数值 B
定义一个阈值 C (e.g. 0.1)
那么

$f(x) = \text{Asin}(R1) + \text{Bg}(x)$ 其中 $g(x) = \begin{cases} 0, R2 < C \\ 1, R2 \geqq C \end{cases}$

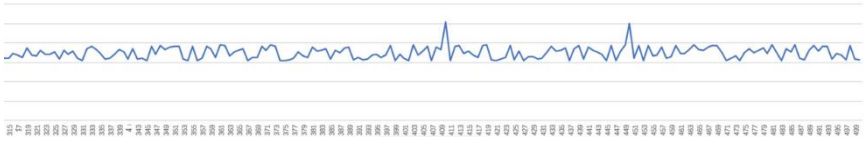


图 1 数据产生策略-1

2. 同样使用 sin 函数生成，但是缩小 R1 范围，事先生成一千条数据，并将这一千条数据排序，每次顺序取数据，同时取数据的时候一定概率加上突变因子。生成数据的例子：

图 2 数据产生策略-2

3. 采用分段函数来生成数据：

$$f(x) = \begin{cases} x^2, -5 < x < 0 \\ \sqrt{x}, 0 < x < 5 \end{cases}$$

图 3 数据产生策略-3

第三章 小组分工

本次 web 项目的开发我们采用的是比较标准的 web 项目开发工作流。产品经理对产品需求进行分析和评估，构思一套业务逻辑方案并且画出原型图将方案可视化，其他组员对该方案的实现进行评估讨论，方案或将做出一定的调整。原型设计完成并确认逻辑无误后，设计可以开始根据原型图进行美化设计，与此后台开发根据原型图上的逻辑进行开发。当设计完成后，前端（包括 web 端和移动端）根据设计图进行页面展示的代码实现，页面完成后，前端可以使用 mock 数据对页面的逻辑进行开发，实现前后端分离以进行并行开发。待前后端基本开发完毕后，由后端为主导对前后端交互 API 进行定制，将前后端逻辑调通，最后测试无误后部署上线。另外，前后各端在开发阶段均使用

github 进行版本控制。

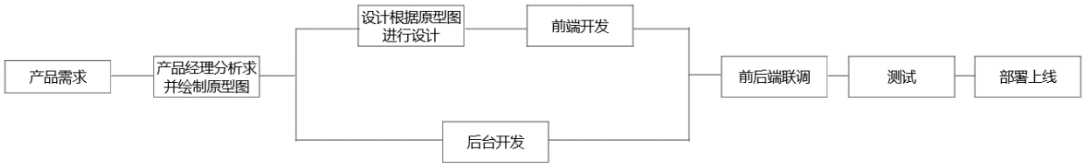


图 4 项目工作流示意图

3.1 原型设计

项目的原型使用 Mockplus 对 web 端和移动端的业务逻辑进行设计，对页面的跳转及展示进行设置与说明，使用日期对版本进行控制，最后导出 HTML 演示，即可以在浏览器上打开原型设计图并具有简单的交互功能，以供设计和后台开发参考。

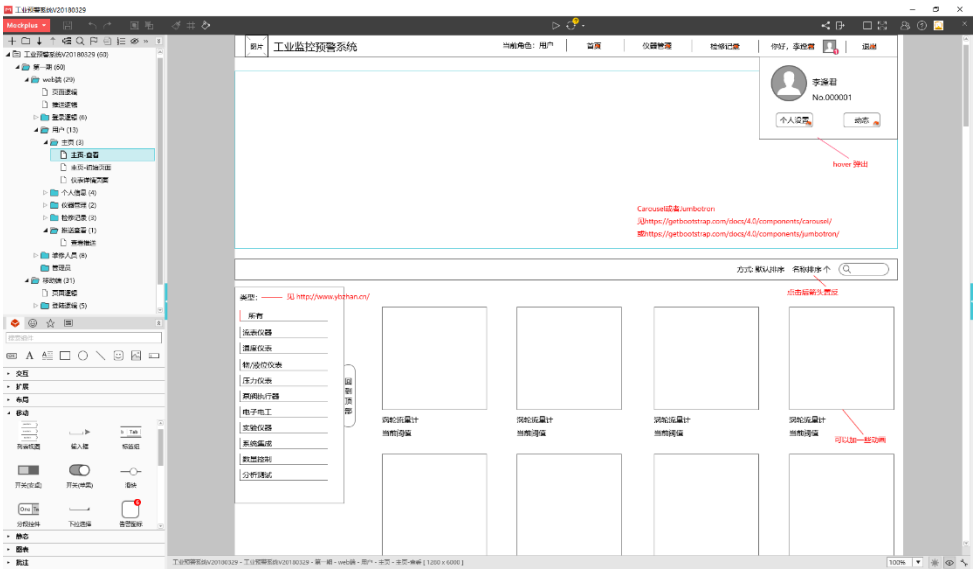


图 5 产品原型设计 web 端示意图

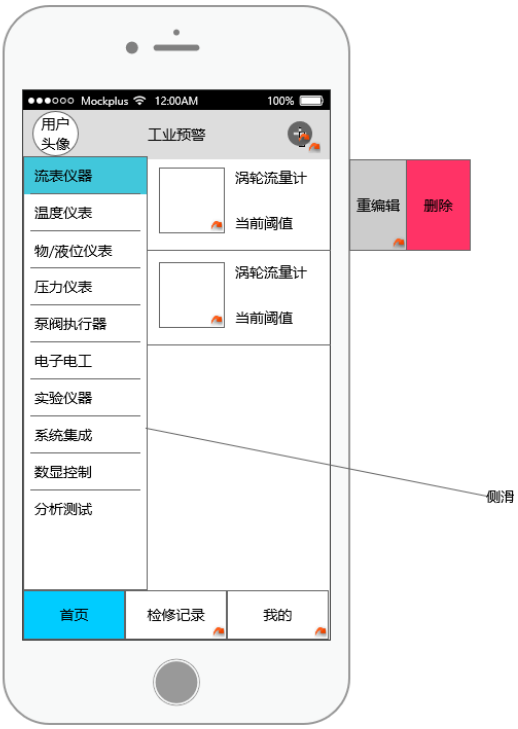


图 6 产品原型设计移动端示意图

3.2 页面设计

3.2.1 介绍

在保证原型图逻辑不变的基础上使用 Photoshop 等工具对原型图进行页面设计与美化，对设计原图进行必要的标注如字体、色号等，对原图和标注进行图片导出，对于无法直接在网页上实现的图片，使用 PS 抠图后按照 png 格式进行导出。最后将设计原图 PSD，效果图，标注图，png 格式的切图一并打包发给前端。

在进行“工业预警”项目的 UI 设计时需要充分考虑布局的合理化问题，遵循用户从上而下，自左向右浏览、操作习惯，避免常用业务功能按键排列过于分散，以造成用户鼠标移动距离过长的弊端。同时，在不同界面的设计当中，多做“减法”运算，将不常用的功能区块隐藏，以保持界面的简洁，使用户专注于主要业务操作流程，有利于提高软件的易用性及可用性。

3.2.1 布局

- 菜单：在“菜单”设计中，设计者尽量保持了菜单简洁性及分类的准确性，避免菜单深度超过 3 层，以免用户对信息处理产生困惑。同时，菜单功能通

过打开一个新页面来完成，通过在菜单名字后加入链接来实现。

- 按钮：本设计中，确认操作按钮放置于弹窗左侧，取消或关闭按钮放置于右边。
- 排版：页面的排版布局上，所有文字内容排版避免贴边显示（页面边缘），尽量保持 10-20 像素的间距并在垂直方向上居中对齐；各控件元素间也保持至少 10 像素以上的间距，并确保控件元素不会紧贴于页面边沿。
- 滚动条：本设计在页面布局设计时尽量避免了出现横向滚动条，最大程度上使得用户视角在页面中部停留，像素在 1024px 像素之内停留。
- 页面导航：本设计在页面比较显眼的位置设置了面包屑导航栏，让用户知道当前所在页面的位置，同时，明确导航结构，
- 信息提示窗口：信息提示窗口应位于当前页面的居中位置，并适当弱化背景层以减少信息干扰，让用户把注意力集中在当前的信息提示窗口。设计选用在信息提示窗口的背面加一个半透明颜色填充的遮罩层。

3.2.2 页面可读性设计

- 文字长度：设计中采用部分显示，使得阅读更加流畅。文字长度，是仪器详情界面设计中比较重要的考虑因素，太长会导致眼睛疲惫，阅读困难。太短又经常会造成尴尬的断裂效果，为防止断裂严重的影响阅读的流畅性。
- 空间和对比度：本设计采用选择一个最小的文字大小的 150%为空间距离，因此足以留下足够多的空间。字符同线路长度，间距也非常重要。因此每个字符之间的空间至少等于字符的尺寸，当每一行中读取大段的文字，且线路长度过多或线之间的空间太少，都会造成理解困难。
- 对齐方式：在主页面的仪器展示中，设计采用中间对齐的方式，并简要列举仪器属性。文本的对齐相当重要，可以极大地影响页面的可读性。
- 在推送查看的页面设计中，采用左侧对齐的方式，符合用户的阅读习惯。

3.2.3 收获

本次网页 UI 设计，对于自身来说是一个比较大的挑战。Adobe Photoshop 的使用也在这次开发中得到了锻炼。在设计中：整体布局、文本设计、导航栏设计、背景选择等，都是需要考虑的因素。例如：

1. 在仪器信息查看中，有大段文字介绍仪器信息，此时需要运用“做减法”的原则，将大段文字省去，以免给展现造成冗余、繁杂的文字信息。
2. 在推送查看中，设计选用了面包屑导航，让用户实时跟进维修信息。给用

户创造较好的交互体验。

3. 在仪器管理的页面设计中，选择平铺的设计，直观简洁，使单位面积展现的信息最大化，最大程度上提升页面的可读性

最后，在本次开发中，多次采用批处理及技巧，智能变换，智能对象处理，icon 设计，以及图标设计等。

图 7 主页设计图

3.3 后台开发

3.3.1 介绍

后台开发使用工具 idea，用 SpringBoot 作为主要框架，集成了 SpringDataJPA，RabbitMQ，Quartz，Shiro 等框架。

数据库使用 MySQL 存储主要信息，redis 进行缓存。

权限控制主要通过 Shiro 的集成实现，完成了各方面细粒度的权限校验。

后台架构设计：

- 考虑到生产环境中数据端与系统端的分离，为减少系统耦合，将整个后端分为两部分，一部分为 user 端，一部分为 machine 端。
- user 端采用后台开发中常见的分层架构设计模式，将整个系统分为 controller（API 控制转发，数据返回层），service（业务逻辑处理层），repository（数据库交互层），entity（实体类层）和其他工具配置层。
- machine 端功能不复杂，主要进行数据的模拟发送功能，实现为通过一个定时任务执行数据的产生与发送功能。
- 两端进行交互的逻辑为用户端创建仪器后，通过一个异步线程执行数据的绑定工作（具体为发送一个带有参数的 http 请求）。在 machine 端收到请求后对数据进行定时发送。

3.3.2 难点

- 与客户端进行 WebSocket 的数据推送。
- 设计时对仪器数据与用户端分离的设计考虑。
- 复杂的业务逻辑较难处理。

3.3.2 不足点

- 在设计考虑时忽视了软删除，导致后台逻辑中的许多删除由于关联关系无法正常处理，没有做好这方面的协调。硬删除影响了部分逻辑的实现。
- 在权限方面，虽然粒度很细，但是在很多地方没有做到较好的抽象，部分权限管理与接口之间强关联。不能动态的配置权限，一部分地方无法对代码进行复用。
- 整个项目的设计有些许不合理之处，主要存在于我们对产品的定位和产品的使用场景认识不足，无法完好的应用到实际场景中去。
- 开发过程中产品需求中不合理的改动，也影响了开发的进度，造成了许多时间浪费与资源浪费。

3.3.3 收获

实践了 Shiro，RabbitMQ，Quartz 等框架与 SpringBoot 的集成。做到了权限的细粒度控制和数据的合理发送机制。数据的发送与处理能承载更多的负载。

对整个后台的逻辑部分进行了合理的抽象，使得一个接口的调用流程较为清晰易懂，便于后续进行各种开发和修改。

学习了与前端进行 WebSocket 的数据推送，更为深刻的认识与了解了 WebSocket 协议。

对一些耗时较长的工作，使用了线程池中的线程进行处理，使得与客户端的交互更为便捷，是的用户体验得到了提升。

使用了各种设计模式，例如在处理数据的位置采用了生产者消费者模式（machine 端将数据发送到消息队列中，再由我 user 端的 consumer 进行消费），使得系统能处理更多的并发量。

总结：注意到的一些设计方面的东西，系统的设计，架构，数据库设计等方面更为熟练。

3.4 前端开发

3.4.1 介绍

前端开发使用工具 webstorm 进行开发，利用 vue 及以生态系统支持的库构建以数据为驱动的 WebApp。

逻辑开发使用 vue-cli 进行脚手架的快速搭建，使用生态库 vue-router 做前端路由的搭建，使用 vuex 做数据状态的管理，使用 axios 进行前后端的交互，并且对 axios 进行封装。

样式开发使用了预处理器 sass 和后处理器 postcss，提高复用性和跨浏览器的兼容性。为了减少设计的负担，管理员部分的页面使用 element-ui 的 UI 框架进行开发，另外使用 particles 库对页面进行美化，以及使用了 echart 进行仪器历史数据的可视化。

对开发环境和生产环境进行区分，使用 webpack 做前端工程的构建工具，集成了代码压缩、图片压缩、图片转 base 64、sprite 图、eslint、ES6-babel 转化等功能，减少大量不必要的精力浪费。同时，使用了 webpack-dev-server 进行热更新，提高开发效率。最后生产环境打包生成一个 SPA（单页应用）进行部署。

3.4.2 难点

- Vue 模块化和组件化的思想和设计，包括子父组件的数据通信，事件传递，插槽，就如何提高复用性和灵活性需要深度的思考和不断的尝试。
- 如何做好函数节流和函数去抖，动画减少的回流与重绘，避免强制重排等，提高页面性能，以用户体验为中心。
- 主页部分滚动加载的实现，对于多条件过滤的分页实现。
- 使用 websocket 与 spring boot 的交互(sockJS 和 stomp)的使用。
- 前后端联调如何解决跨域（浏览器同源）问题。
- 等等

3.4.3 不足点

- 对于页面组件化管理有些混乱，特别是最初开发的几个页面中，组件化思想理解不到位，有几个不必要的模块被抽象为组件，导致维护起来比较困难。
- 页面的自适应不是做的非常好，在低分辨率上面的显示不是很理想，另外由于有移动端 APP 的开发，所以这次开发没有做响应式。
- 还有一个可以提高的地方就是可以使用服务器渲染，一来是可以增加首屏速度，提高用户体验，二来更加有利于 SEO。
- 没有实现模块动态的加载，导致第一次打开应用时间比较长。

3.4.4 收获

本次开发首次尝试使用了 vue 全家桶对 web 应用进行开发，一个非常直观的感受就是，这种以数据驱动的框架开发速度不是使用传统 js 框架比如 jquery 或者原生 js 进行开发能够比拟的。另外使用 webpack 作为前端项目的构建工具，里面集成了许许多多方便的工具减少了很多开发以外的时间与精力的开销。

单页应用的带来的用户体验是多页应用远远不能比拟的，虽然整个 webApp 最初加载的时间比较长，但是一旦加载完之后使用该网站就像使用一个手机 App，而不仅仅只是一个网页，使得用户体验得到提升。

自己尝试封装了模态弹框 modal 作为全局组件，也将一些共同的代码进行提取，对于组件化和模块化的开发有了更深的理解，对于交互部分的封装与设计借鉴学长的方式，使得交互接口更加具有复用性。

另外，对于前端工作流更加熟悉，开始慢慢了解一些前端工程化的知识，对前端打开的方式和使用更加高效了。

3.5 移动开发

3.5.1 介绍

移动端 iOS 端使用苹果官方指定 ide: Xcode 进行开发， 使用官方图形界面框架 UIKit 构建图形界面交互 UI。iOS 端主要负责仪器用户和仪器维修人员的逻辑构建。网络请求发送使用异步请求框架 AFNetworking 并自己对本次涉及到的 http 请求进行封装，减去自己开网络线程的麻烦。网络图片下载使用异步下载缓存框架 SD_WebImage。简化了图片在 cache 和硬盘做缓存的工作。应用本地化持久化数据保存使用原生 plist 文件，UserDefaults 和 FileManager 存储数据到本地，由于不用存储大量数据到本地，就没有使用 sqlite 进行本地数据库构建。使用 Charts 框架对仪器数据进行绘图。所有第三方框架管理使用

cocoapod，简化了自己编译第三方框架并加入本项目的过程，并可以简单实用 podfile 文件添加和删除第三方框架。

3.5.2 难点

本次开发难点主要在于向用户发送消息，当用户的仪器超过报警阈值后，后台会向用户发送消息，如果由客户端采用 http 向服务器固定一段时间拉取消息可能回导致大量的网络流量消耗。所以消息的发送我们采用 web socket 实现。使用 web socket 实现长连接，当服务器段产生消息后由服务器向用户段推送。

3.5.3 不足点

- 本次开发的时候有些部分设计不合理，导致联调时修改 bug 时很难修改。
- 本次开发使用了较多第三方框架，以后有机会应该更多的自己设计一些框架。
- 本次开发主要为了展示设计逻辑，并未太注意交互 UI 的设计。

3.5.4 收获

- 加载页面，渲染数据时，首先应该从本地持久化数据中获取数据进行渲染，然后再从服务器拉取数据，再根据服务器的数据更新 UI 数据和更新本地数据，期中更新 UI 应该放在主线程中，更新本地持久化数据应该放在其他线程中。
- 和后台交互时，从后台拉取到的数据有些可以抽象成一个类时，应该抽象成一个对象，并将对象的属性用枚举表示出来，这样可以方便从后台拉取到的数据中获取数据。例如：抽象出来的 Device 类
- 将网络层面独立开，即与后台交互的部分与其他逻辑分离，单独抽象成一个类。
- 将重复的小代码段用 NSSnippets 保存起来，即将小段代码保存成模版保存在本地。或者使用函数封装起来。
- 处理 TableView 分页时，直接将从服务器拉取到的分页号和内容组成键值对存储到 Dictionary 中，不要将所有分页都组合到一个 Array 里面（这样可能导致本地逻辑不好处理）
- 多使用工厂模式和观察者模式。在本次课设中使用到的工厂模式的地方：

  FilePathFactory　　// 用于生成文件保存路径

  AlertFactory　　// 用于生成各种 alertcontroller

  DateFactory　　// 用于生成时间日期
- 使用到观察者模式的地方：

NotificationManager // 用于接收服务器发来的通知，并向其观察者发送通知。

整体设计应该严格遵守 MVC 或则其他标准化的移动开发设计模式。在同一个 ViewController 中应该多使用函数模块。

- 将其他项目也可能用到的代码单独抽象出来，便于其他项目的复用

3.6 前后端交互

3.6.1 假数据

在后端开发完成之前，我们使用线上的 mock 服务进行后台数据模拟，Easy Mock (https://www.easy-mock.com)是一个可视化,并且能快速生成模拟数据 的持久化服务。

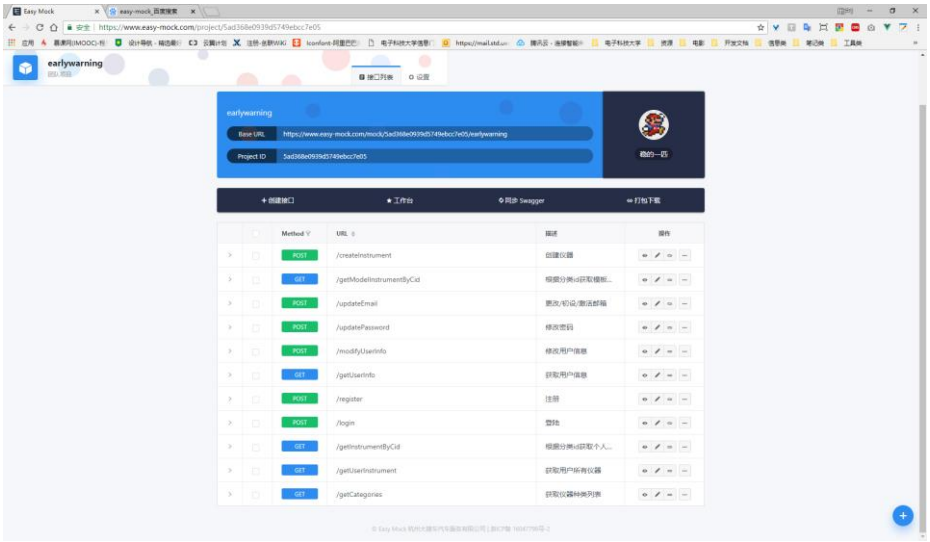


图 8 easy-mock 示意图 1

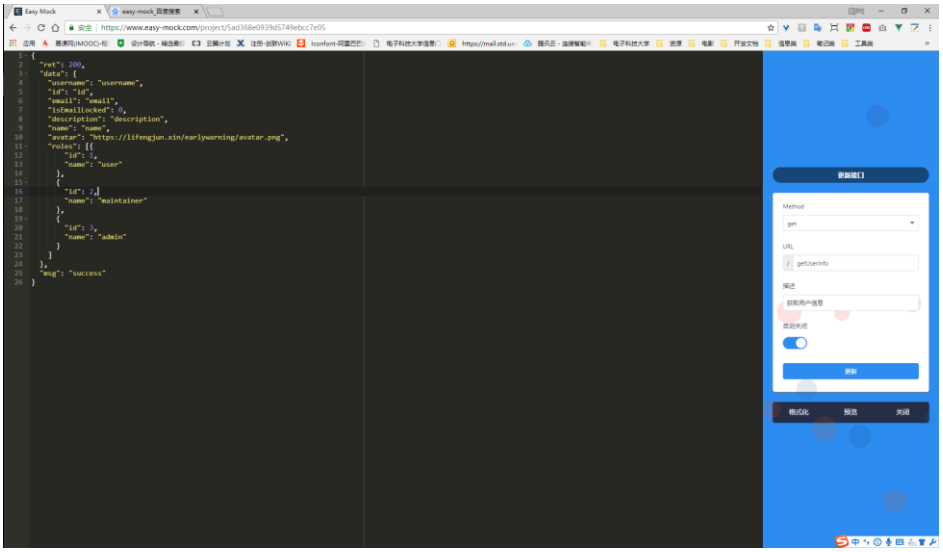


图 9 easy-mock 示意图 2

3.6.2 API 接口

前后端文档的定制使用 Cmd Markdown 编辑阅读器提供的服务，URL 为 https://www.zybuluo.com/AlexZFX/note/1082854

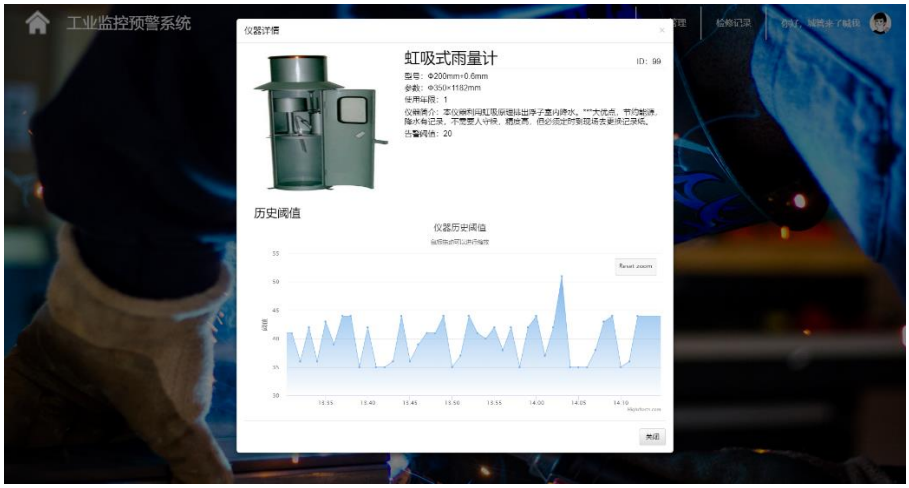
图 10 线上 API 文档示意图

第四章 功能实现

4.1 web 端

- **用户/维修人员**

图 11 注册登录


工业监控预警系统
仪器详情
虹吸式雨量计
ID：99
历史阈值
仪器历史阈值


工业监控预警系统
当前角色：用户
仪器管理
检修记录
关键词搜索
环境监测仪器
方式：默认排序
ID排序
所有
环境监测仪器
温度仪器
生命科学仪器
实验室设备仪器
流量仪表
泵阀执行器

图 12 仪器查看

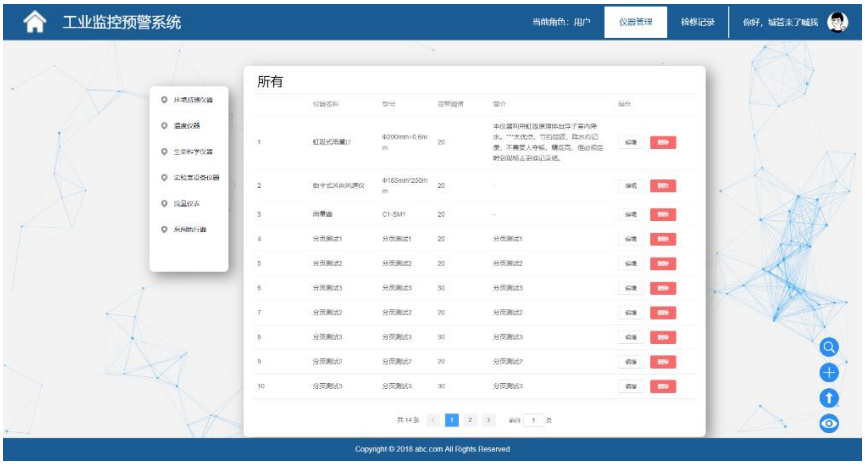


图 13 仪器管理

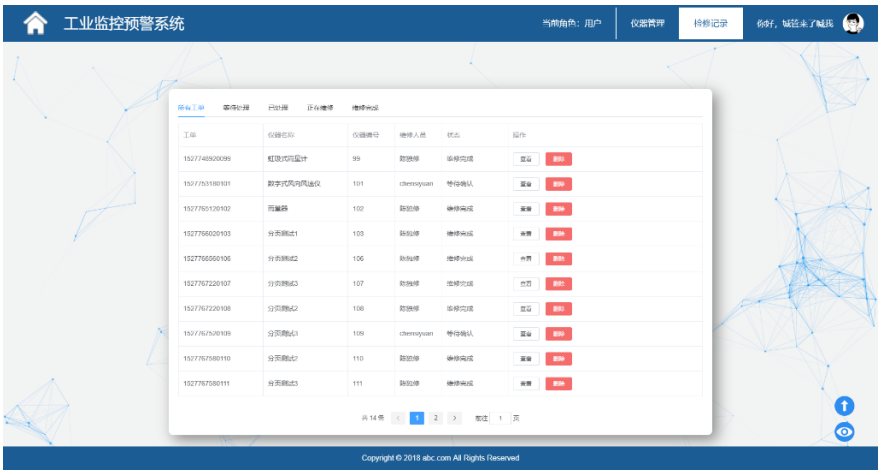


图 14 检修记录

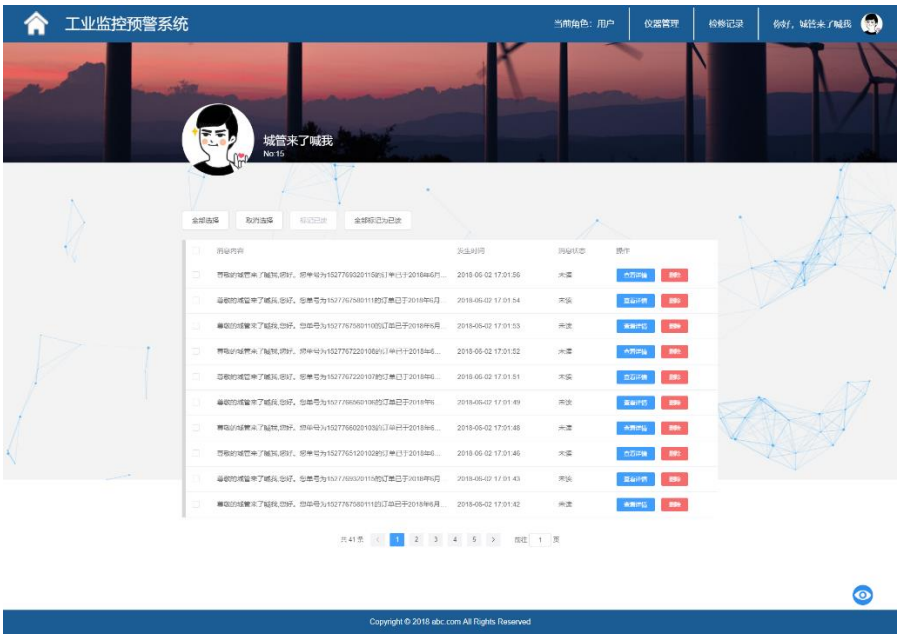


图 15 消息记录

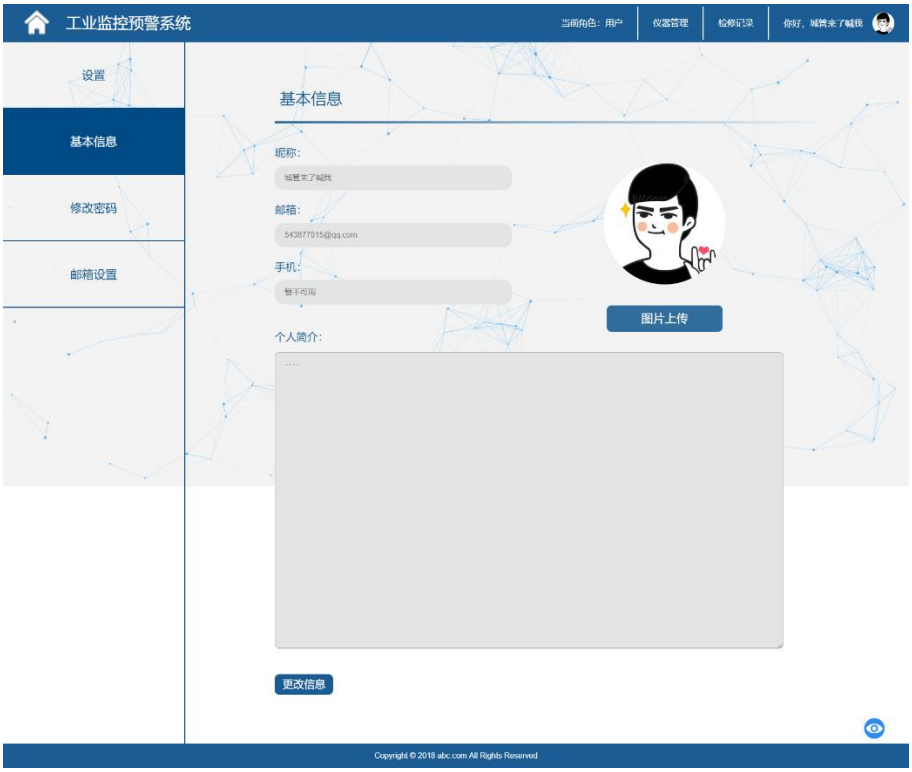


图 16 个人信息设置

- **管理员**

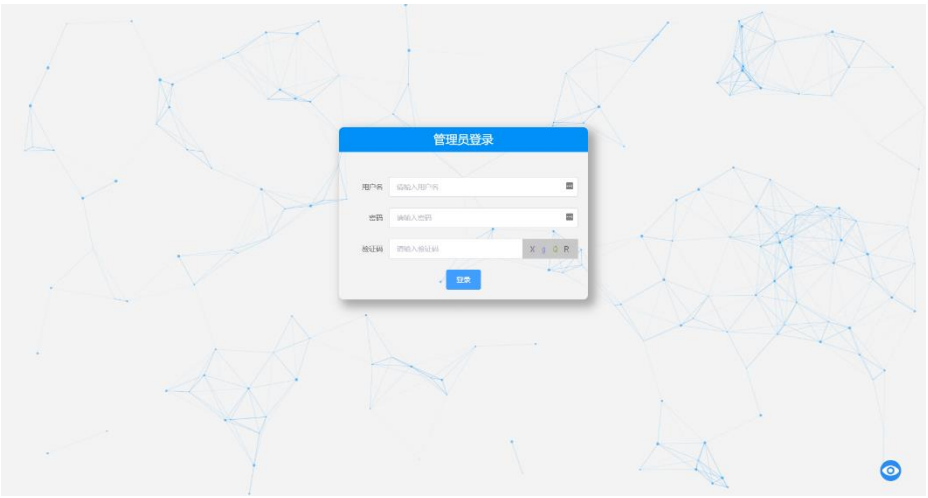


图 17 登录

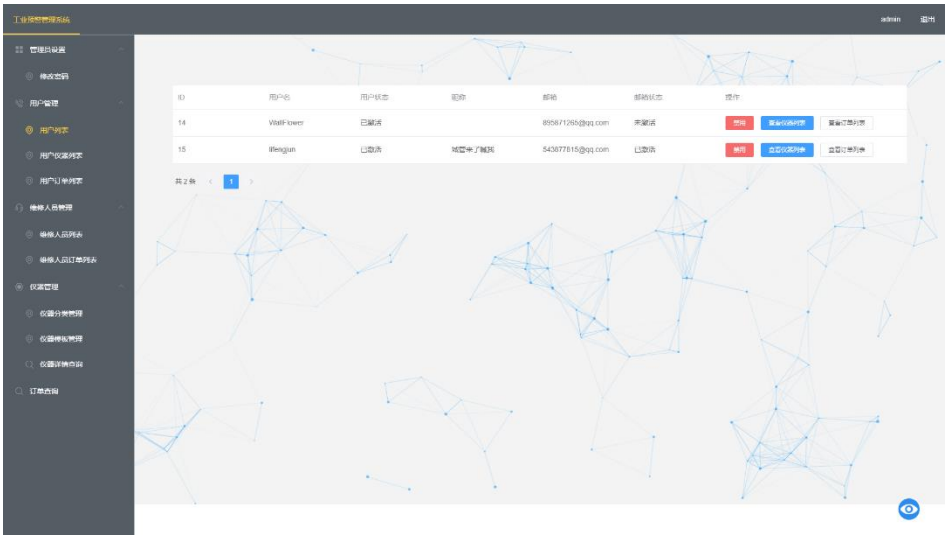


图 18 功能模块

4.2 移动端

图 19 登录注册

图 20 仪器管理

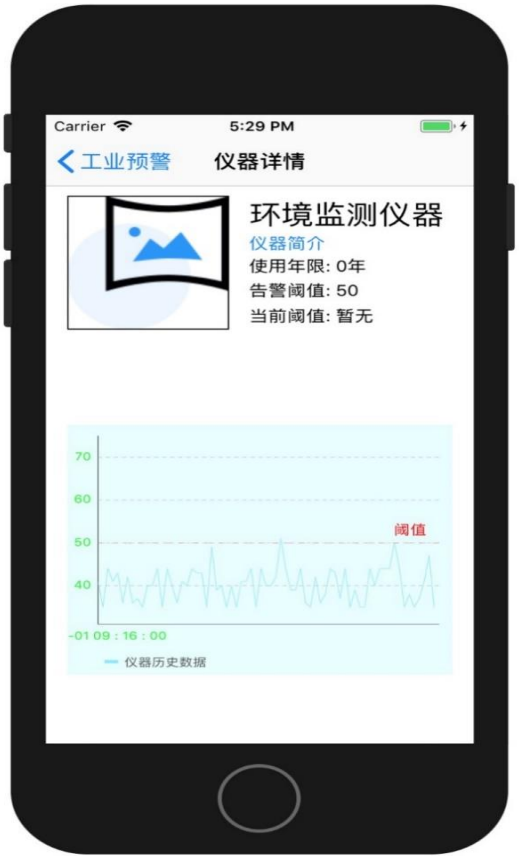


图 21 仪器数据查看

图 22 订单管理

图 23 维修人员界面

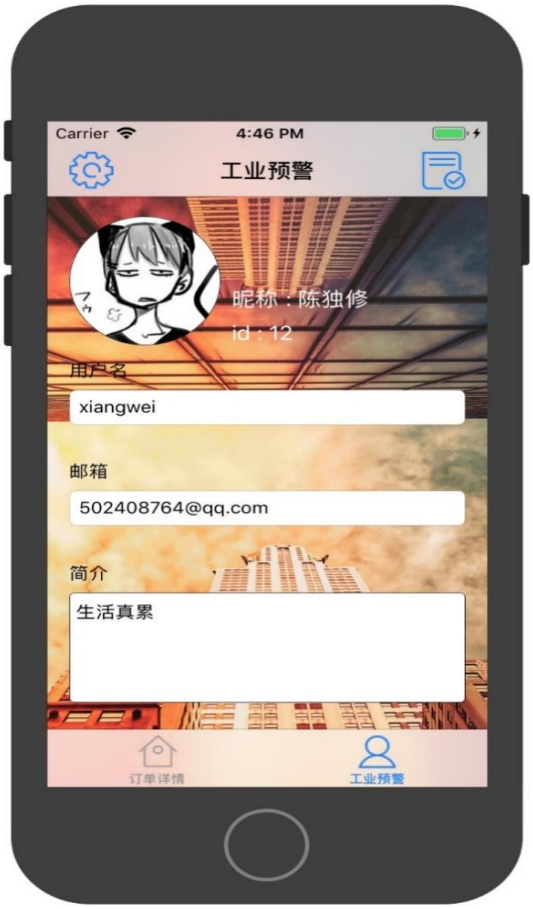


图 24 个人信息设置

第五章 实现结果

5.1 线上部署

Url：http://early.lifengjun.xin/#/login

5.2 设计及源码

原型：链接：https://pan.baidu.com/s/1L6Fu5xGhxvL-r5zqQ5BShQ 密码：e2j1
设计：https://pan.baidu.com/s/1_7K_x9npP1WoZ6H-nguSHQ
后台：https://github.com/AlexZFX/earlywarning
前端：https://github.com/543877815/earlywarning
移动端：https://github.com/SwordAndTea/EarlyWarning
API 文档地址：https://www.zybuluo.com/AlexZFX/note/1082854
Mock：https://www.easy-mock.com/project/5ad1831570181a3cc827d770

第六章 总结

6.1 项目亮点

本次课设我们在原有的任务基础上对功能进行拓宽，使功能更加完善并尽量实现功能闭环。通过三种角色的设置，使整个项目更加符合实际场景。另外除了 browser 端和 server 端的开发，我们还加入了移动端的开发，并且技术栈上均使用了比较前沿的技术进行开发。在开发的过程中十分强调项目后期可拓展性，并且具有良好的版本控制意识和团队协作意识，因此在项目整体架构上更加灵活，并且具有很好的健壮性和可维护性。

6.2 优化与改进

各端优化与改进的建议已在小组分工处提及，此处不再复述。

对于整个项目而言，项目的逻辑可以设计更加简洁，部分赘写的功能可以简化进行。

对于前后端交互的 API 定制，应该使用更加规范的标准制定，比如 RESTful 标准，使开发更加规范。

6.3 结论

本次项目达成课题的要求。通过本次综合设计的开发，学习到了许多互联网软件开发相关相关知识，对“互联网+”思维模式有了更深的思考。