**基于SSM的工单系统设计**

# 一、系统模块设想

## 1.系统分析

    "机房报修管理系统"需要实现的操作就是：报修人扫描二维码报修电脑—>后台管理员获得报修信息—>管理员回应已收到报修请求—>后台管理人员派人维修—>维修完成后管理员写维修经历并把表单转为历史表单。

[CheungChingYin的博客\_CSDN博客-学习笔记,机房报修管理系统,SpringBoot领域博主](https://blog.csdn.net/qq_33596978)

## 2.功能分块

基于以上的操作需求，我们能够做以下功能分块：

**管理员模块**

能对于管理员进行管理，能够使用管理员进行后台管理登录。

**维修工单模块**

对报修人提供的报修信息进行记录，包括出现的问题，提交时间，维修状态等。

**历史工单模块**

和维修工单模块差不多，但是精简了一些不必要的信息，如报修日期；新增管理员对此次维修的备注。

**实训室模块**

能够记录实训室电脑数，属于哪一栋实训楼。

**实训楼模块**

为实训室提供所属实训楼。

# 二、数据库设计

    讲完了对系统模块的设想后，就需要开始认真设想数据库了，设计数据库需要遵守一个原则，那就是不能有太大的冗余，例如实训室和实训楼模块，需要在实训室设计一个字段，用于表示实训室属于哪一栋实训楼，一般来说我们会直接在实训室中新建一个所属实训楼字段building，然后把该实训楼直接填进去。

    这样做确实可行，但是这样做会出现两个问题：一是数据冗余，试想一下，一大堆实训楼A、实训楼B、实训楼C的数据在你的表中重复出现，一旦数据量多了，数据库占用的空间就会很大；二是耦合程度太高了，不利于以后的修改，一旦我实训楼A变成了教学楼A呢？那样需要改动大量数据，显得费时费力。

    所以说我们应该把实训楼和实训室分为两个对象，由于实训室属于实训楼，我们能够以外键的形式将两个表连接起来，这样实训室的building字段能够用实训楼的ID代替，这样依赖减少了数据冗余，毕竟相对于文字来说，数据所占的空间更少；当需要修改实训楼名称的时候质询要修改实训楼表中相关实训楼名称即可，降低了耦合。

## 1.数据表

### （一）管理员表administrator

    上面提到我们需要一个管理员模块用于管理及登录，所以我们把管理员单独一个表，以下为设计字段。

admin\_id（管理员唯一ID）

admin\_password（管理员密码）

admin\_name（管理员姓名）

admin\_phone（管理员手机号）

role\_id（管理员角色）

admin\_email（管理员邮箱）

### （二）角色表role

由于管理员有两种角色：普通管理员和超级管理员，所以需要建立角色表，以下为设计字段。

role\_id（角色ID）

role\_name（角色名称）

### （三）维修工单表orders

在设计当初，曾经想过把用户分离出来独立成一个用户表，但是想到用户并不是固定不变的，每次报修人的信息都是不同的，一般只会使用一次用户信息或者根本不需要，如果再设计一个用户表，相当于是绕了一圈，做了无用功，所以用户表直接和工单包合在一起了。

order\_id（维修工单ID）

problem（工单问题）

computer\_number（损坏的电脑编号）

class\_id（所属实训室ID）

building\_id（所属实训楼ID）

status（工单状态，0为未受理，1为已受理）

submit\_time（工单提交时间）

images\_path（用户上传照片的路径）

admin\_id（受理工单的管理员编号）

user\_name（报修人姓名）

user\_phone（报修人电话）

user\_email（报修人邮箱）

### （四）历史工单complete\_order

历史工单的作用是记录历史工单，当管理员完成维修后，就会把维修工单上的部分信息迁移到历史工单中，由于历史工单中没有报修人相关的字段，以保证报修人的隐私问题。

order\_id（历史工单ID）

problem（工单问题）

remark（工单备注）

admin\_id（接手管理员编号）

complete\_time（完成工单日期）

image\_path（用户上传图片路径）

class\_id（所属实训室ID）

building\_id（所属实训楼ID）

computer\_number（电脑编号）

### （五）实训室class

实训室表用于记录所有实训室信息，如所属实训楼，拥有的电脑数，可用、不可用的电脑数。

class\_id（实训室ID）

class\_name（实训室名称）

building\_id（实训楼ID）

computer\_total（实训室电脑总数）

computer\_enable（实训室可用电脑数）

computer\_disable（实训室不可用电脑数）

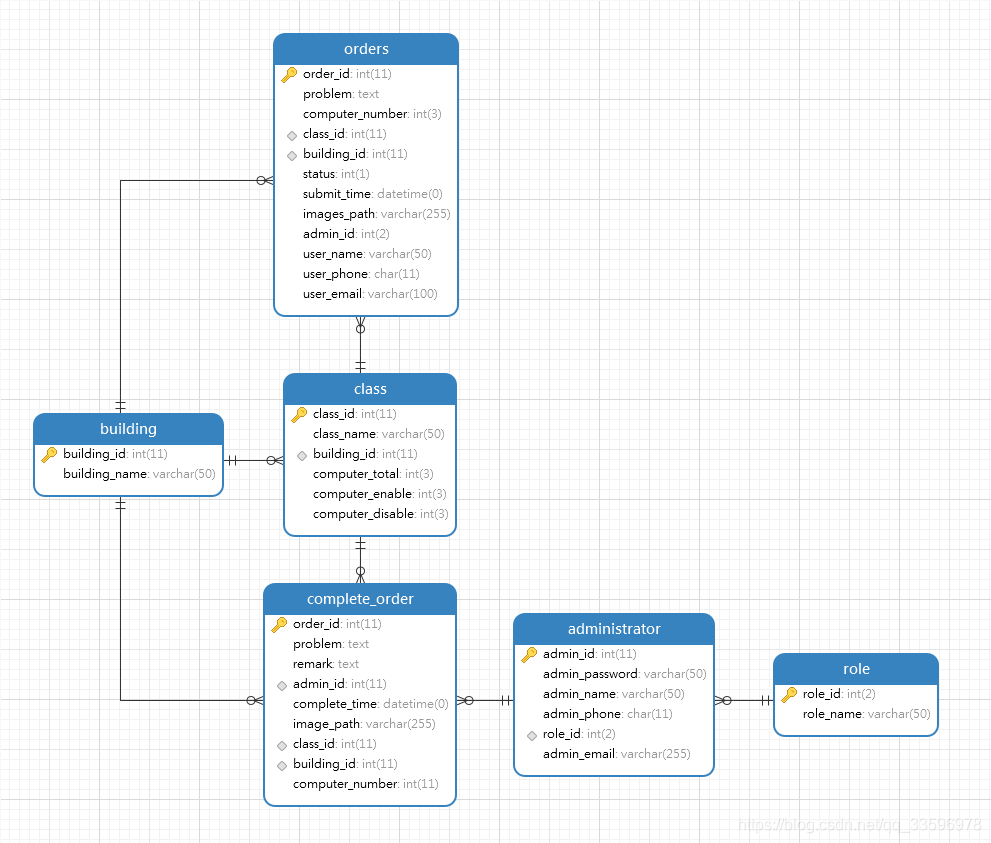
### （六）实训楼building

实训楼表用于积极记录实训楼所有信息

building\_id（实训楼ID）

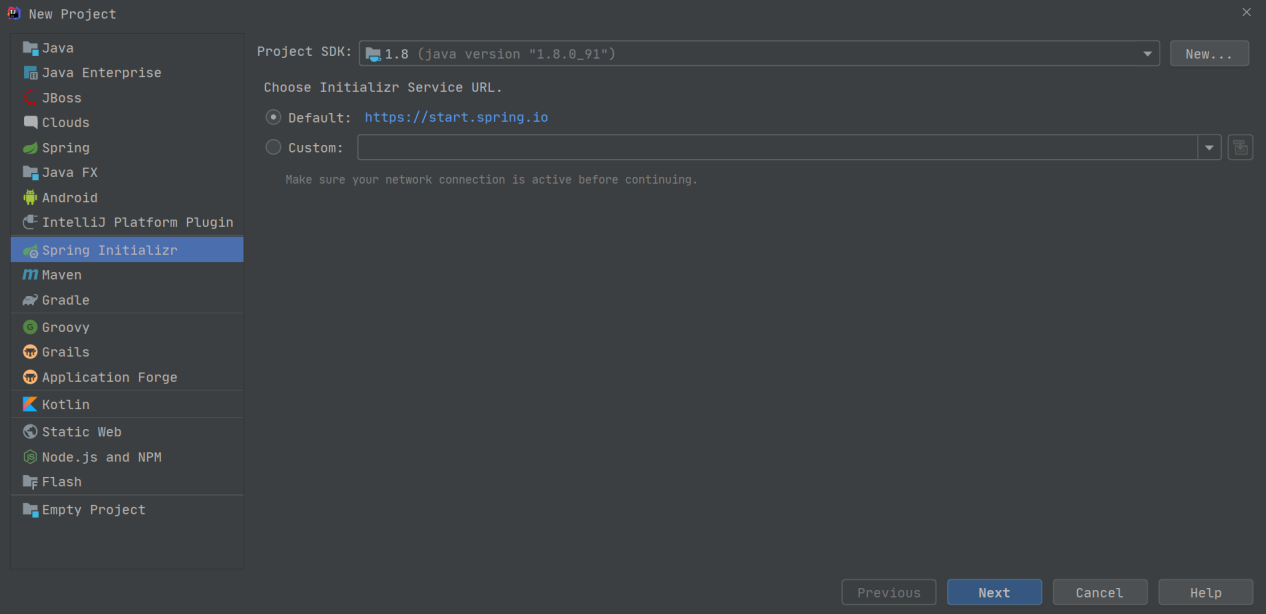
building\_name（实训楼名称）

1. 数据库关系E-R图

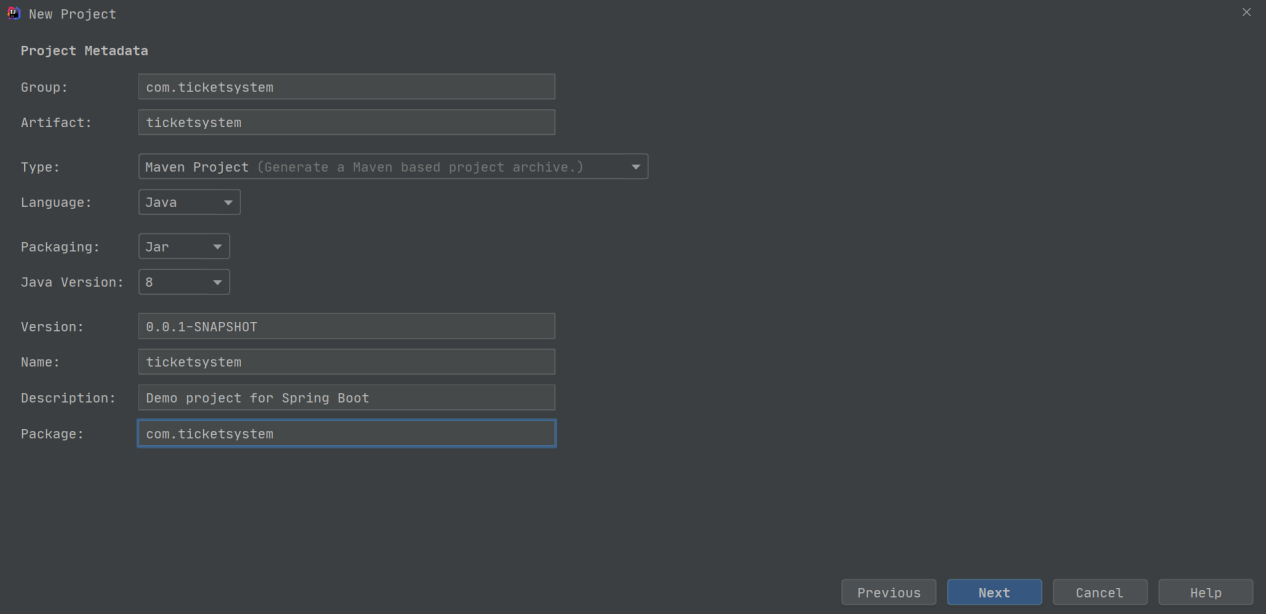


# 项目搭建

## 使用IDEA创建一个SpringBoot或Maven项目



下一步创建名为ticketsystem的项目



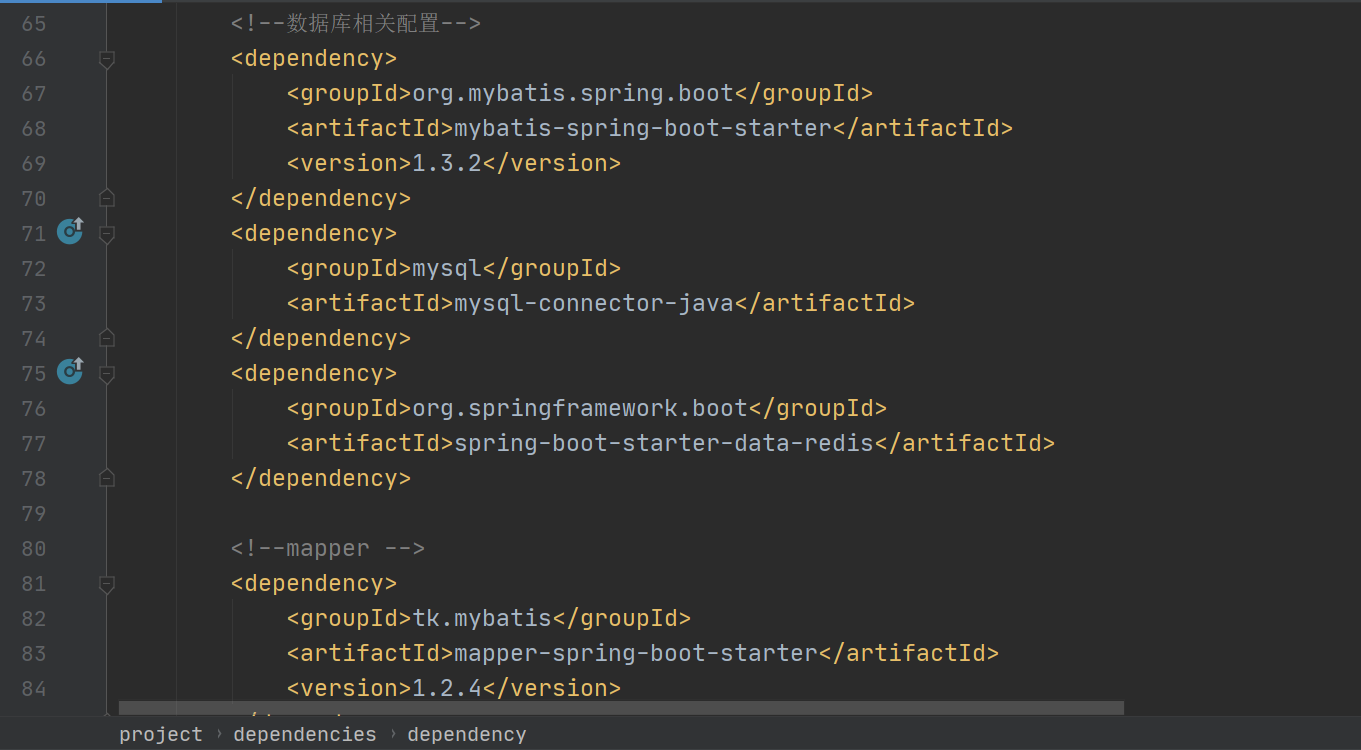
## 配置阿里巴巴Druid数据源

在我们在使用最基本的JDBC连接数据库，每一次需要查询数据库时，每次都会新建一个Connection、Statement等的对象，用完还要一个个把它们关闭掉，这样做费时费力，由于仙剑对象会占用内存的，所以一旦在同一时间大量发送请求，那内存肯定是不够的。在pom.xml文件中配置阿里druid:



## 配置Mybatis、通用Mapper及Pagehelper

在pom.xml文件中配置MyBatis、通用Mapper、PageHelper



## 预先创建整个项目的包

Config:

用于存放配置文件

Dao:

用于存放查询数据库相关的Mapper

Entity:

用于存放实体类

Exception:

用于存放自定义异常

Service:

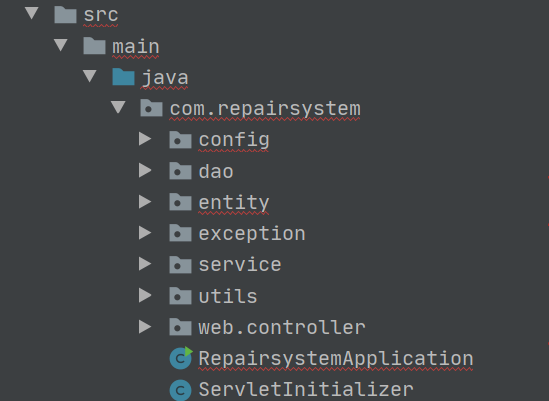
用于存放各种服务

Utils:

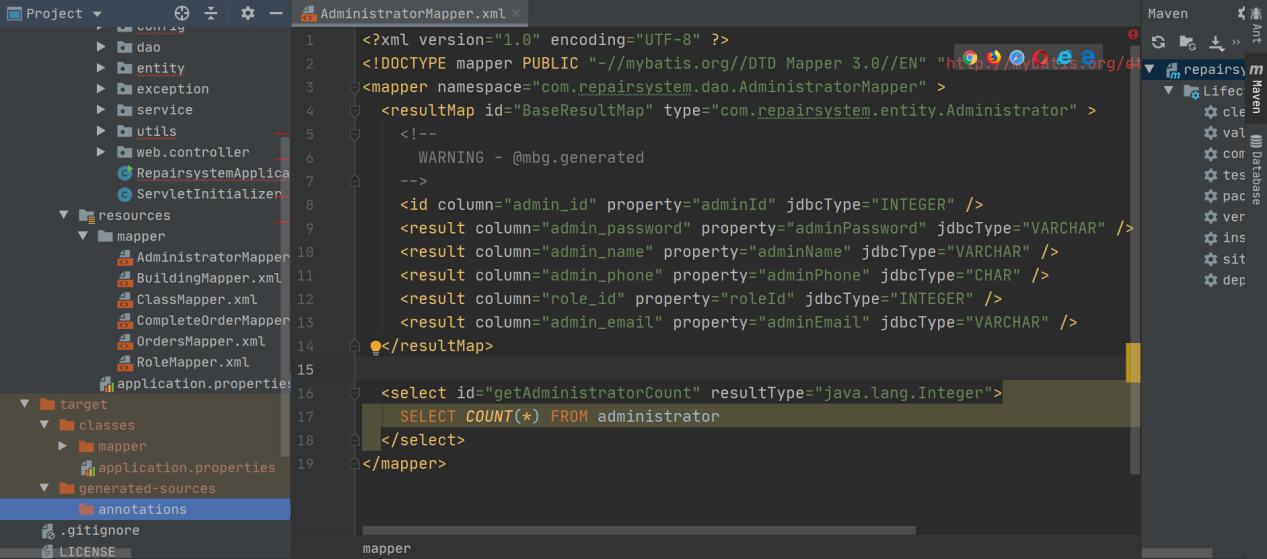
由于存放各种工具包

Web.contoller：

用于存放项目接口

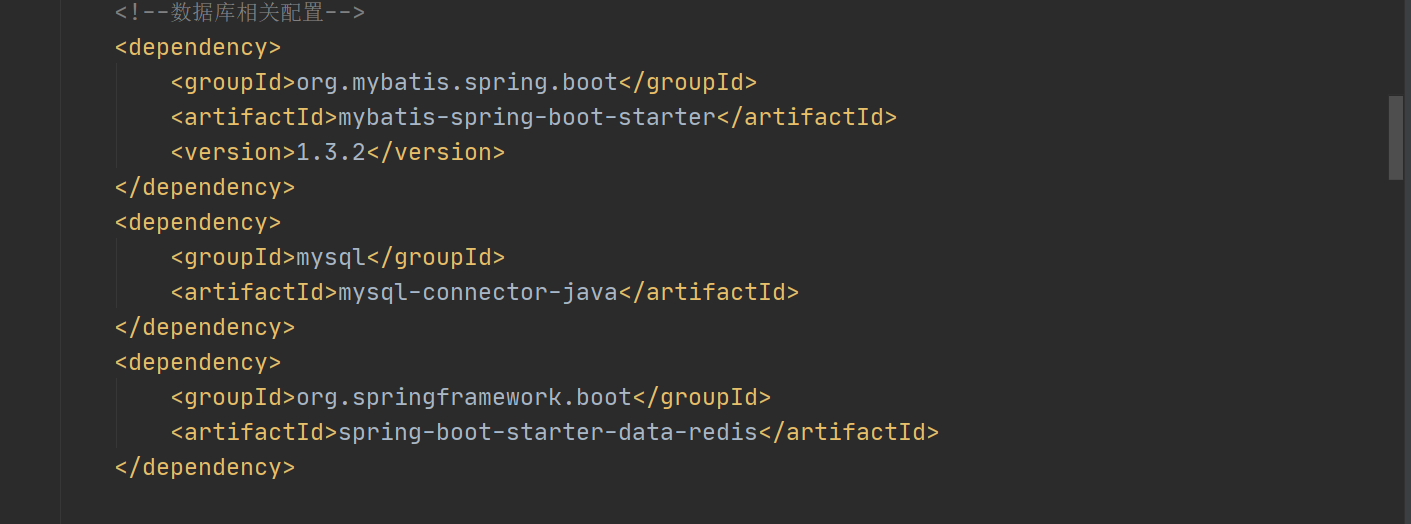


# **使用MybatisGenerator生成entity层和DAO层**

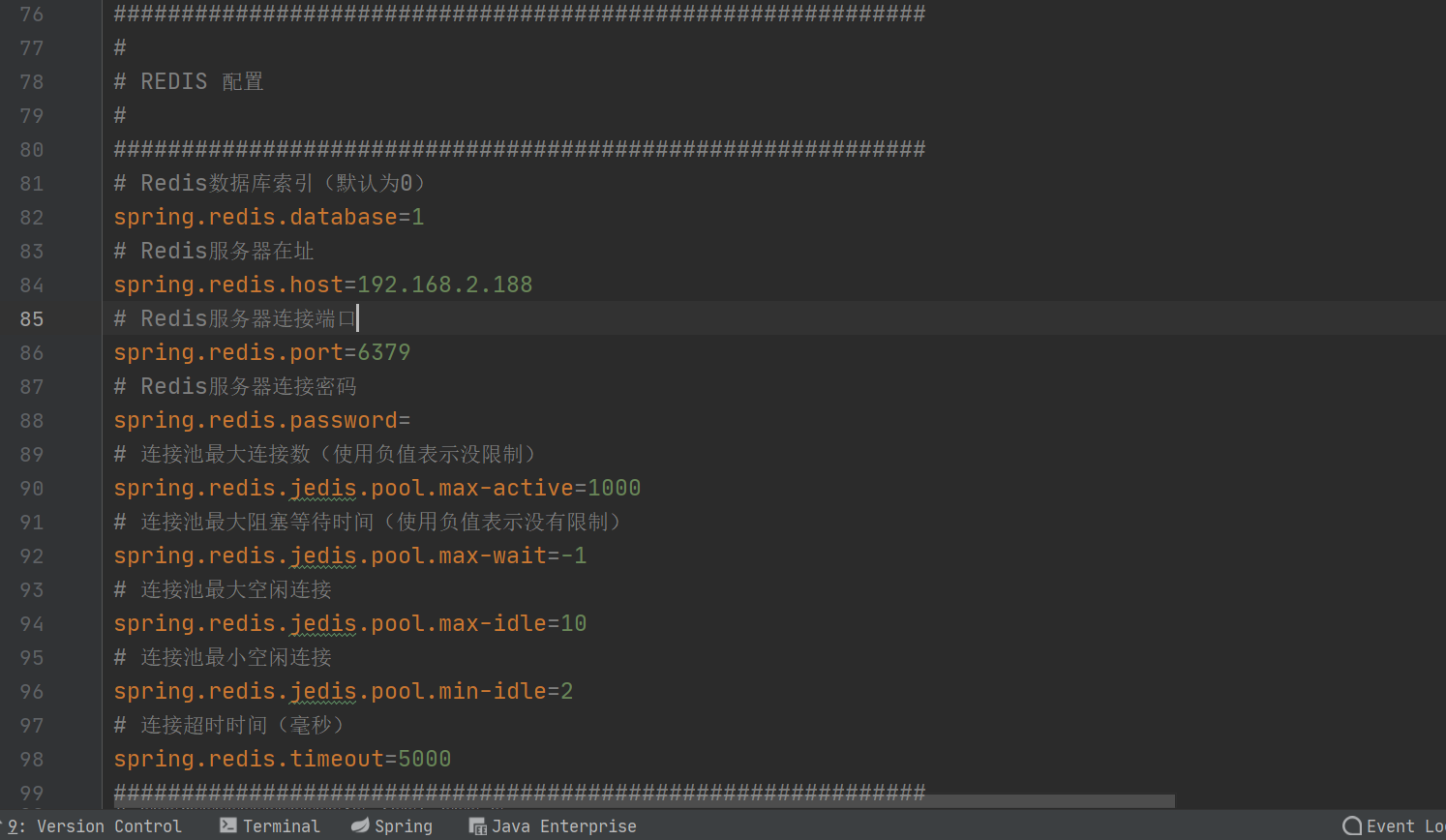


## 配置Redis非关系数据库

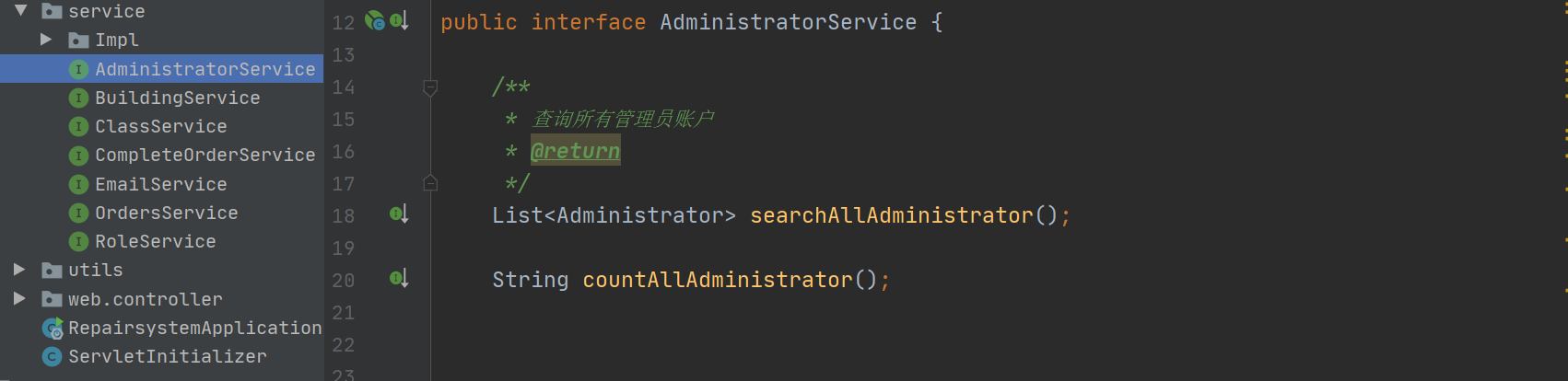
### 6.1 在Maven导入依赖



### 6.2配置Redis

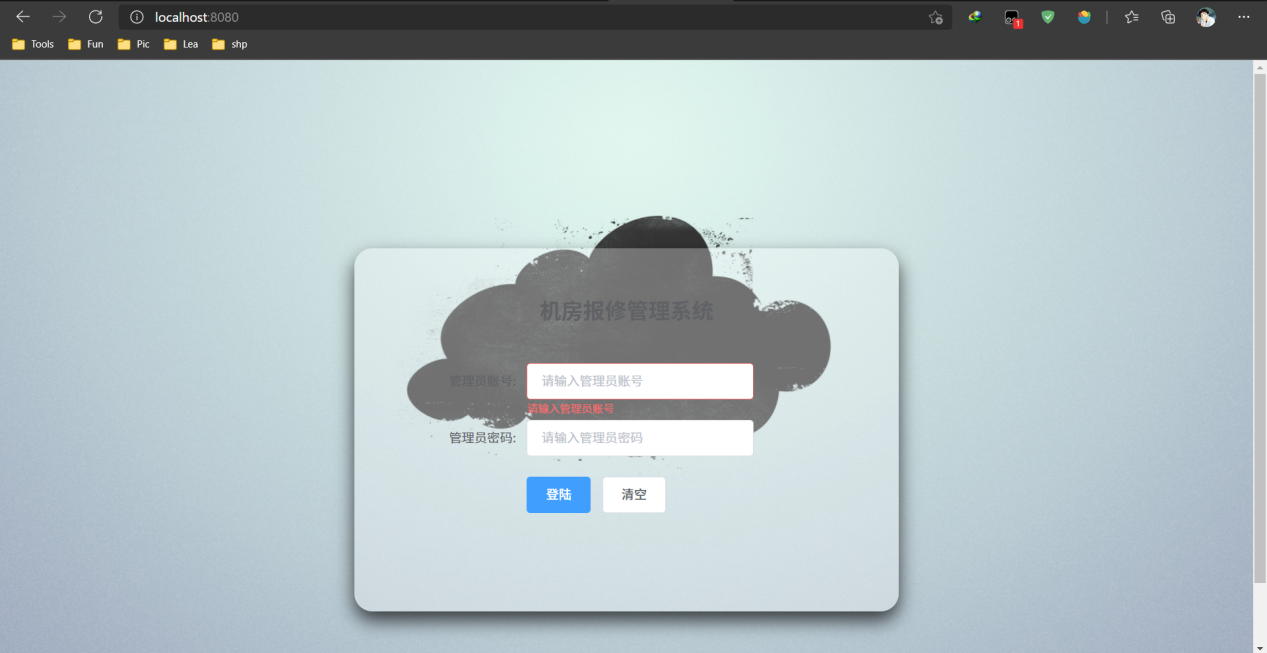


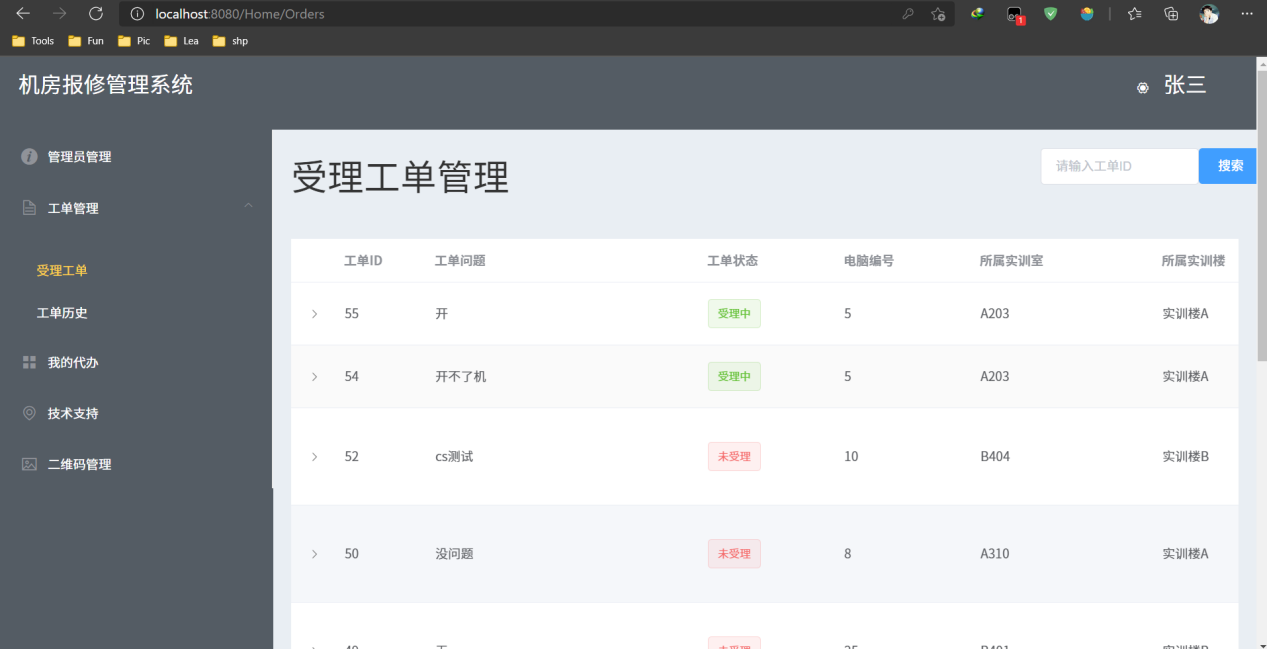
## Service层开发



分别是管理员，实验楼，教室，已完成的工单，邮件，申请工单，角色

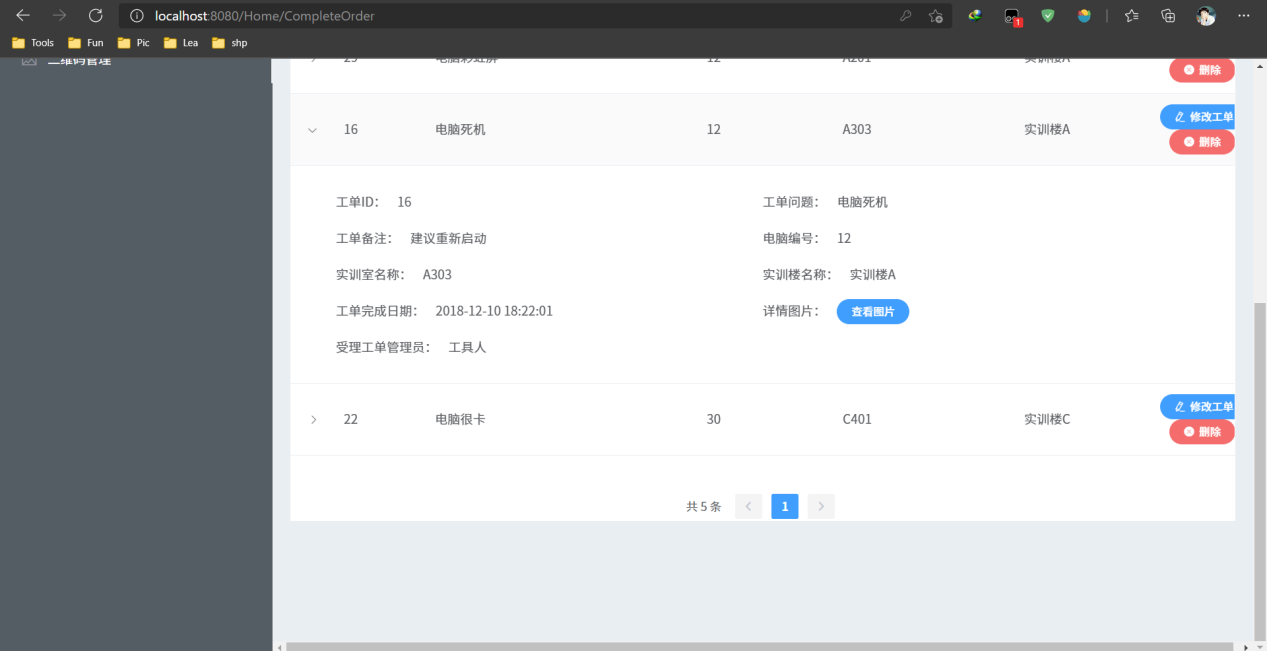
项目运行图











功能实现图

设计技术

1.后端

（1）框架

SpringBoot（含Spring、SpringMVC）

（2）数据库相关

MySQL连接驱动：mysql-connector-java

Redis操作工具：spring-boot-starter-data-redis

Mybatis：mybatis-spring-boot-starter

Mybatis通用Mapper：tk.mybatis

分页插件：com.github.pagehelper

阿里开源数据源: druid

（3）服务器

SpringBoot自带Tomcat：spring-boot-starter-tomcat

（4）其他工具类

SpringBoot测试：spring-boot-starter-test

密码加密工具：commons-codec

字符串判断工具：commons-lang3

IO流工具：commons-io

JavaBean工具：commons-beanutils

接口记录及调试工具：Swagger2

访问权限管理工具：Shiro

二维码生成器插件：qrcode-plugin

压缩文件工具：zip4j

2.项目构建工具

Maven

3.数据库

MySQL 5.7.10

Redis 4.0.2

4.项目管理工具

Git

5.本人电脑环境

操作系统：Windows10

Java版本：1.8.0\_172

IDE：JetBrains IDEA 2019.2.6