城市路灯远程监控系统软件设计与实现

**[摘要]**城市路灯远程监控系统软件，使用B/S模式，一个大型的单页面式应用。后台选择使用springboot，系统中提供监控路灯状态，记录故障信息功能，系统登录权限设置，操作权限设置，并提供维修人员使用的手机端App，维修人员可以接收到监控人员发送的维修信息，提醒，路灯信息统计。监控人员可以在PC端监控城市路灯的情况，发现故障可以向维修人员报告位置信息，维修人员进行修复。统计路灯故障率，维修记录，处理信息记录等信息，使用图表展示。

工作人员可以修改密码，工作状态，系统收集所有工作人员信息，设备信息，设备供应商信息，用户可以查询，获取所需信息。

后台设置了日志记录，按天统计，http请求auth认证。

**关键字：**React、Redux、react-router、node.js、axios、java、spring boot、mysql、Javascript、Webpack、npm、mybatis、logback、aop、material-ui、cookies

**目录**

前言.......................................................................................................................... 1

1. 绪论.............................................................................................................2

1.1 城市路灯远程监控系统软件研究背景与意义.......................................2

1.2 React介绍................................................................................................2

1.2.1 React的特点.....................................................................................2

1.2.2 React的应用情况 ........................................................................... 2

1.3 Redux........................................................................................................2

1.4 React-router .............................................................................................3

1.5 axios..........................................................................................................3

1.6 universal-cookies.......................................................................................3

1.7 material-ui ................................................................................................3

1.8 gdmap .......................................................................................................3

1.9 rechart.......................................................................................................3

1.10 springboot...................................................................................................3

1.11 mybatis.......................................................................................................3

1.12 aop..............................................................................................................4

1.13 logback.......................................................................................................4

1.14 mysql..........................................................................................................4

1. 系统分析 ..................................................................................................5

2.1 系统可行性分析.......................................................................................5

2.2 系统功能需求分析...................................................................................5

1. 系统设计....................................................................................................7

3.1 系统总体设计...........................................................................................7

3.1.1 系统目标设计..................................................................................7

3.1.2 开发设计思想..................................................................................7

3.1.3 系统功能模块设计..........................................................................7

3.2 数据库设计...............................................................................................9

3.2.1 数据库需求分析..............................................................................9

3.2.2 数据库概念结构设计......................................................................10

3.3 pc 端页面设计...........................................................................................12

3.3.1 Redux设计........................................................................................10

3.3.2 router设计........................................................................................15

3.3.3 axios使用示例..................................................................................20

3.3.4 pc端页面展示...................................................................................21

3.3.5 app页面展示....................................................................................28

3.4 后台设计...................................................................................................31

3.4.1 aop拦截web请求...........................................................................31

3.4.2 controller接口..................................................................................33

3.4.3 配置跨域..........................................................................................35

3.4.4全局异常处理...................................................................................35

3.4.5 mybatis配置......................................................................................36

3.4.6 mysql配置.........................................................................................38

3.4.7 logback配置......................................................................................38

1. App测试.....................................................................................................40

4.1 测试结果说明............................................................................................40

4.2 测试局限性分析........................................................................................40

结论.............................................................................................................................41

附录.............................................................................................................................42

附录一 主要参考资料.......................................................................................42

**前言**

鉴于路灯遍布在城市的各个角落，人工管理难度较大，建立路灯远程监控系统，对路灯损坏、维修不及时、电缆被盗等情况是十分必要的。传统的人工巡查和群众举报方式工作量大，监管效率低。相对来说，建立远程监控系统不仅可以实现全天候24h不间断的控制，还能够估计到城市的每一个角落，大大节省了人力物力成本，将被动维护转为主动抢修，提高了工作效率。

目前的城市路灯建设现状存在很多问题，建设市场混乱、建设缺乏规划、管理部门之间的沟通配合不到位，都严重影响城市路灯建设，影响城市的发展。开发一套城市路灯在线监控系统软件可以将多部门联合起来，做到发现故障及时处理，在线进行任务分配。故障路灯位置汇报，维修人员维修任务分配，及时上报维修进度。做到多部门线上联合，协同处理。这样极大的提高了工作效率。

在本选题中我使用了react来开发前台界面，spring boot来开发后台，数据库选用mysql。本文主要介绍该项目实现的一些功能和技术实现方式。

1. **绪论**

**1.1 城市路灯远程监控系统软件研究背景与意义**

城市的路灯建设质量是城市发展的情况的重要指标。随着城市的快速发展，人们对夜间照明的要求也越来越高，城市路灯的建设可以有效提高人们夜间出行的安全性，减少夜间交通安全事故的发生。在完成城市的路灯建设工作后，后续的维护管理工作也十分重要。随着计算机软件技术的发展，路灯远程监控系统可以极大的方便路灯的维护管理工作。

目前的城市路灯建设现状存在很多问题，建设市场混乱、建设缺乏规划、管理部门之间的沟通配合不到位，都严重影响城市路灯建设，影响城市的发展。开发一套城市路灯在线监控系统软件可以将多部门联合起来，做到发现故障及时处理，在线进行任务分配。故障路灯位置汇报，维修人员维修任务分配，及时上报维修进度。做到多部门线上联合，协同处理。这样可以极大的提高工作效率。

**1.2 React介绍**

**1.2.1 React的特点**

React是Facebook公司开发的一款JS库，Facebook公司给出的开发解释是他们认为MVC无法满足他们的拓展请求，由于他们代码库和组织非常巨大，随着功能或特性的增加，系统的复杂度就呈指数级增长，因此之前的代码就变得异常脆弱，同时也难以理解和调试，特别是模型和试图之间可能存在的双向数据流动。未解决这个难题，他们设计了React—一个Javascript框架，用于构建“可预期的”和“声明式的”Web用户界面，它已经使Facebook更快地开发Web应用。

React特点归结如下：

* + - 1. 声明式设计： React采用声明范式，可以轻松描述应用。
      2. 组件式设计： 可以将ui封装为一个组件，可以多次使用。
      3. 灵活： 独特的JSX格式，js和html可以混合编写。

**1.2.2 React的应用情况**

React在国外的应用较多，facebook、Yahoo、Reddit等，国内像阿里巴巴。React的应用范围是非常广泛的，而且随着技术的成熟，使用范围将会变得更加广泛。

**1.3 Redux**

Redux是javascript状态容器，提供可预测化的状态管理。存储原理：应用中所用的state都以一个对象树的形式存储在一个单一的store中，唯一改变state的办法是触发action，一个描述发生什么的对象。为了描述action如何改变state树，需要编写reducers。在本项目中我把一些在多个页面使用的数据放在了store中，方便各页面的使用。

**1.4 React-router**

React-router是一个针对React而设计的路由解决方案、可以友好的帮你解决React components到URL之间的同步映射关系。在本App中引入了React-router4。

通过hash值的变化来切换页面，项目中引入了history用来控制页面的跳转。

**1.5 axios**

axios是一个基于promise的HTTP库，可以用在浏览器和nod.js中，使用方便。

**1.6 universal-cookies**

Universal-cookies提供了set和get方法来设置或获取cookie，在本项目中用来实现记住密码的功能。

cookies.set("roadlogname", this.state.logname, {maxAge: 30\*24\*60\*60})

cookies.set("roadpwd", this.state.pwd, {maxAge: 30\*24\*60\*60})

**1.7 Material-ui**

Material-ui是一个使用react编写的ui库，界面美观，本项目使用了material-ui的各种组件。

**1.8 gdmap**

高德地图是国内优秀的地图提供商，通过注册为开发者可以使用高德地图的api服务，本项目中使用高德地图来记录路灯位置信息。

**1.9 rechart**

Rechart是基于echart封装的react版本，是国内百度开发的图标记录产品。本项目使用rechart来显示设备信息。

**1.10 spring boot**

Spring Boot是Spring社区较新的一个项目。该项目的目的是帮助开发者更容易的创建基于Spring的应用程序和服务，让更多人的人更快的对Spring进行入门体验，让Java开发也能够实现Ruby on Rails那样的生产效率。为Spring生态系统提供了一种固定的、约定优于配置风格的框架。

Spring Boot具有如下特性：

1.为基于Spring的开发提供更快的入门体验

2.开箱即用，没有代码生成，也无需XML配置。同时也可以修改默认值来满足特定的需求。

3.提供了一些大型项目中常见的非功能性特性，如嵌入式服务器、安全、指标，健康检测、外部配置等。

4.Spring Boot并不是不对Spring功能上的增强，而是提供了一种快速使用Spring的方式。

本项目使用了springboot进行后台的开发。

**1.11 mybatis**

 MyBatis 是一个可以自定义SQL、存储过程和高级映射的[持久层](https://baike.sogou.com/v55187093.htm" \t "https://baike.sogou.com/_blank)框架。MyBatis 摒除了大部分的JDBC代码、手工设置参数和[结果集](https://baike.sogou.com/v66020459.htm" \t "https://baike.sogou.com/_blank)重获。MyBatis 只使用简单的XML 和注解来配置和映射基本数据类型、Map 接口和POJO 到数据库记录。

MyBatis 是支持定制化 SQL、存储过程以及高级映射的优秀的持久层框架。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以对配置和原生Map使用简单的 XML 或注解，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。

本项目使用mybatis进行数据库操作的预处理。优化数据操作。

**1.12 aop**

AOP为Aspect Oriented Programming的缩写，意为：[面向切面编程](https://baike.sogou.com/v53402515.htm" \t "https://baike.sogou.com/_blank)（也叫面向方面），可以通过[预编译](https://baike.sogou.com/v8400613.htm" \t "https://baike.sogou.com/_blank)方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。AOP实际是GoF设计模式的延续，设计模式孜孜不倦追求的是调用者和被调用者之间的解耦，AOP可以说也是这种目标的一种实现。

本项目中队请求的header查询就是通过aop实现。

**1.13 logback**

Logback是spring boot内置的日志记录插件，可以讲应用的运行状况进行记录，并支持输出到文件或数据库。本项目中通过配置logback-spring.xml文件将日志记录到log文件夹，按天保存。

**1.14 mysql**

MySQL是一个[关系型数据库管理系统](https://baike.sogou.com/v400453.htm" \t "https://baike.sogou.com/_blank)，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 最流行的关系型数据库管理系统，在 WEB 应用方面MySQL是最好的 RDBMS (Relational Database Management System，[关系数据库管理系统](https://baike.sogou.com/v400453.htm" \t "https://baike.sogou.com/_blank)) 应用软件之一。

MySQL是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个[大仓库](https://baike.sogou.com/v816184.htm" \t "https://baike.sogou.com/_blank)内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、[总体拥有成本](https://baike.sogou.com/v60488729.htm" \t "https://baike.sogou.com/_blank)低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为[网站数据库](https://baike.sogou.com/v7927370.htm" \t "https://baike.sogou.com/_blank)。

本项目使用mysql进行数据的存储。

1. **系统分析**

**2.1 系统可行性分析**

可行性研究的目的就是用最小的代价在尽可能短的时间内确定问题是否能够解决，确定问题是否值得去解决。通过客观分析，从而判断系统目标和规模是否现实，系统完成后所能带来的收益是否大到值得开发这个系统的程度。因此，可行性研究实质上就是要进行一次已经压缩简化了的系统分析和设计的过程，也就是在较高层次上以较为抽象的方式进行的系统分析和设计的过程。从以上原则来看，使用新兴技术重新开发城市路灯远程监控系统，将多部门联合起来，做到发现故障及时处理，在线进行任务分配。故障路灯位置汇报，维修人员维修任务分配，及时上报维修进度。做到多部门线上联合，协同处理，提高路灯维护工作的工作效率。同时本设计可行性概括如下：

1. 技术上可行。本人对js有一定基础，有React开发经验，可以完成web页面的编写，结合Redux和React-router实现数据存储和页面跳转。对springboot框架有使用经历，可以完成后台代码编写，通过axios连接前后端，处理请求，Mysql存储用户信息。
2. 经济可行性。React、Redux、React-router、springboot等需要使用的工具都是开源的，无需支付任何费用，只需支付数据库方面和人工费用，成本比较低，在软件完成后如果进一步改进后投入市场，可以方便用户管理路灯维护工作，操作简单。用户无需进行复杂的操作便可完成处理过程。无需进行专业的使用教程便可自由的操作本软件。

所以从以上几个方面的分析看，开发本软件是完全可行的。

**2.2 系统功能需求分析**

**城市路灯远程监控系统软件实现的功能：**

1. 用户登录
2. 记住密码
3. 查看设备实时监控
4. 查看设备信息，位置
5. 处理故障设备
6. 选择在职维修人员分配维修任务
7. 查看维修人员信息，查询
8. 查看处理记录，查询
9. 查看维修记录，提醒维修人员尽快进行维修
10. 查看设备供应商信息
11. 记住密码
12. 修改工作状态
13. 退出登录
14. 维修人员接受任务和提醒

**研究内容：**

1. React复用组件的封装过程以及使用方法
2. Redux store使用，状态管理和数据更新过程
3. React-router路由跳转设置，动态加载原理
4. Springboot框架的使用，mybatis配置
5. 日志记录配置
6. Aop实现请求拦截处理方法
7. **系统设计**

**3.1 系统总体分析**

**3.1.1 系统目标设计**

经过系统设计与实验，最终产生城市路灯远程监控系统软件设计与实现。

**3.1.2 开发设计思想**

使用react开发城市路灯远程监控系统软件，使用B/S模式，开发一个大型的单页面式应用。后台选择使用springboot，由于硬件方面难以调试，所以我考虑在软件中操作是发送模拟指令，不涉及到真实路灯控制，系统中提供监控路灯状态，记录故障信息功能，系统登录权限设置，操作权限设置，并提供维修人员使用的手机端App，维修人员可以接收到监控人员发送的维修信息，提醒。监控人员可以在PC端监控城市路灯的情况，发现故障可以向维修人员报告位置信息，维修人员进行修复。统计路灯故障率等信息，使用图表展示。

为了满足不同用户的需求，界面尽量简单，使用且方便，对用户友好。

**3.1.3 系统功能模块设计**

页面关系及预计实现功能分布

1. **pc端进入系统**

修改状态

修改密码

登录

设备供应商信息

维修人员信息

管理人员信息

处理信息记录

维修任务记录

故障路灯处理

路灯监控信息

实时监控信息

信息实时反应那个

信息记录

查询

提醒维修

维修记录

分配维修

故障记录

位置信息

故障数量

维修数量

1. **App端进入系统**

修改状态

修改密码

登录

任务

设置

用户

设备

提醒

任务

**3.2 数据库设计**

**3.2.1 数据库需求分析**

根据要实现的功能要求需要

1. 管理人员登录表——保存用户的账号、密码、员工代码
2. 路灯信息表——路灯编号、位置、经纬度信息、生产厂商代码、设备状态、故障时间、维修职员代码
3. 7日故障数量统计
4. 人员信息表——代码、姓名、联系方式、状态
5. 设备供应商信息表——代码、名称、地址、联系方式
6. 处理记录表——路灯信息、处理人员信息、处理时间
7. 维修人员登录表——账号、密码、员工代码
8. 维修任务表——路灯信息

**3.2.2 数据库概念结构设计**

得到上面的数据项和数据结构以后，就可以设计出能够满足用户需求的各种实体结构，以及他们之间的关系，为以后的逻辑结构打下基础。

1. 管理人员登录表



1. 路灯信息表



1. 7日故障数量统计



1. 人员信息表



1. 设备供应商信息表



1. 处理记录表



1. 维修人员登录表



1. 维修任务表



**3.3 pc端页面设计**

**创建项目:**

npx install -g create-react-app manage

cd manage

npm install redux

npm install react-router

npm install react-dom

npm install material-ui

npm install universal-cookie

npm install antd

npm install axios

npm install history

现在已经导入了项目需要的依赖，npm start可以运行项目

**3.3.1 Redux设计**

当用户登录时需要保存用户信息，退出登录事件处理，故障详情页临时数据，维修详情页临时数据。

在src目录下新建actions文件夹，其中新建index.js文件

具体实现如下：

export const SET\_LOGIN = 'SET\_LOGIN';

export const REMOVE\_LOGIN = 'REMOVE\_LOGIN';

export const SET\_ERROR = 'SET\_ERROR';

export const REMOVE\_ERROR = 'REMOVE\_ERROR';

export const SET\_FIX = 'SET\_FIX';

export const REMOVE\_FIX = 'REMOVE\_FIX';

export function login(login) {

return {

type: SET\_LOGIN,

login

}

};

export function loginout() {

return {

type: REMOVE\_LOGIN

}

}

export function error(error) {

return {

type: SET\_ERROR,

error

}

};

export function removeerror() {

return {

type: REMOVE\_ERROR

}

}

export function fix(fix) {

return {

type: SET\_FIX,

fix

}

};

export function removefix() {

return {

type: REMOVE\_FIX

}

}这些函数分别可以处理上述需求功能，接下来设置reducers来对actions中的函数进行处理。

在src目录下新建reducers文件夹，其中新建index.js文件

具体实现：

import { combineReducers } from 'redux';

import { SET\_LOGIN, REMOVE\_LOGIN, SET\_ERROR, REMOVE\_ERROR, SET\_FIX, REMOVE\_FIX } from '../actions';

function login(state = {}, action) {

switch(action.type) {

case SET\_LOGIN:

return action.login;

case REMOVE\_LOGIN:

return {};

default:

return state;

}

}

function error(state = {}, action) {

switch(action.type) {

case SET\_ERROR:

return action.error;

case REMOVE\_ERROR:

return {};

default:

return state;

}

}

function fix(state = {}, action) {

switch(action.type) {

case SET\_FIX:

return action.fix;

case REMOVE\_FIX:

return {};

default:

return state;

}

}

const road = combineReducers({

login,

error,

fix

});

export default road;

这样在其他页面就可以使用store中的数据了，也可以进行更改。

使用数据：

function select(state) {

return {

login: state.login

}

}

export default connect(select)(user)

通过select函数可以对数据进行筛选，connect将数据绑定到页面中

更改数据：

this.props.dispatch(函数名(payload))

更改括号内的函数名和参数即可分发事件来更改store中的值。

index.js加入store

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import { createStore } from 'redux';

import { Provider } from 'react-redux';

import registerServiceWorker from './registerServiceWorker';

import road from './reducers';

import R from './router';

let store = createStore(road);

ReactDOM.render(

<Provider store={store}>

<R />

</Provider>, document.getElementById('root'));

registerServiceWorker();

**3.3.2 router设计**

history/index.js

/\*导入history包，路由跳转时使用\*/

import createHistory from 'history/createBrowserHistory'

export default createHistory()

Router/index.js

import React from 'react';

import {

Route,

Router,

Switch

} from 'react-router';

import axios from 'axios';

import history from '../history';

import M from '../page/menu/menu';

import L from '../page/login/login';

axios.defaults.headers.common['auth'] = 'road2018';

axios.defaults.timeout = 3000;

const router = () => (

<Router history={history}>

<Switch>

<Route exact path="/" component={L} />

<Route exact path="/control" component={M} />

<Route exact path="/chart" component={M} />

<Route exact path="/errortable" component={M} />

<Route exact path="/fixtable" component={M} />

<Route exact path="/record" component={M} />

<Route exact path="/managep" component={M} />

<Route exact path="/maintainp" component={M} />

<Route exact path="/proserver" component={M} />

<Route exact path="/errormsg" component={M} />

<Route exact path="/fixmsg" component={M} />

<Route exact path="/user" component={M} />

<Route exact path="/changepwd" component={M} />

</Switch>

</Router>

);

export default router;

上面还配置了axios的通用设置，使用上面的配置，可以实现页面的跳转，通过

history.push()或history.goBack()来进行页面间的跳转。

通用菜单栏设置

import React, { Component } from 'react';

import { connect } from 'react-redux'

import {

Route,

Router,

Switch

} from 'react-router';

import history from '../../history';

import MuiThemeProvider from 'material-ui/styles/MuiThemeProvider';

import AppBar from 'material-ui/AppBar';

import Drawer from 'material-ui/Drawer';

import MenuItem from 'material-ui/MenuItem';

import RemoveRedEye from 'material-ui/svg-icons/image/remove-red-eye';

import IconButton from 'material-ui/IconButton';

import IconMenu from 'material-ui/IconMenu';

import MoreVertIcon from 'material-ui/svg-icons/navigation/more-vert';

import PersonAdd from 'material-ui/svg-icons/action/account-circle';

import ContentLink from 'material-ui/svg-icons/action/settings';

import Divider from 'material-ui/Divider';

import ContentCopy from 'material-ui/svg-icons/action/help';

import Delete from 'material-ui/svg-icons/action/exit-to-app';

import Airplay from 'material-ui/svg-icons/av/airplay';

import Chart from 'material-ui/svg-icons/editor/insert-chart';

import Error from 'material-ui/svg-icons/alert/error';

import Build from 'material-ui/svg-icons/action/build';

import Content from 'material-ui/svg-icons/content/content-paste';

import Work from 'material-ui/svg-icons/action/work';

import Perm from 'material-ui/svg-icons/action/perm-identity';

import Account from 'material-ui/svg-icons/action/supervisor-account';

import Control from '../control/control';

import Map from '../map/map';

import ETable from '../maintain/errortable';

import RTable from '../record/record';

import MP from '../managep/managep';

import MAP from '../maintainp/maintainp';

import PS from '../proserver/proserver';

import ERRM from '../maintain/errmessage';

import FT from '../fix/fixtable';

import FIXM from '../fix/fixmessage';

import U from '../user/user';

import CP from '../user/changepwd';

import { loginout, removeerror, removefix } from '../../actions';

class M extends Component {

state = {

open: false

}

componentDidMount() {

if(this.props.login.logname === undefined) {

history.push('/');

}

}

handleToggle = () => this.setState({open: !this.state.open});

handleClose = () => this.setState({open: false});

loginout = () => {

this.props.dispatch(removeerror());

this.props.dispatch(removefix());

this.props.dispatch(loginout());

history.push('/');

}

render() {

const Logged = (props) => (

<IconMenu

{...props}

iconButtonElement={

<IconButton><MoreVertIcon /></IconButton>

}

targetOrigin={{horizontal: 'right', vertical: 'top'}}

anchorOrigin={{horizontal: 'right', vertical: 'top'}}

>

<MenuItem primaryText={"用户" + this.props.login.logname} leftIcon={<PersonAdd />} />

<MenuItem primaryText="详细信息" leftIcon={<RemoveRedEye />} onClick={() => history.push('/user')}/>

<MenuItem primaryText="修改密码" leftIcon={<ContentLink />} onClick={() => history.push('/changepwd')}/>

<Divider />

<MenuItem primaryText="操作帮助" leftIcon={<ContentCopy />} />

<Divider />

<MenuItem primaryText="退出登录" leftIcon={<Delete />} onClick={this.loginout}/>

</IconMenu>

);

Logged.muiName = 'IconMenu';

return (

<Router history={history}>

<MuiThemeProvider>

<div>

<AppBar

title="远程路灯监控管理系统"

style={{width: '100%'}}

onLeftIconButtonClick={() => this.setState({open: true})}

iconElementRight={<Logged />}

/>

<Drawer

docked={false}

width={200}

open={this.state.open}

onRequestChange={(open) => this.setState({open})}

>

<MenuItem onClick={() => {history.push('/control');this.setState({open:false})}} leftIcon={<Airplay />}>实时监控信息</MenuItem>

<MenuItem onClick={() => {history.push('/chart');this.setState({open:false})}} leftIcon={<Chart />}>路灯监控信息</MenuItem>

<MenuItem onClick={() => {history.push('/errortable');this.setState({open:false})}} leftIcon={<Error />}>故障路灯处理</MenuItem>

<MenuItem onClick={() => {history.push('/fixtable');this.setState({open:false})}} leftIcon={<Build />}>维修任务记录</MenuItem>

<MenuItem onClick={() => {history.push('/record');this.setState({open:false})}} leftIcon={<Content />}>处理信息记录</MenuItem>

<MenuItem onClick={() => {history.push('/managep');this.setState({open:false})}} leftIcon={<Work />}>管理人员信息</MenuItem>

<MenuItem onClick={() => {history.push('/maintainp');this.setState({open:false})}} leftIcon={<Perm />}>维修人员信息</MenuItem>

<MenuItem onClick={() => {history.push('/proserver');this.setState({open:false})}} leftIcon={<Account />}>设备供应商信息</MenuItem>

</Drawer>

<Switch>

<Route exact path="/control" component={Control} />

<Route exact path="/chart" component={Map} />

<Route exact path="/errortable" component={ETable} />

<Route exact path="/fixtable" component={FT} />

<Route exact path="/record" component={RTable} />

<Route exact path="/managep" component={MP} />

<Route exact path="/maintainp" component={MAP} />

<Route exact path="/proserver" component={PS} />

<Route exact path="/errormsg" component={ERRM} />

<Route exact path="/fixmsg" component={FIXM} />

<Route exact path="/user" component={U} />

<Route exact path="/changepwd" component={CP} />

</Switch>

</div>

</MuiThemeProvider>

</Router>

);

}

}

function select(state) {

return {

login: state.login

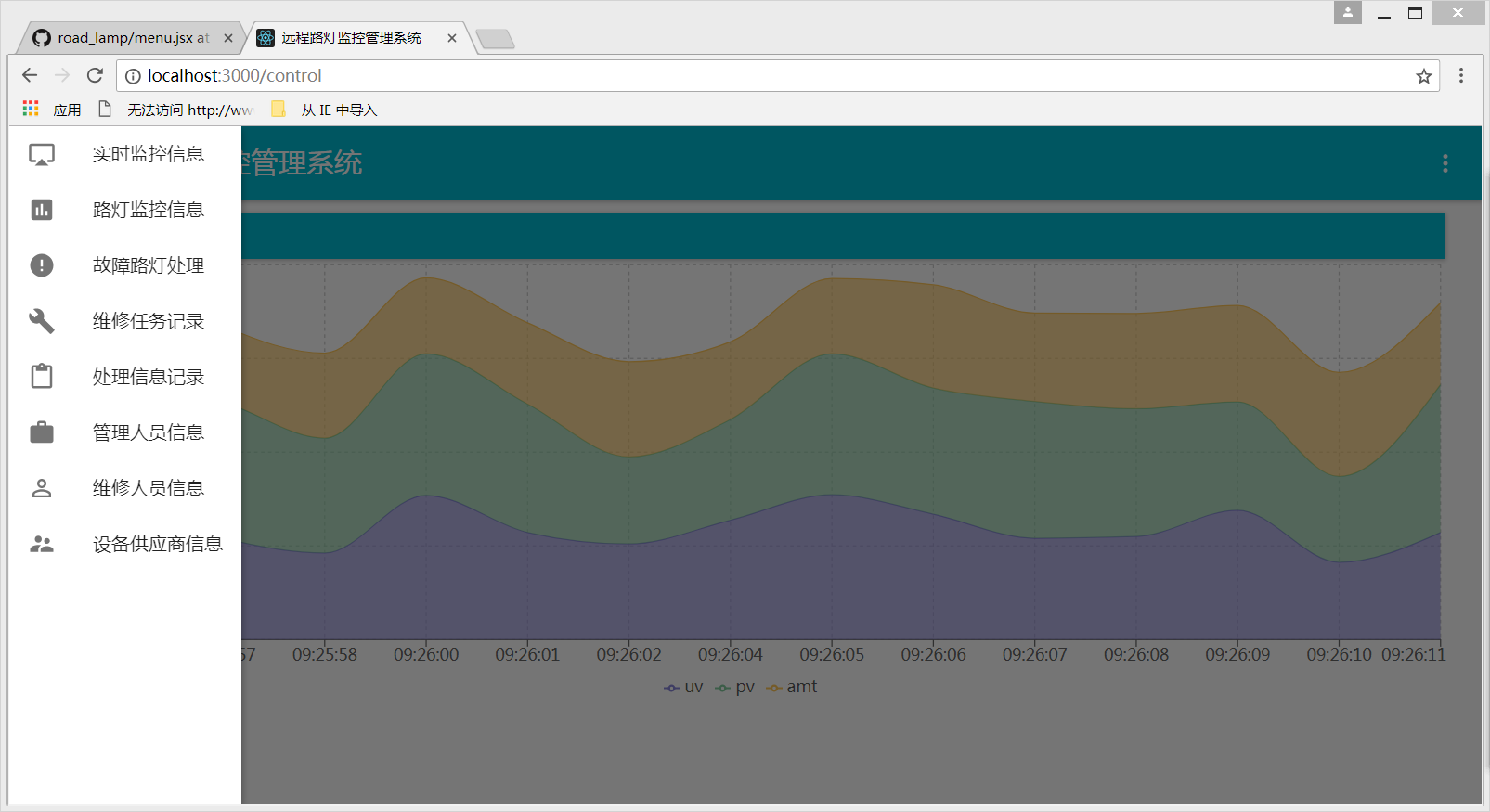
}

}

export default connect(select)(M);

这样菜单栏可以控制页面的跳转，而且也是一个公用组件。

实现效果：



**3.3.3 axios使用示例**

import axios from 'axios';

import { message } from 'antd';

axios.post('http://localhost:1111/data')

.then((response) => {

if(response.data.success) {

let da = response.data.countrecord;

let d = [];

for(let i = 0; i < da.length; i++) {

let temp = {};

temp.name = da[i].time;

temp.故障 = da[i].error;

temp.维修 = da[i].fix;

d.unshift(temp);

}

let today = {};

today.name = moment().format('YYYY-MM-DD').toString();

today.故障 = response.data.errorcount;

today.维修 = response.data.fixcount;

d.push(today);

this.setState({

data: d

})

} else {

message.error('获取数据失败！');

}

})

.catch(function (error) {

console.log(error);

message.error('网络连接失败！');

});

这样可以请求到后台，并拿到需要数据。

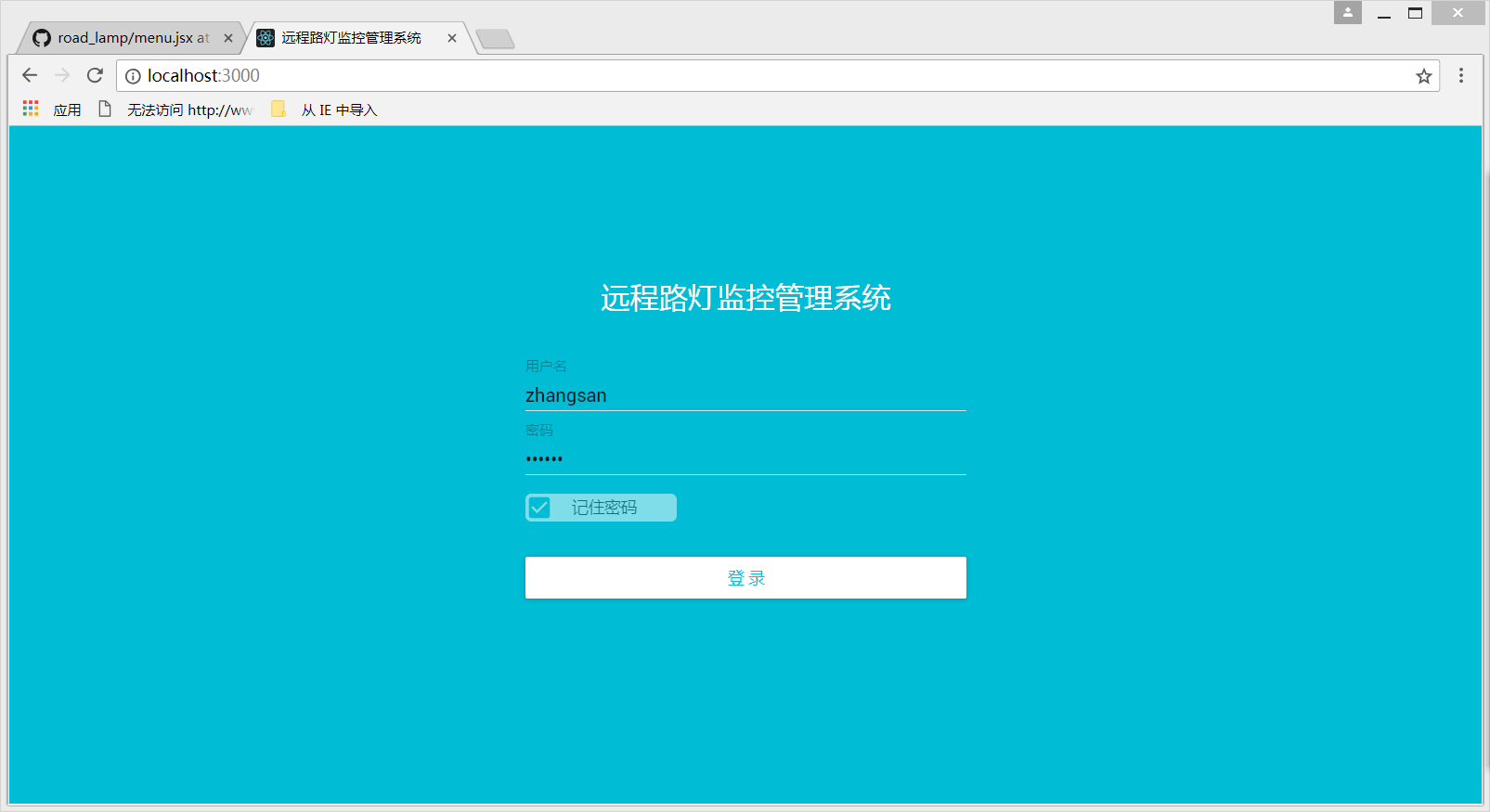
axios.defaults.headers.common['auth'] = 'road2018';

axios.defaults.timeout = 3000;

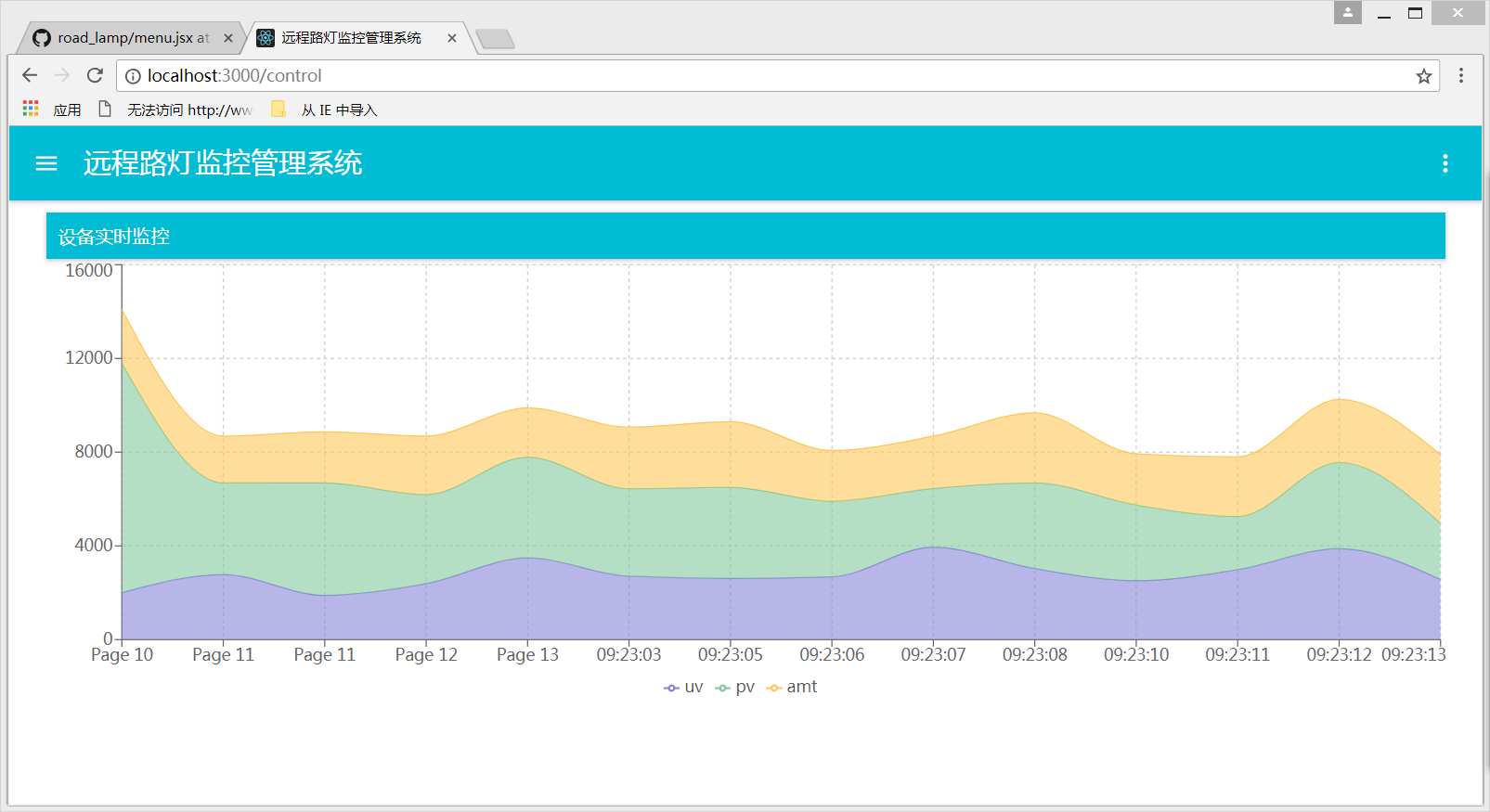
设置请求头信息和默认超时时间。

**3.3.4 pc端页面展示**

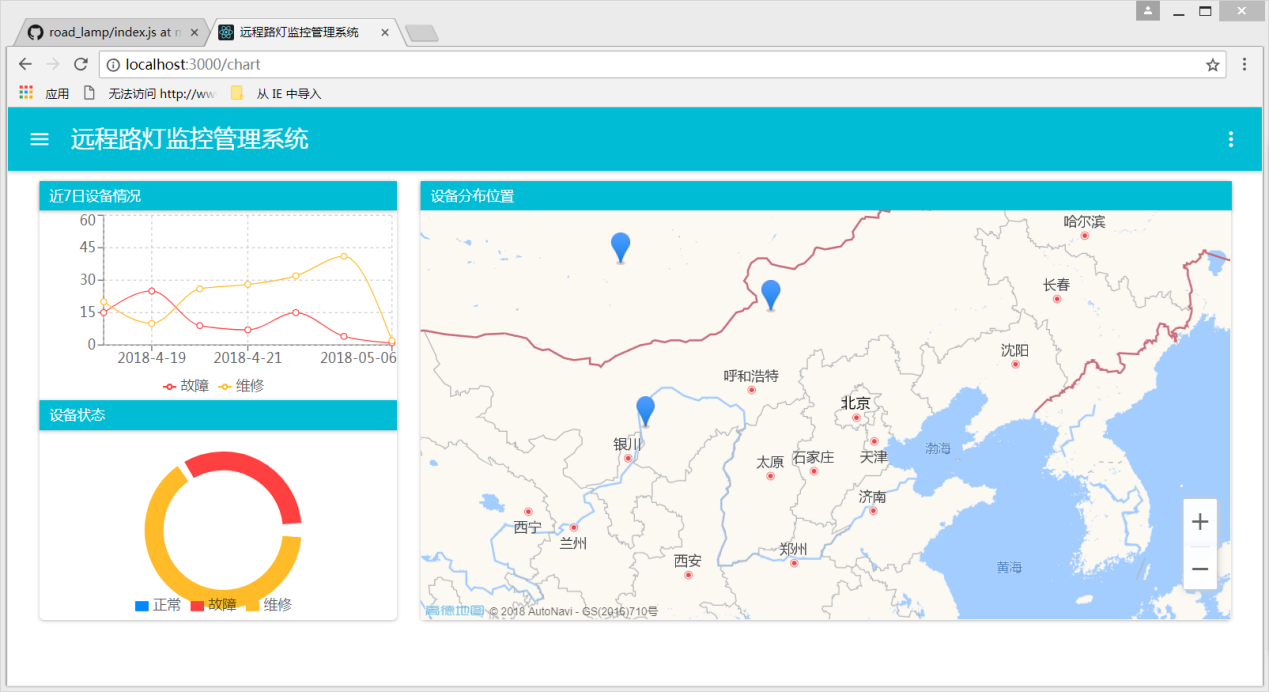
1.登录页面



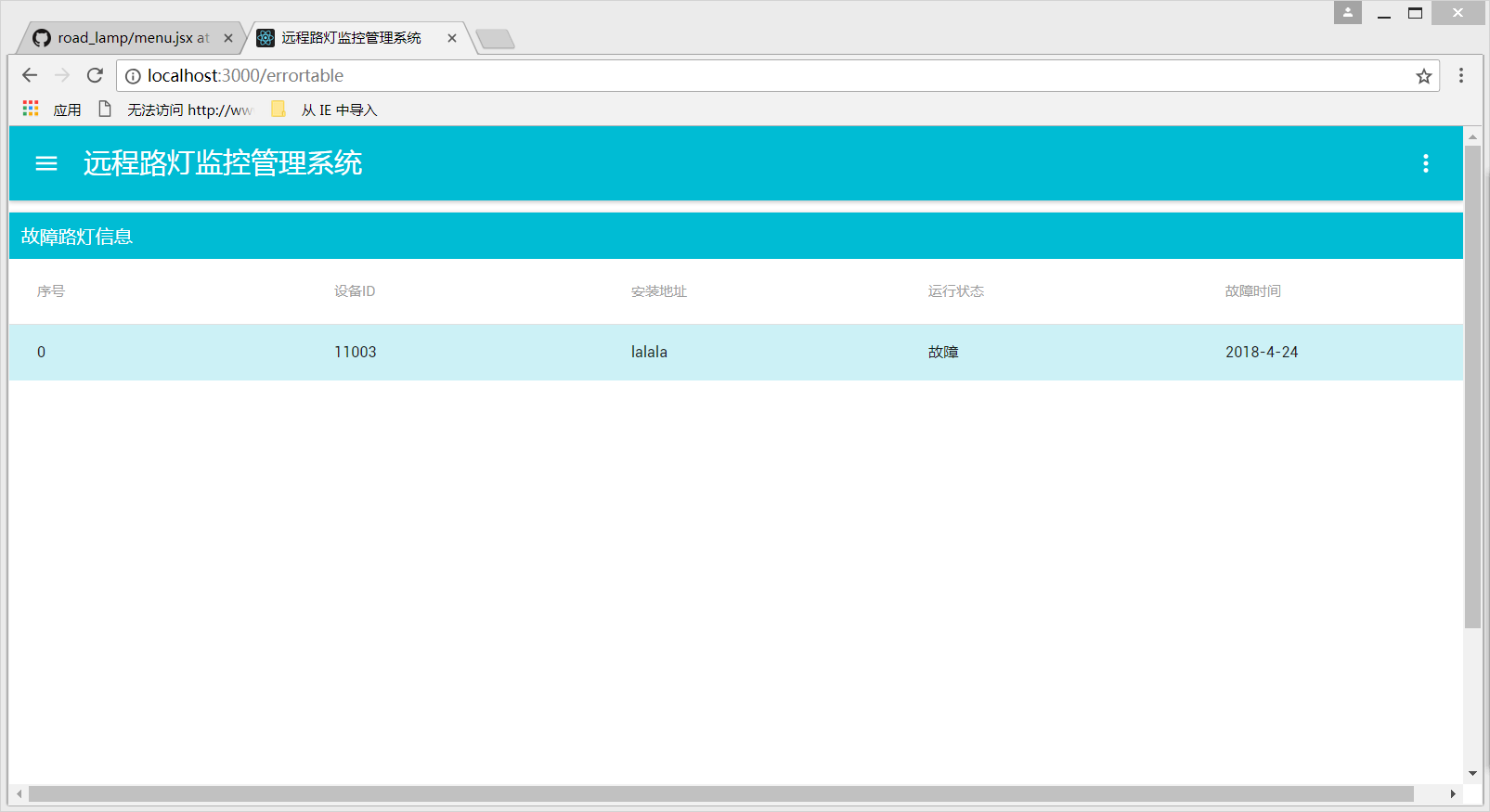
2.实时设备信息页面

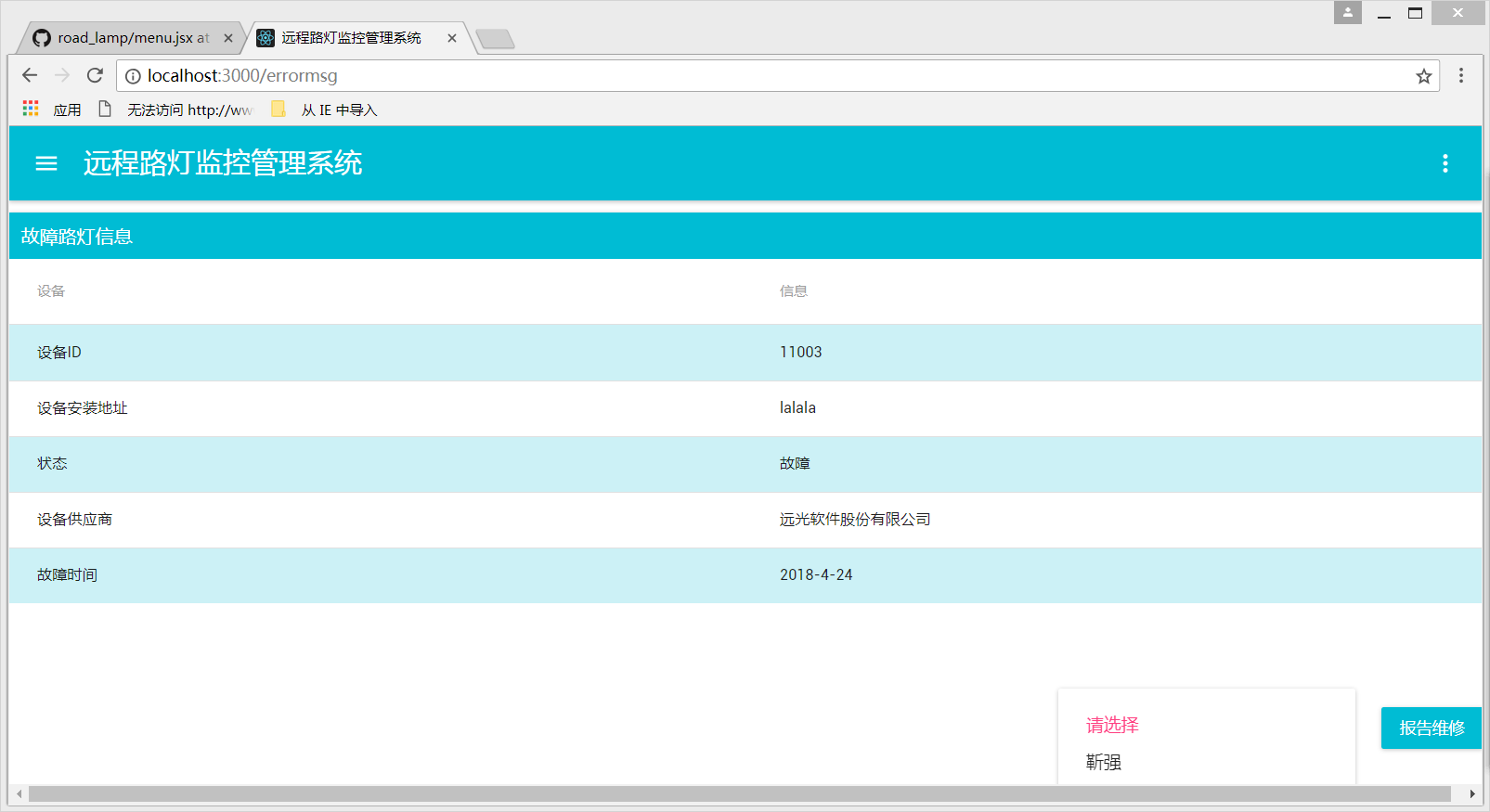


3.设备统计信息页面

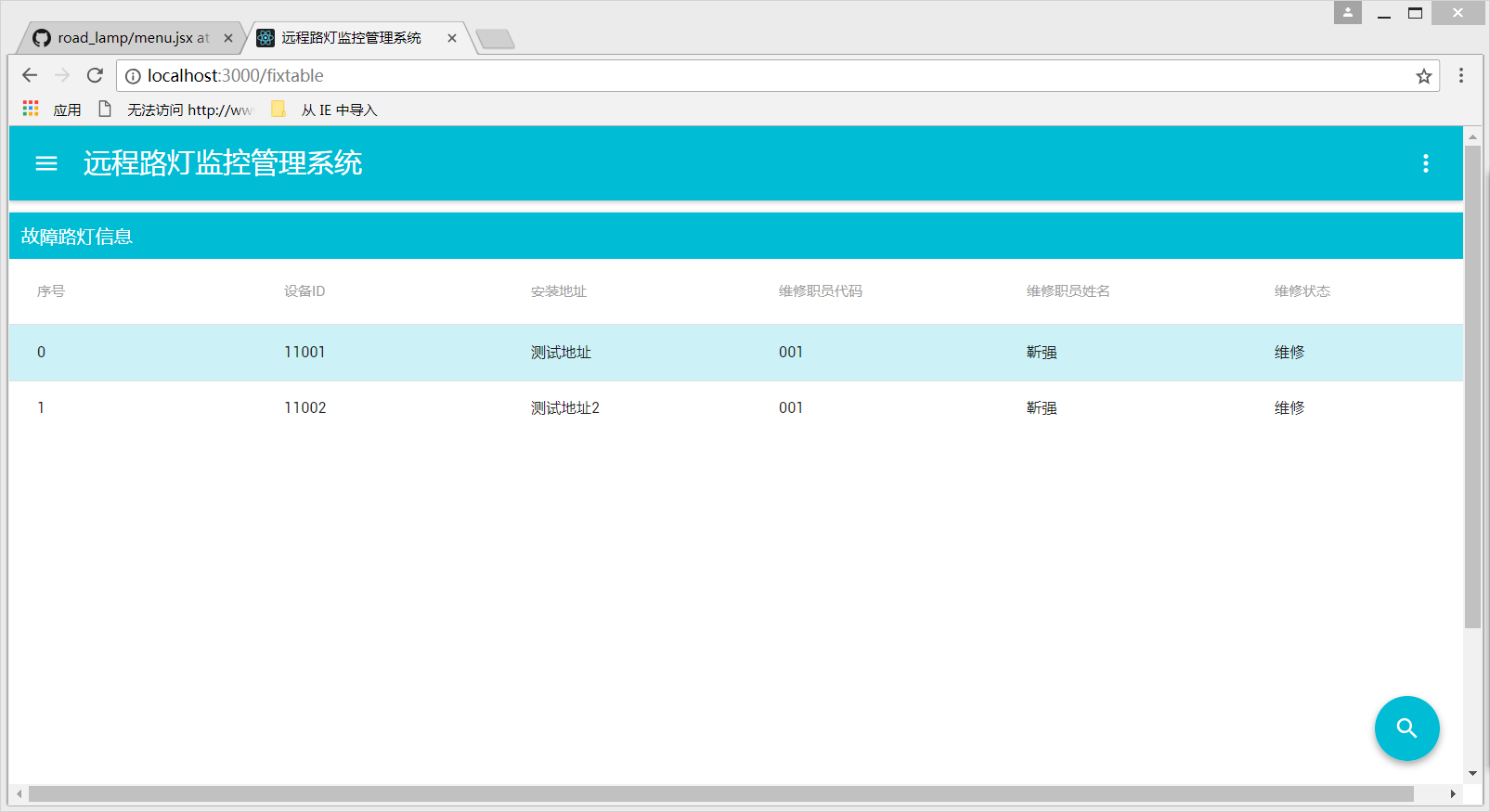


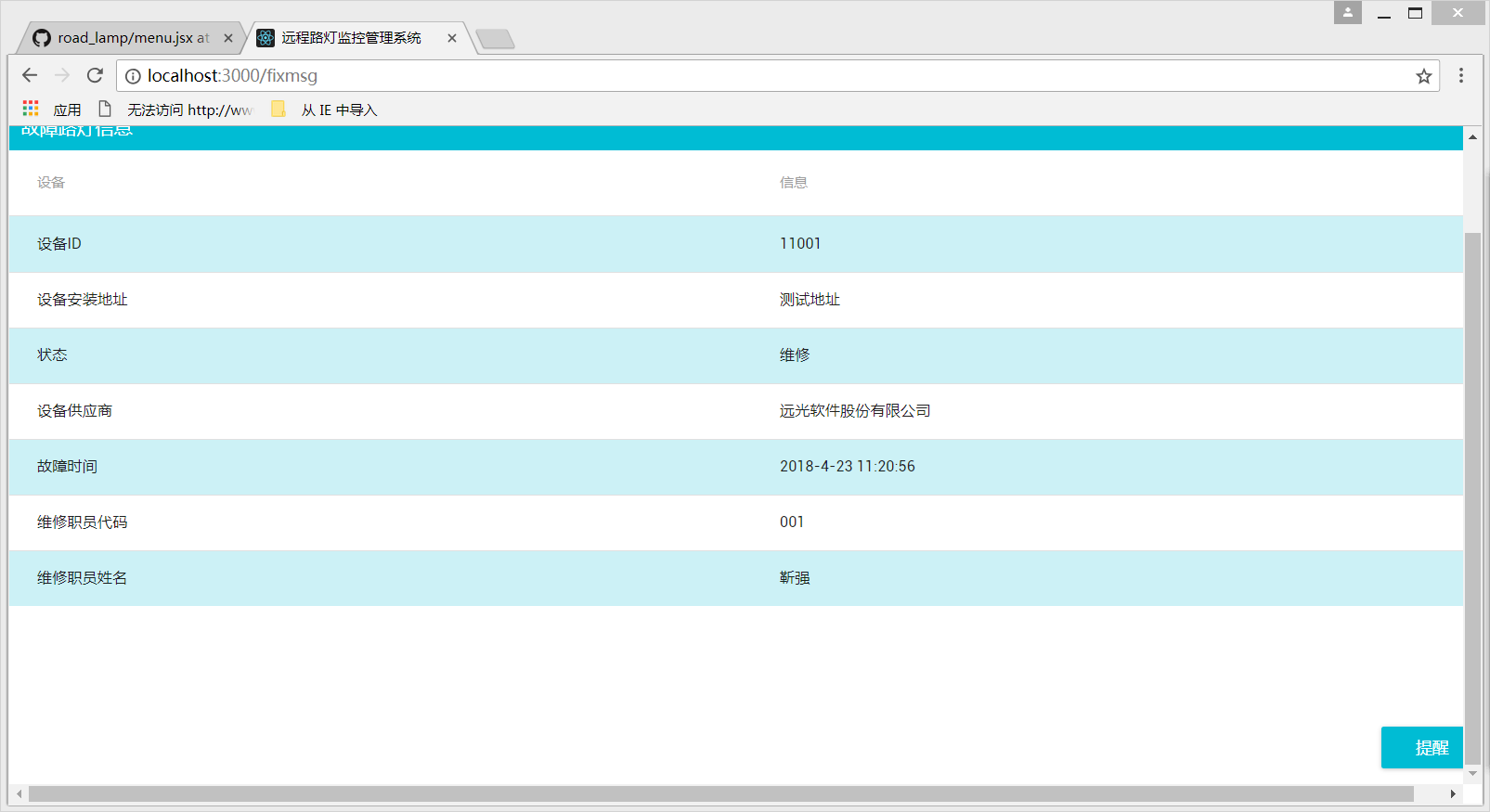
4.故障路灯信息及处理

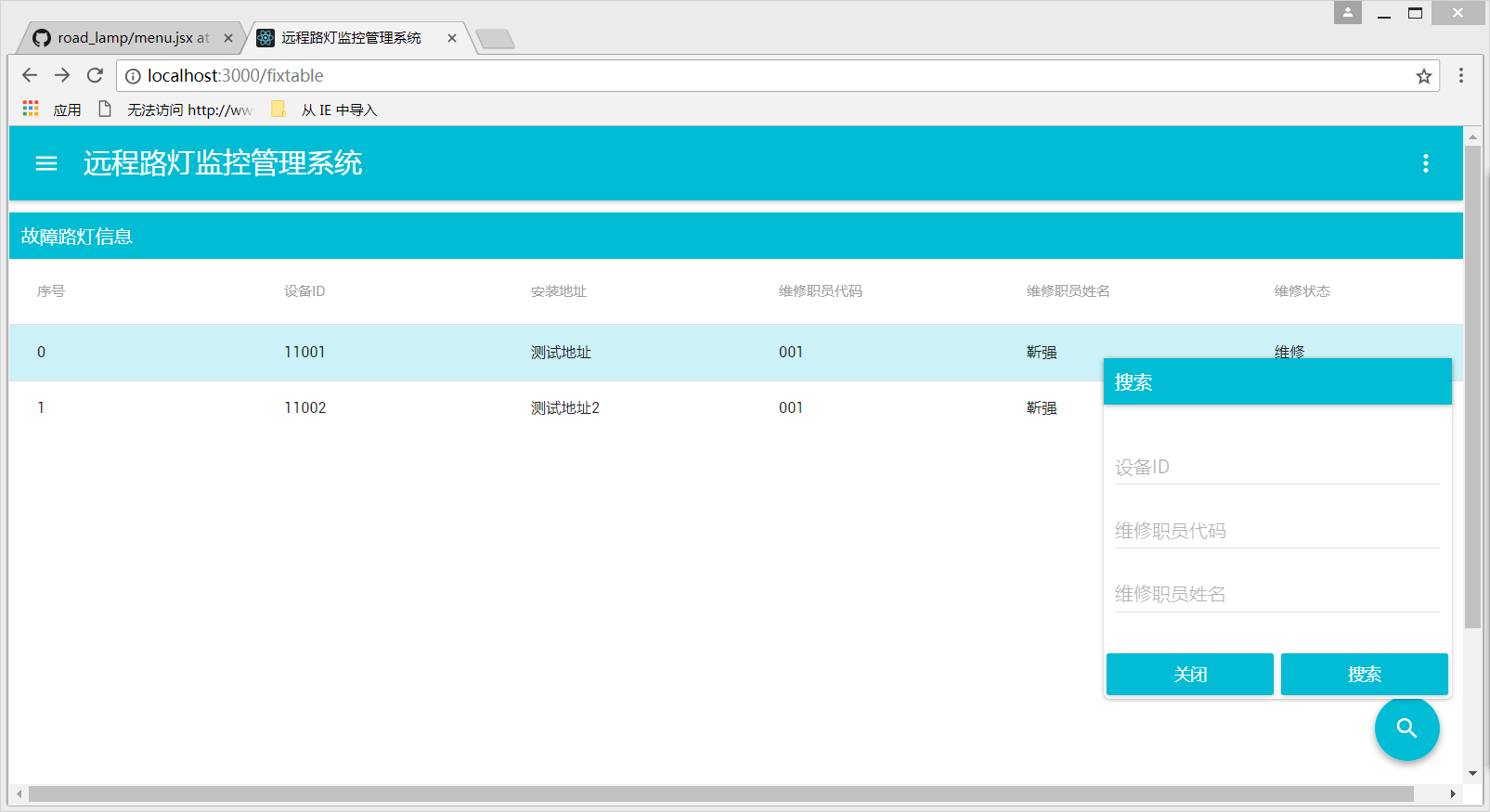




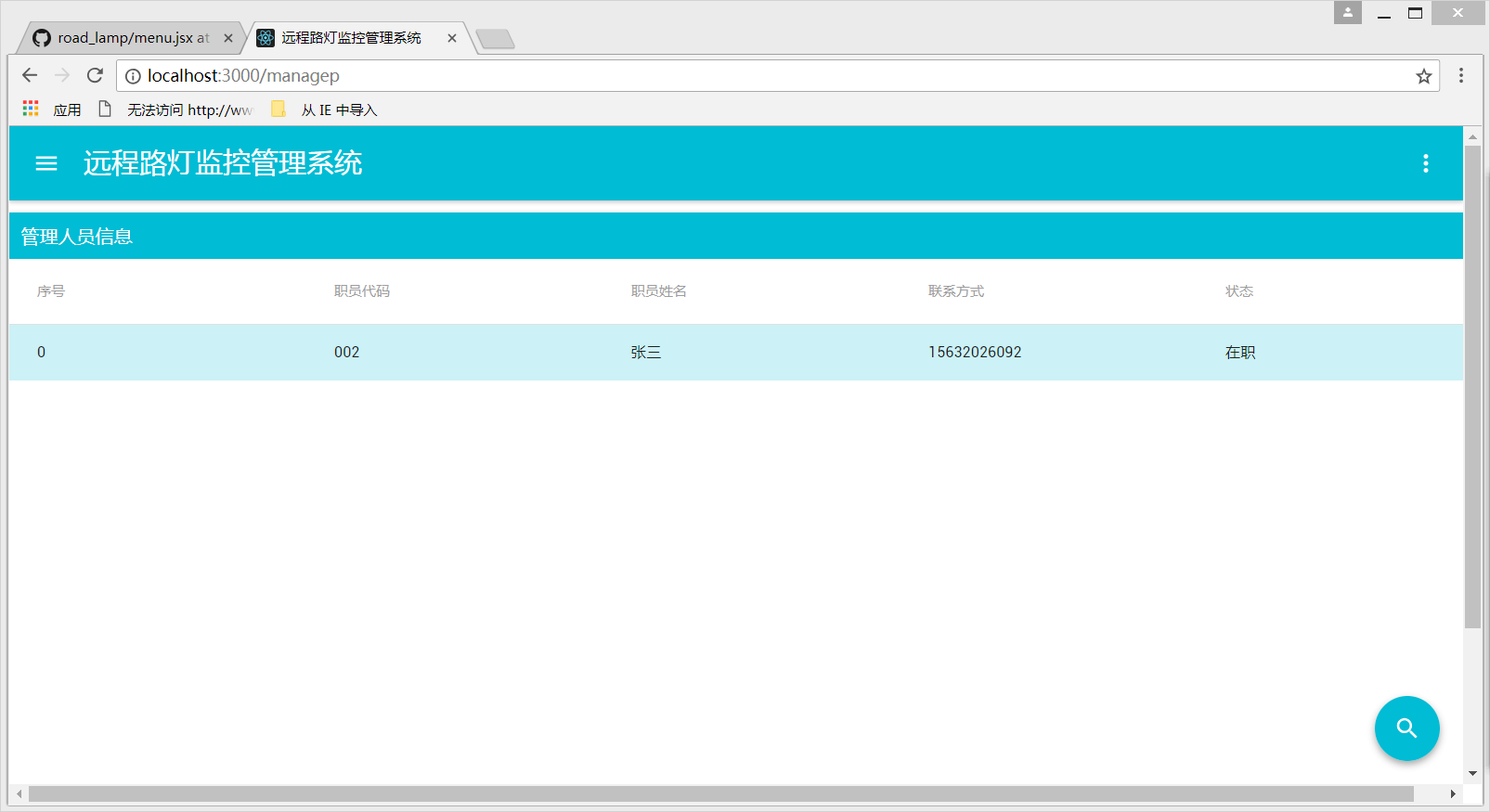
5.维修信息记录





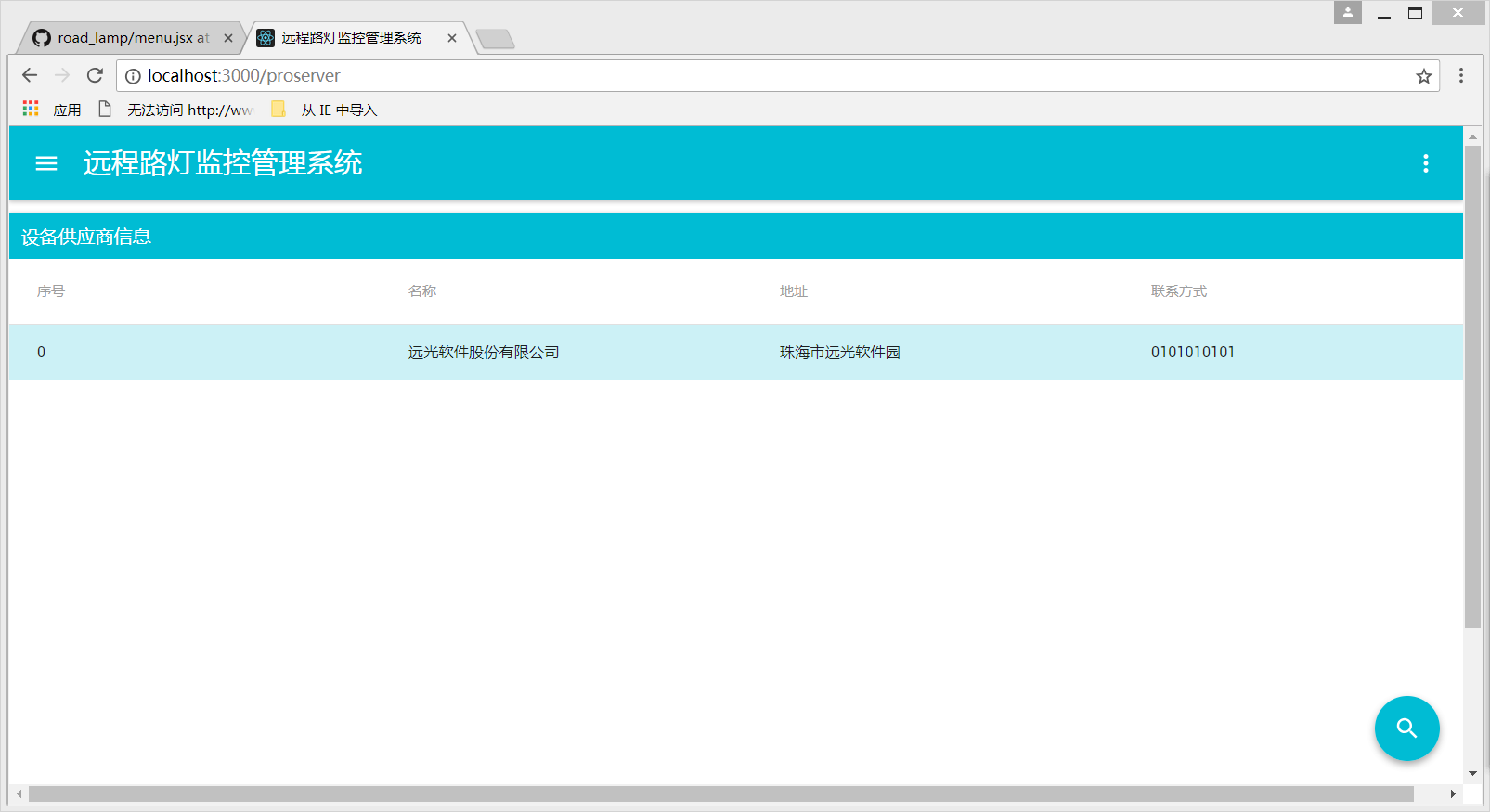


6.管理人员信息

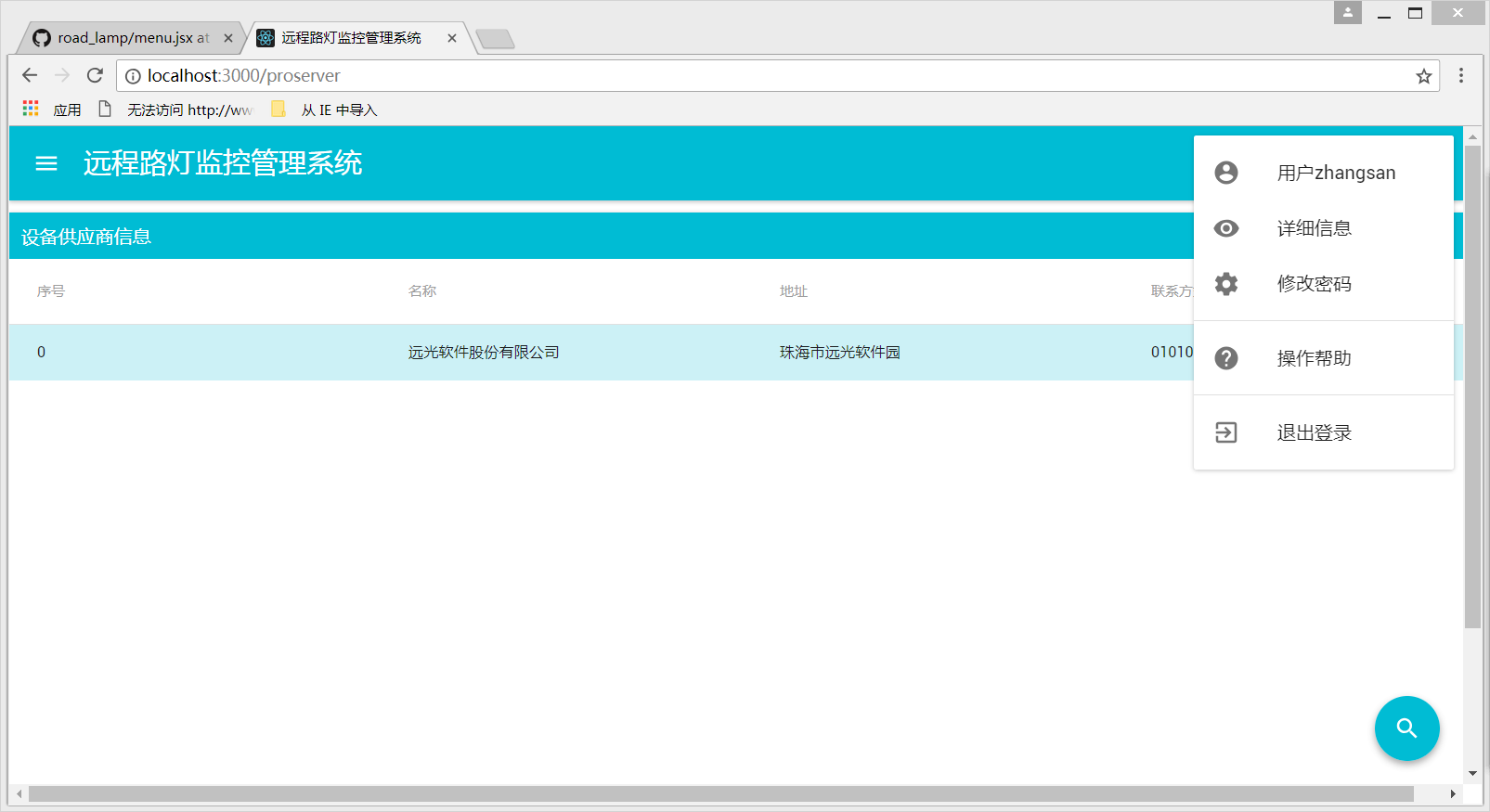


7.维修人员页面和管理人员页面相同

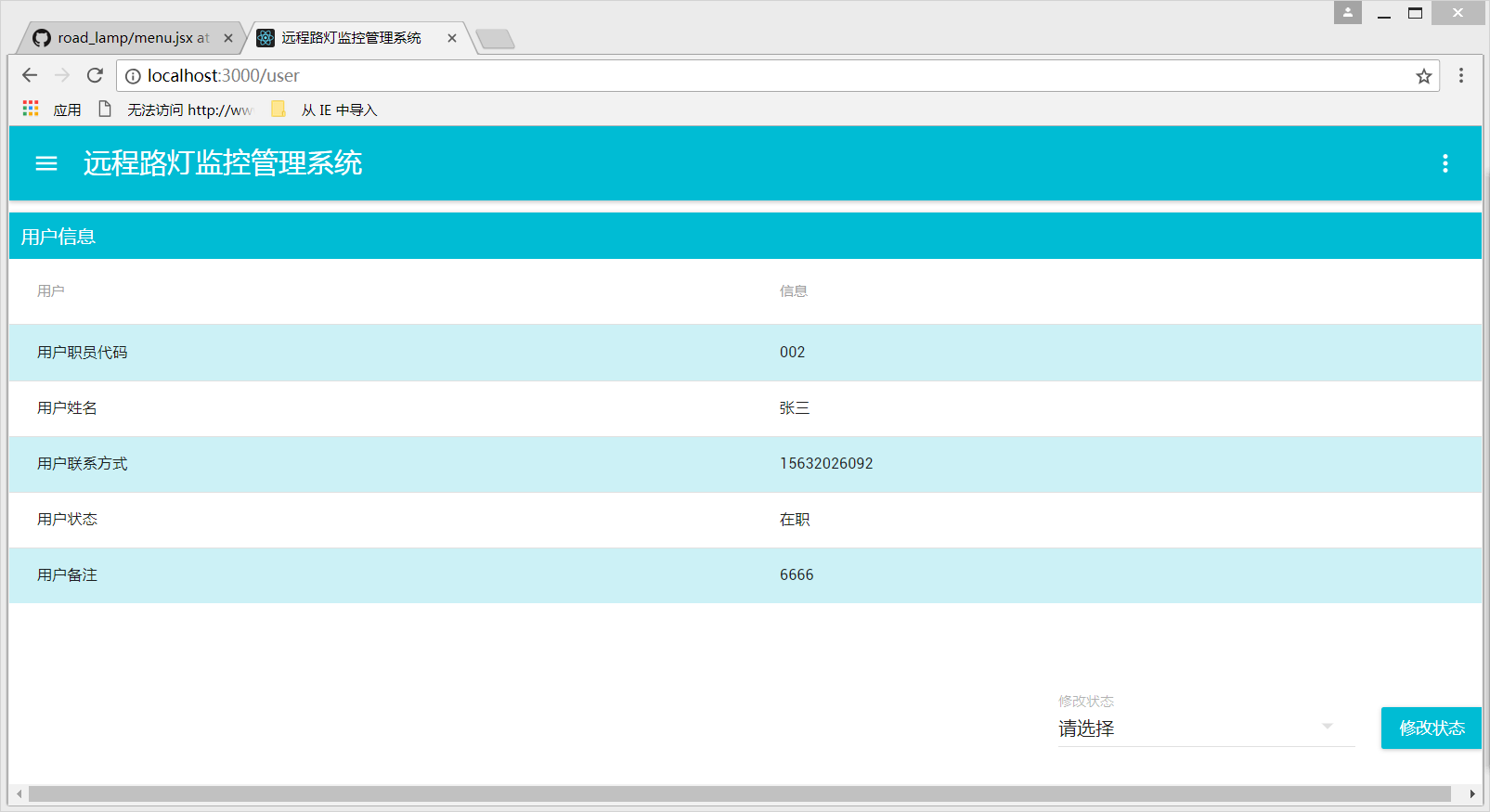
8.设备供应商页面

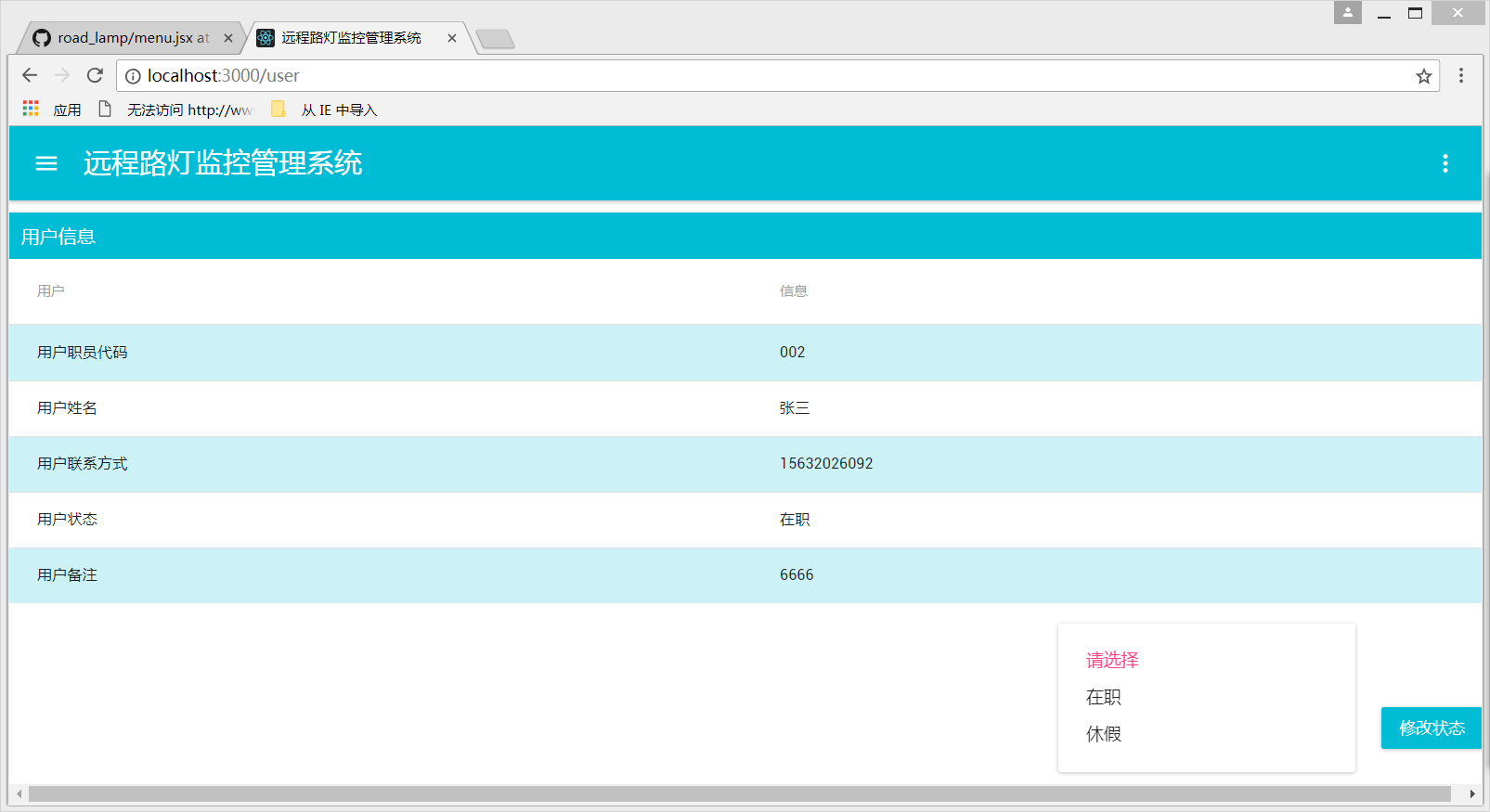


9.功能栏



10.用户详细信息、修改状态



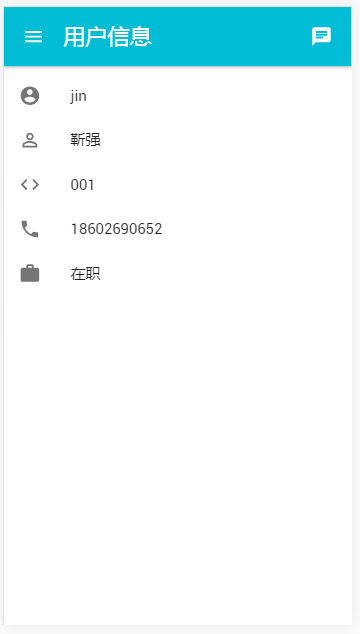
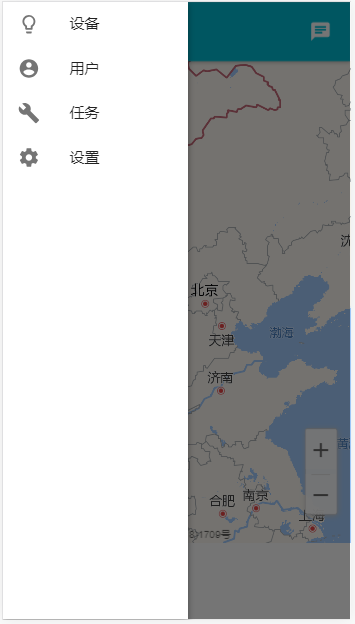


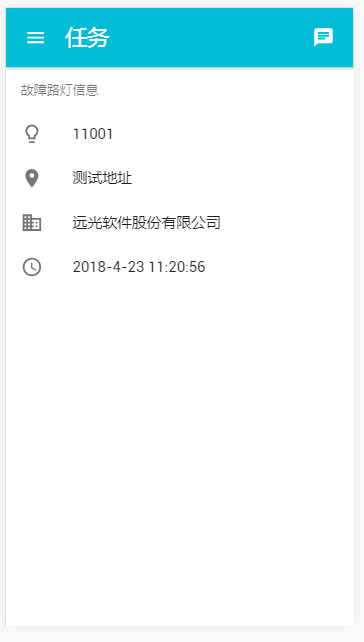
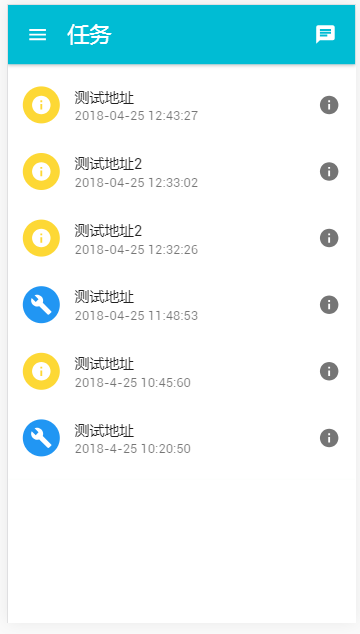
11.修改密码，修改之后会重新登录

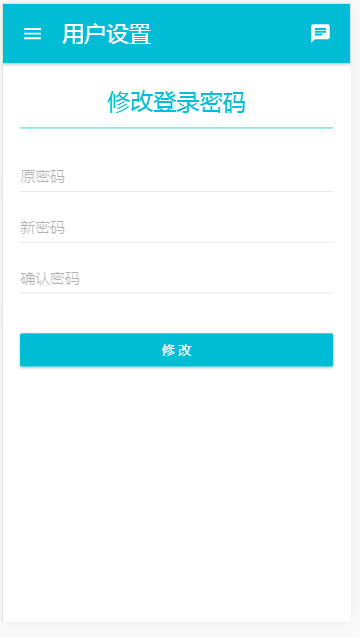
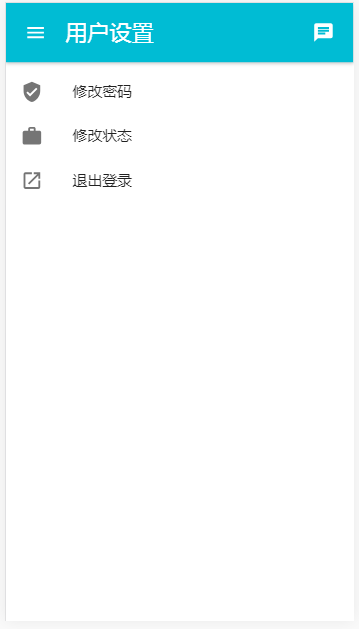


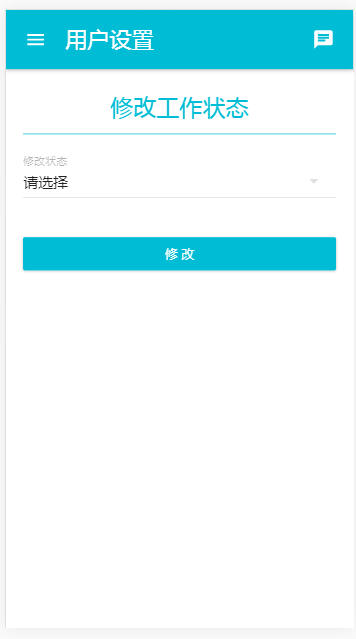
**3.3.5 app页面展示**

****

****

****

****

****

**3.4 后台设计**

**3.4.1 aop拦截web请求、使用自定义注解绑定所需要拦截的controller**

Auth.java自定义注解

package com.jin.road.auth;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

@Target({ElementType.METHOD, ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

public @interface Auth {

String s() default "auth";

}

AuthAspect.java

package com.jin.road.auth;

import static org.hamcrest.CoreMatchers.nullValue;

import javax.servlet.http.HttpServletRequest;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;

import org.springframework.web.context.request.RequestContextHolder;

import org.springframework.web.context.request.ServletRequestAttributes;

@Component

@Aspect

public class AuthAspect {

private static final Logger logger = LoggerFactory.getLogger(AuthAspect.class);

@Pointcut(value="@annotation(com.jin.road.Auth)")

public void access() {

}

@Around("@annotation(auth)")

public Object around(ProceedingJoinPoint point, Auth auth) {

ServletRequestAttributes attributes = (ServletRequestAttributes)RequestContextHolder.getRequestAttributes();

HttpServletRequest request = attributes.getRequest();

if(request.getHeader("auth").equals("road2018")) {

try {

return point.proceed();

} catch (Throwable e) {

e.printStackTrace();

return null;// TODO: handle exception

}

} else {

return null;

}

}

}

只有请求header中包含auth时才将请求分发到controller，进行处理，否则不进行处理。

**3.4.2 controller接口**

package com.jin.road.Controller;

import com.jin.road.Dao.RoadUser;

import com.jin.road.Mapper.UserMapper;

import com.jin.road.auth.Auth;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@RestController

public class LoginController {

@Autowired

private UserMapper userMapper;

@Auth

@RequestMapping(value = "/login", method = RequestMethod.POST)

public Map<String, Object> login(@RequestParam String logname, @RequestParam String pwd) throws Exception {

System.err.println(logname);

Map<String, Object> map = new HashMap<>();

RoadUser roaduser = userMapper.findByLogname(logname);

if(pwd.equals(roaduser.getPwd())) {

map.put("success", true);

map.put("data", roaduser);

} else {

map.put("success", false);

}

return map;

}

@Auth

@RequestMapping(value = "/loginwork", method = RequestMethod.POST)

public Map<String, Object> loginwork(@RequestParam String logname, @RequestParam String pwd) throws Exception {

System.err.println(logname);

Map<String, Object> map = new HashMap<>();

RoadUser roaduser = userMapper.findWorkByLogname(logname);

if(pwd.equals(roaduser.getPwd())) {

map.put("success", true);

map.put("data", roaduser);

} else {

map.put("success", false);

}

return map;

}

@Auth

@RequestMapping(value = "/changepwd", method = RequestMethod.POST)

public Map<String, Object> changepwd(@RequestParam String logname, @RequestParam String oldpwd, @RequestParam String newpwd) throws Exception {

Map<String, Object> map = new HashMap<>();

RoadUser roaduser = userMapper.findByLogname(logname);

if(oldpwd.equals(roaduser.getPwd())) {

userMapper.changepwd(newpwd, logname);

map.put("success", true);

} else {

map.put("success", false);

}

return map;

}

@Auth

@RequestMapping(value = "/changeworkpwd", method = RequestMethod.POST)

public Map<String, Object> changeworkpwd(@RequestParam String logname, @RequestParam String oldpwd, @RequestParam String newpwd) throws Exception {

Map<String, Object> map = new HashMap<>();

RoadUser roaduser = userMapper.findWorkByLogname(logname);

if(oldpwd.equals(roaduser.getPwd())) {

userMapper.changeworkpwd(newpwd, logname);

map.put("success", true);

} else {

map.put("success", false);

}

return map;

}

@Auth

@RequestMapping(value = "/changestate", method = RequestMethod.POST)

public Map<String, Object> changestate(@RequestParam String state, @RequestParam String code) throws Exception {

Map<String, Object> map = new HashMap<>();

userMapper.changestate(state, code);

map.put("success", true);

return map;

}

}

响应请求，进行数据查询，并将json格式的数据返回到客户端。

**3.4.3 配置跨域**

因为前后端分别部署，需要进行跨域设置。

private CorsConfiguration buildConfig() {

CorsConfiguration corsConfiguration = new CorsConfiguration();

corsConfiguration.addAllowedOrigin("\*");

corsConfiguration.addAllowedHeader("\*");

corsConfiguration.addAllowedMethod("\*");

return corsConfiguration;

}

/\*\*

\* 跨域过滤器

\* @return

\*/

@Bean

public CorsFilter corsFilter() {

UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", buildConfig()); // 4

return new CorsFilter(source);

}

**3.4.4 全局异常处理**

在数据请求出现错误时进行异常处理，将错误信息以json格式返回给客户端。

package com.jin.road.exception;

import java.util.HashMap;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestController;

@ControllerAdvice

@RestController

public class GlobalExceptionHandler {

private static final Logger logger = LoggerFactory.getLogger(GlobalExceptionHandler.class);

@ExceptionHandler(value = Exception.class)

public Map<String, Object> defaultErrorHandler(HttpServletRequest req, Exception e) throws Exception {

logger.error(e.getMessage());

Map<String, Object> map = new HashMap<>();

map.put("success", false);

return map;

}

}

**3.4.5 mybatis配置**

application.properties中设置包扫描范围

mybatis.mapper-locations=classpath:MapperXml/\*.xml

pom中加入配置

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

</includes>

</resource>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

Mapper 接口

package com.jin.road.Mapper;

import com.jin.road.Dao.RoadUser;

import com.jin.road.Dao.WorkUser;

import java.util.List;

import org.apache.ibatis.annotations.Mapper;

import org.apache.ibatis.annotations.Param;

import org.apache.ibatis.annotations.Select;

import org.apache.ibatis.annotations.Update;

import org.springframework.stereotype.Component;

@Component

@Mapper

public interface UserMapper {

RoadUser findByLogname(@Param("logname") String logname);

RoadUser findWorkByLogname(@Param("logname") String logname);

@Update("UPDATE LOGUSER SET pwd = #{pwd} WHERE logname LIKE #{logname}")

void changepwd(@Param("pwd") String pwd, @Param("logname") String logname);

@Update("UPDATE LOGINWORK SET pwd = #{pwd} WHERE logname LIKE #{logname}")

void changeworkpwd(@Param("pwd") String pwd, @Param("logname") String logname);

@Update("UPDATE USER SET state = #{state} WHERE code LIKE #{code}")

void changestate(@Param("state") String state, @Param("code") String code);

@Update("UPDATE USER SET STATE = #{state} WHERE code LIKE #{code}")

void fix(@Param("state") String state, @Param("code") String code);

@Select("SELECT \* FROM USER WHERE POSITION LIKE #{position}")

List<WorkUser> findByPosition(@Param("position") String position);

@Select("SELECT \* FROM USER WHERE POSITION LIKE '维修工' AND STATE LIKE '在职'")

List<WorkUser> findWork();

}

复杂的sql语句可以使用xml格式

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.jin.road.Mapper.UserMapper" >

<select id="findByLogname" resultType="com.jin.road.Dao.RoadUser">

SELECT loguser.id, loguser.logname, loguser.pwd, loguser.code , user.name, user.phone, user.state, user.remark

FROM loguser

LEFT JOIN user

ON loguser.code like user.code

WHERE logname like #{logname}

</select>

<select id="findWorkByLogname" resultType="com.jin.road.Dao.RoadUser">

SELECT loginwork.id, loginwork.logname, loginwork.pwd, loginwork.code , user.name, user.phone, user.state, user.remark

FROM loginwork

LEFT JOIN user

ON loginwork.code like user.code

WHERE logname like #{logname}

</select>

</mapper>

命名空间写需要引入的接口名称。

**3.4.6 mysql配置**

application.properties加入

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource

spring.datasource.url=jdbc:mysql://localhost:3306/road?useUnicode=true&characterEncoding=utf8&useSSL=false

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.initialSize=5

spring.datasource.minIdle=5

spring.datasource.maxActive=20

spring.datasource.maxWait=60000

spring.datasource.timeBetweenEvictionRunsMillis=60000

spring.datasource.minEvictableIdleTimeMillis=300000

spring.datasource.validationQuery=SELECT 1 FROM DUAL

spring.datasource.testWhileIdle=true

spring.datasource.testOnBorrow=false

spring.datasource.testOnReturn=false

spring.datasource.poolPreparedStatements=true

spring.datasource.maxPoolPreparedStatementPerConnectionSize=20

spring.datasource.filters=stat,wall,log4j

spring.datasource.connectionProperties=druid.stat.mergeSql=true;druid.stat.slowSqlMillis=5000

这里我使用了durid连接池。

3.3.6.7 logback配置

Logback-boot.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>

%d %p (%file:%line\)- %m%n

</pattern>

<charset>UTF-8</charset>

</encoder>

</appender>

<appender name="roadlog" class="ch.qos.logback.core.rolling.RollingFileAppender">

<File>log/sys.log</File>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>log/road.%d.%i.log</fileNamePattern>

<maxHistory>30</maxHistory>

<timeBasedFileNamingAndTriggeringPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">

<maxFileSize>10MB</maxFileSize>

</timeBasedFileNamingAndTriggeringPolicy>

</rollingPolicy>

<encoder>

<pattern>

%d %p (%file:%line\)- %m%n

</pattern>

<charset>UTF-8</charset>

</encoder>

</appender>

<root level="info">

<appender-ref ref="STDOUT" />

</root>

<logger name="com.jin" level="DEBUG">

<appender-ref ref="roadlog" />

</logger>

</configuration>

这里配置了日志记录的格式，显示级别，命名方式，存储位置。

Application.properties加入配置文件位置

logging.config=classpath:logback-boot.xml

这样就可以实现日志的记录。

1. **项目测试**

**4.1 测试结果说明**

编写好前后端后，进行项目的测试，查看数据导入是否正确，登录是否能够成功，处理故障路灯信息是否成功，提醒维修是否成功，搜索结果显示是否正确，记住密码功能是否正常，用户操作是否正常，是否流畅，app端接收信息是否成功。

继续按此顺序进行其他测试。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 用户 | 密码 | 注册 | 信息 | 修改密码 | 退出登录 | 修改状态 |
| zhangsan | 123456 | 成功 | 准确 | 成功 | 成功 | 成功 |
| jin | 123456 | 成功 | 准确 | 成功 | 成功 | 成功 |
| lisi | 123456 | 成功 | 准确 | 成功 | 成功 | 成功 |

测试效果均成功。

**4.2 测试局限性分析**

测试时是在chrome下调试运行，可能测试结果可信度不高，测试时数据量不大，可能存在数据处理方面的不足，数据的实时反应可能会使系统卡顿。

但系统设计逻辑清晰，基本实现了预期功能，测试用例反应也良好，操作很流畅。可以说此次系统的设计基本算是成功的。

**结论**

完成这个毕业设计前前后后大约花费了2个月的时间，总体来说，过程还是比较顺利的，但有时候也会遇到一些棘手，难以解决的问题，比如mybatis扫描不到xml配置，axios的跨域问题，多表联查，统一异常处理，aop连接请求。都需要查看大量的资料，但是经过网络查找解决方法，这些问题都得到了解决，同时也让我有了很多的项目实战经验，为以后的工作打下了好的基础。

经过此次毕业设计，我不仅熟悉了react、redux、springboot等等框架的使用方法，同时也完成了研究内容，react相比于传统的html，更加简洁，更加高效。Springboot简化了sprng的xml配置，自动装配，简洁的resource配置就可以完成庞大的工作，极大的简化了开发人员的工作量，以后一定会更加火爆。uo经过这个系统的设计与实现，我对如何开发单页面应用有了一定的了解，对后台的编写也有了经验，现在流行的技术框架基本在本项目中都用到了，虽然系统可能功能不能达到商业用途，但是也是一个好的开端。总之，这次毕业设计我收获了很多，同时作品效果也令人满意

**附录**

**附录1 主要参考资料**

[1]顾勇.浅析城市路灯的建设与管理[J].经济视野，2013（24）：23～24.

[2]翟芳，冯学奎，张悦.城市路灯建设与管理工作思考[J].城市，2009（9）：76～78.

[3]耿桂英.浅谈路灯建设与管理的主要问题[J].城市建设理论研究：电子版，2012：11～12.

[4]郑远鹏.城市路灯建设措施及管理工作开展分析[J].建筑工程技术与设计，2014（32）：36～37.

[5]陈屹，深入React技术栈.人民邮电出版社

[6]程墨，深入浅出React和Redux，机械工业出版社.

[7]克雷格.沃斯，Spring Boot实战，人民邮电出版社

[8]Brian Goetz,Tim Peierls,Joshua Bloch，Java Concurrentcy in Practice，机械工业出版社.

[9]Bohme T, Rahm E. Supporting Efficient Streaming and Insertion of XML Data inRDBMS[C] / /Proc. 3rd Int. Workshop Data Integration over the Web (D IWeb) ,2004.

[10]陈韶健，深入实践Spring Boot，机械工业出版社.

[11]杨开振，深入浅出MyBatis技术原理与实战，电子工业出版社

[12]福塔，MySQL必知必会，人民邮电出版社.

[13]Dilwyn Edards, Mike Hamson. Guide to Mathematical Mdellng. WorldPublishing Corp. 1989

[14]James Laudon, Lawrence Spracklen. The Coming Wave of Multithreaded ChipMultiprocessors[J]. International Journal of Parallel Programming . 2007 (3)

[15]Y.Liu , C.J.Liu. Scheduling strategy design of thread pool for real-time controlsystem,Computer Engineering and Applications. 2010. 46(32): 71-73