**武汉大学计算机学院**

**本科生实验报告**

**软件架构与体系结构实验报告文档**

专 业 名 称 ： 计算机科学与技术

课 程 名 称 ： 软件设计与体系结构

指 导 教 师 ： 王健

团 队 成 员 ： 金泽辉

团 队 成 员 ： 朱家伟

目录

[1实验内容介绍 3](#_Toc56808242)

[1.1项目介绍 3](#_Toc56808243)

[1.2设计模式 3](#_Toc56808244)

[2实验过程 4](#_Toc56808245)

[2.1 后端 4](#_Toc56808246)

[主要功能 5](#_Toc56808247)

[数据库设计 7](#_Toc56808248)

[Dao层设计 7](#_Toc56808249)

[Service层设计 15](#_Toc56808250)

[Utils 层 16](#_Toc56808251)

[Controller层 18](#_Toc56808252)

[2.2 前端 19](#_Toc56808253)

[Main.js 入口文件编写 20](#_Toc56808254)

[Router 路由文件编写 21](#_Toc56808255)

[store 文件（vuex部分） 22](#_Toc56808256)

[axios封装 23](#_Toc56808257)

[组件编写 25](#_Toc56808258)

[3 实验结果 26](#_Toc56808259)

[3.1 后端Restful api文档展示 26](#_Toc56808260)

[3.2 前端结果展示 29](#_Toc56808261)

[4成员分工与贡献 30](#_Toc56808262)

# 1实验内容介绍

## 1.1项目介绍

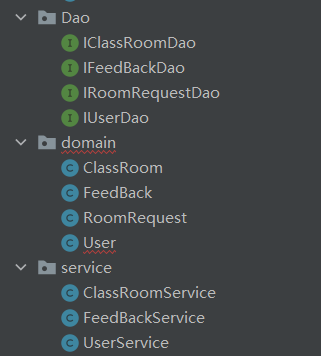
本项目是一个教室管理系统，目的是给老师以及同学们提供一个完整的校园教室租借申请，以及申请的处理的解决方案。

后端为springboot+mybatis的Restful api 提供了本项目中所需要用到的全部api，前端则为vue的多页面交互程序，但由于vue的了解不深只做了一两个简单的界面。

## 1.2设计模式

本项目主要用到的设计模式为抽象工厂模式，数据访问模式和MVC模式。

抽象工厂模式：在本项目中，所有的User，Classroom，Request，Feedback都被封装成了一个个类作为Domain层，在其基础上建立Dao层和Services层。



通过将各种常用类通过封装，我们可以实现直接通过统一的接口从高层调用Service层，Service层再层层下降来调用Dao和domain层，从而实现对操作类的不同实例的快速操作。

数据访问模式：数据访问模式更不用多说，本项目在后端将domain层和mysql中的数据紧密联系，高层则通过mybatis提供的mapper将数据操作语句和具体的sql语句进行紧密贴合，最终完成对mysql中对象的高效读取和修改。

MVC模式：在本项目中的mvc模式中，我们实现了前后端分离操作，于后端部分中使用Rest api完成api的搭建，通过swagger拓展包完成对api文档的详细搭建。前端由vue实现，其中http操作主要通过axios完成，通过前后端分离，可以使得我们的应用开发分工更加明确，同时后端仅提供api和前端主要实现页面绘制也可以方便应用的扩建。

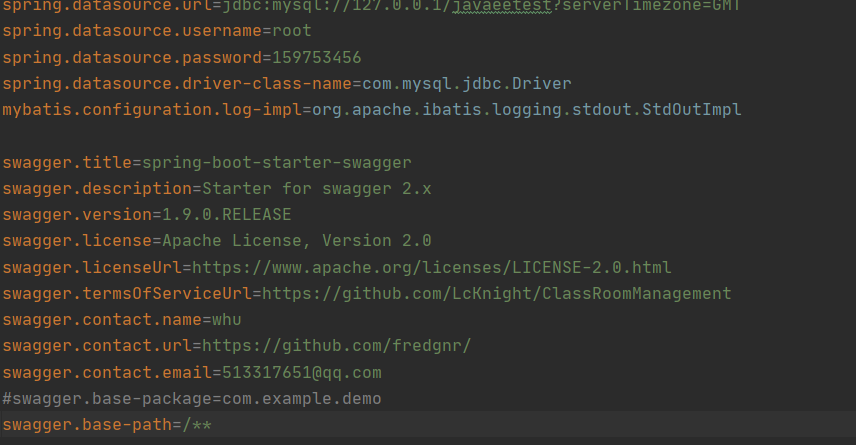
# 2实验过程

## 2.1 后端

后端博客如下：https://blog.csdn.net/qq\_41076492/article/details/109780286

后端主要通过springboot搭建，其中主要用到mybatis，相比于传统的springmvc模式，springboot搭建更快捷也更加容易部署。

项目的部分设定参考如下：



其中包括了数据库datasource设置和一些项目的相关信息。

本系统后端利用springboot+mybatis作为基础架构，实现了对于数据库的操作以及与前端进行交互的RestAPI接口。功能大致分为四类，用户基本操作、教室信息操作、反馈操作以及教室预约操作。本系统实现了前三类功能，教室预约操作本系统没有实现。

### 主要功能

1. 用户操作

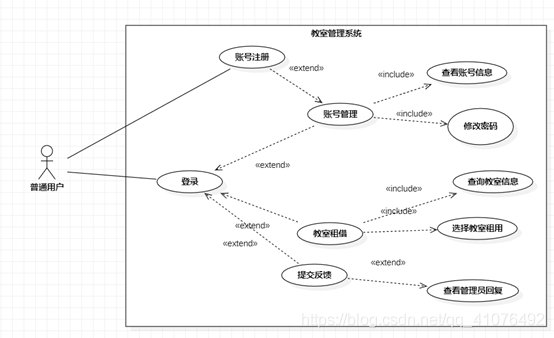
管理员操作  
查看所有用户信息  
删除用户  
登录  
普通用户操作  
登录  
修改密码  
修改姓名  
注册

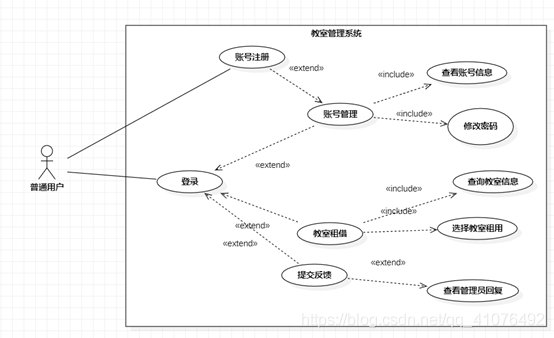
1. 教室操作

普通用户操作  
查看教室信息  
管理员操作  
增加教室  
删除教室  
修改教室信息

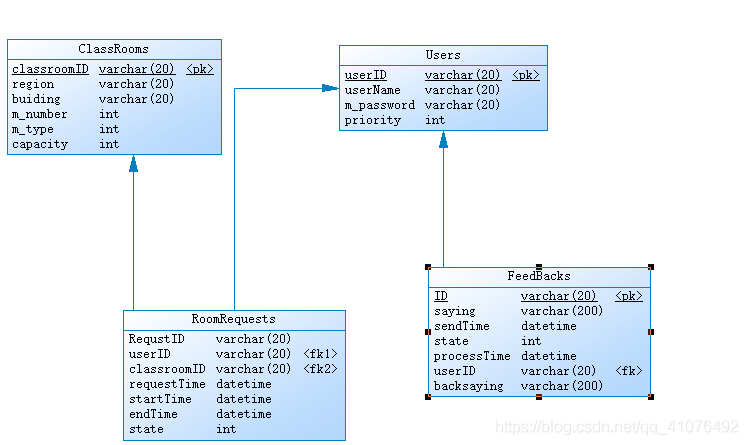
1. 反馈操作

普通用户操作  
用户提交反馈  
用户查看自己提交的反馈以及回复  
查看所有反馈  
管理员操作  
回复反馈





### 数据库设计



User表记录了用户信息，包括普通用户与管理员，通过权限（priority）字段字段进行区分

Classroom表记录了所有教室信息

Feedback表记录了所有反馈的信息，包括反馈时间与管理员回复

RoomRequest表记录了所有的教室请求

### Dao层设计

IUserDao此接口实现了对User表的crud操作

代码如下：

package com.example.demo.Dao;

import com.example.demo.domain.User;

import org.apache.ibatis.annotations.\*;

import org.springframework.context.annotation.Bean;

import org.springframework.scheduling.support.SimpleTriggerContext;

import org.springframework.stereotype.Repository;

import java.util.List;

@Mapper

@Repository

public interface IUserDao {

    //获取所有的用户信息

    @Select("select \*from Users")

    List<User> findAll();

    //插入用户记录

    @Insert("insert into Users(userID,userName,m\_password,priority) values(#{userID},#{userName},#{m\_password},#{priority})")

    void saveUser(User user);

    //根据UserID查询用户信息

    @Select("select \*from Users where userID=#{userID}")

    User findByID(@Param("userID") String userID);

    //查询特定权限的用户

    @Select("select \*from Users where priority=#{priority}")

    List<User> findByPriority(@Param("priority") Integer priority);

    //根据姓名查询用户

    @Select("select \*from Users where userName=#{userName}")

    User findByName(@Param("userName") String userName);

    //删除用户

    @Delete("delete from Users where userID=#{userID}")

    void delete(User user);

    //更新记录

    @Update("update Users set " +

            "userName=#{userName},m\_password=#{m\_password},priority=#{priority}" +

            " where userID=#{userID}")

    void update(User user);

     //根据姓名对账号进行模糊搜索

    @Select("select \* from Users where userName like CONCAT('%',#{namepart},'%')")

    List<User> findNamePart(@Param("namepart") String namepart);

}

2.IClassRoomDao此接口实现了对ClassRoom表的查询 代码如下：

package com.example.demo.Dao;

import com.example.demo.domain.ClassRoom;

import org.apache.ibatis.annotations.\*;

import org.springframework.stereotype.Repository;

import java.math.BigInteger;

import java.util.List;

@Mapper

@Repository

public interface IClassRoomDao {

    @Select("select \* from ClassRooms")

    List<ClassRoom> findAll();

    @Select("select \* from ClassRooms where classroomID=#{ID}")

    ClassRoom findByID(BigInteger ID);

    @Select("<script>"+

            "select \* from ClassRooms " +

            "<where>"+

            "<if test='region!=null'>"+

            " and region like CONCAT('%',#{region},'%')"

            +"</if>"+

            "<if test='building!=null'>"+

            " and building like CONCAT('%',#{building},'%')"

            +"</if>" +

            "<if test='capacity!=null'>"+

            " and capacity>=#{capacity}"

            +"</if>" +

            "<if test='m\_type!=null'>"+

            " and m\_type=#{m\_type}"

            +"</if>" +

            "<if test='m\_number!=null'>"+

            " and m\_number=#{m\_number}"

            +"</if>" +

            "</where>"+

        "</script>")

    List<ClassRoom> findByArgs(

            @Param("region") String region,

            @Param("building") String building,

            @Param("m\_number") Integer number,

            @Param("m\_type") Integer type,

            @Param("capacity") Integer capacity

    );

    @Insert("insert into ClassRooms(region,building,m\_number,m\_type,capacity) " +

            "values(#{region},#{building},#{m\_number},#{m\_type},#{capacity})")

    @Options(useGeneratedKeys = true,keyProperty = "classroomID",keyColumn = "ID")

    void addClassRoom(ClassRoom classRoom);

    @Update("update ClassRooms set " +

            "region=#{region},building=#{building},m\_number=#{m\_number},m\_type=#{m\_type},capacity=#{capacity}" +

            "where classroomID=#{classroomID}")

    void updateClassRoom(ClassRoom classRoom);

    @Delete("delete from ClassRooms where classroomID=#{id}")

    void deleteClassRoom(@Param("id") BigInteger id);

    @Select("select \* from ClassRooms" +

            " where capacity>=#{a} ")

    List<ClassRoom> findByCapcity(@Param("a") Integer a);

}

IFeedBackDao对FeedBack表进行crud操作

代码如下：

package com.example.demo.Dao;

import com.example.demo.domain.FeedBack;

import org.apache.ibatis.annotations.\*;

import org.springframework.stereotype.Repository;

import java.math.BigInteger;

import java.sql.Time;

import java.sql.Timestamp;

import java.util.List;

@Mapper

@Repository

public interface IFeedBackDao {

    @Select("select \* from FeedBacks")

    List<FeedBack> findAll();

    @Select("select \* from FeedBacks WHERE ID=#{ID}")

    FeedBack findByID(@Param("ID") Integer id);

    @Select("<script>"+

            "select \* from FeedBacks " +

            "<where> " +

            "<if test='saying!=null'>" +

            " and saying like CONCAT('%',#{saying},'%') " +

            "</if> " +

            "<if test='state!=null'>" +

            " and  state=#{state} " +

            "</if> " +

            "<if test='userID!=null'>" +

            " and userID=#{userID} " +

            "</if>" +

            "<if test='backsaying!=null'>" +

            " and backsaying like CONCAT('%',#{backsaying},'%') " +

            "</if>" +

            "<if test='startleft!=null and startright!=null'>" +

            " and (unix\_timestamp(#{startleft})&lt;unix\_timestamp(sendTime)" +

            "   and unix\_timestamp(sendTime)&lt;unix\_timestamp(#{startright}))" +

            "</if>" +

            "<if test='endleft!=null and endright!=null'>" +

            " and (state=1" +

            " and unix\_timestamp(#{endleft})&lt;unix\_timestamp(processTime)" +

            "   and unix\_timestamp(processTime)&lt;unix\_timestamp(#{endright}) )" +

            "</if>" +

            "</where>"+

            "</script>")

    List<FeedBack> findByArgs(

            @Param("saying") String saying ,

            @Param("state") Integer state,

            @Param("userID") String userID,

            @Param("backsaying") String backsaying,

            @Param("startleft") Timestamp startleft,

            @Param("startright") Timestamp startright,

            @Param("endleft") Timestamp endleft,

            @Param("endright") Timestamp endright

    );

    @Options(useGeneratedKeys = true,keyProperty = "ID",keyColumn = "id")

    @Insert("insert into FeedBacks(saying ,sendTime,state,processTime,userID,backsaying) " +

            "values(#{saying} ,#{sendTime},#{state},#{processTime},#{userID},#{backsaying})")

    void addFeedBack(FeedBack feedBack);

    @Delete("delete from FeedBacks where ID=#{ID}")

    void deleteFeedBack(FeedBack feedBack);

    @Update("update FeedBacks set " +

            "saying=#{saying}," +

            "sendTime=#{sendTime}," +

            "state=#{state}" +

            "processTime=#{processTime}" +

            "userID=#{userID}" +

            "backsaying=#{backsaying}" +

            "where ID=#{ID}")

    void updateFeedBack(FeedBack feedBack);

    @Select("select \* from FeedBacks where state=0")

    List<FeedBack> findUndoFeedBack();

    @Select("select \* from FeedBacks where userID=#{userID}")

    List<FeedBack> findByUserID(@Param("userID") String userid);

    @Select("select \* from FeedBacks " +

            "where unix\_timestamp(#{start})<=unix\_timestamp(sendTime)" +

            "and unix\_timestamp(#{end})>=unix\_timestamp(sendTime)")

    List<FeedBack> findBySendTime(@Param("start") Timestamp start,@Param("end") Timestamp end);

    @Select("select \* from FeedBacks " +

            "where unix\_timestamp(#{start})<=unix\_timestamp(processTime)" +

            "and unix\_timestamp(#{end})>=unix\_timestamp(processTime)" +

            " and state=2")

    List<FeedBack> findByProcessTime(@Param("start") Timestamp start,@Param("end") Timestamp end);

}

**4.IRoomRequestDao**

对RoomRequest表进行crud操作 代码如下：

package com.example.demo.Dao;

import com.example.demo.domain.FeedBack;

import com.example.demo.domain.RoomRequest;

import org.apache.ibatis.annotations.\*;

import org.springframework.stereotype.Repository;

import java.math.BigInteger;

import java.sql.Timestamp;

import java.util.List;

@Mapper

@Repository

public interface IRoomRequestDao {

    @Select("select \* from RoomRequests")

    List<RoomRequest> findAll();

    @Select("<script>"+

            "select \* from RoomRequests " +

            "<where>" +

            "<if test='RequstID!=null'>" +

            " and RequstID=#{RequstID} " +

            "</if>" +

            "<if test='userID!=null'>" +

            " and userID=#{userID} " +

            "</if>" +

            "<if test='classroomID!=null'>" +

            " and classroomID=#{classroomID} " +

            "</if>" +

            "<if test='state!=null'>" +

            " and state=#{state} " +

            "</if>" +

            "</where>"+

            "</script>")

    List<RoomRequest> findByArgs(

           @Param("RequstID") BigInteger RequstID,

           @Param("userID") String userID,

           @Param("classroomID") BigInteger classroomID ,

           @Param("state") Integer state

    );

  @Insert("insert into roomrequests(RequstID ,userID,classroomID ,requestTime ,startTime ,endTime, state)"

          + "  values(#{RequstID} ,#{userID},#{classroomID} ,#{requestTime} ,#{startTime} ,#{endTime}, #{state})")

  void addRoomReqeust(RoomRequest roomRequest);

    @Delete("delete from roomrequests where RequstID=#{RequstID}")

    void deleteRoomReqeust(RoomRequest roomRequest);

    @Update("update roomrequests set " +

            "userID=#{userID}," +

            "classroomID=#{classroomID} ," +

            "requestTime=#{requestTime} ," +

            "startTime=#{startTime} ," +

            "endTime=#{endTime}," +

            "state=#{state}" +

            "where RequstID=#{RequstID}")

    void updateRoomReqeust(RoomRequest roomRequest);

    @Select("select \* FROM RoomRequests " +

            "where unix\_timestamp(startTime)>=unix\_timestamp(#{timestamp})")

    List<RoomRequest> findByStartTime(@Param("timestamp") Timestamp timestamp);

    @Select("select \* FROM RoomReqeusts " +

            "where unix\_timestamp(endTime)<=unix\_timestamp(#{timestamp})")

    List<RoomRequest> findByEndTime(@Param("timestamp") Timestamp timestamp);

    @Select("<script>" +

            "select \* FROM  RoomReqeusts " +

            "<where>" +

            "<if test='startt!=null'>" +

            " and  unix\_timestamp(requestTime)&lt;=unix\_timestamp(#{startt})" +

            "</if>" +

            "<if test='endt!=null'>" +

            " and unix\_timestamp(requestTime)&lt;=unix\_timestamp(#{endt})" +

            "</if>" +

            "</where>" +

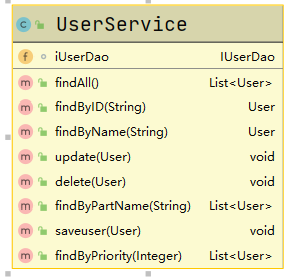
            "</script>")

    List<RoomRequest> findbyRequestTime(@Param("startt") Timestamp start,@Param("endt") Timestamp end);

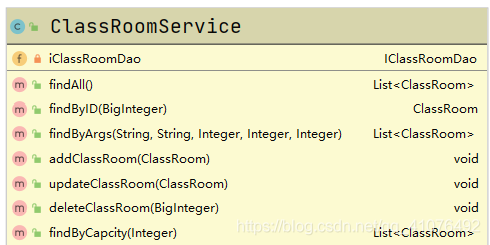
}

### Service层设计

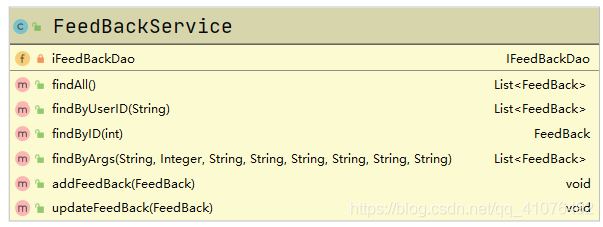
1.UserService

对UserDao的方法进行封装 

2.ClassRoomService

对ClassroomDao中的接口进行封装 

3.FeedBackService

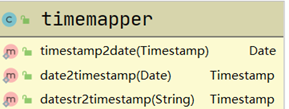
对FeedBackDao中的接口进行封装 

### Utils 层

包含程序所需要的工具类

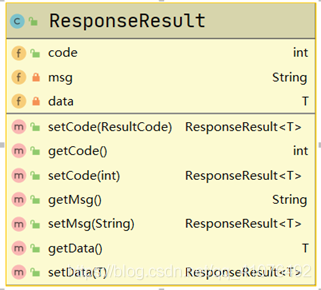
1. Timemapper

实现时间相关的类以及字符串在不同格式下的转换



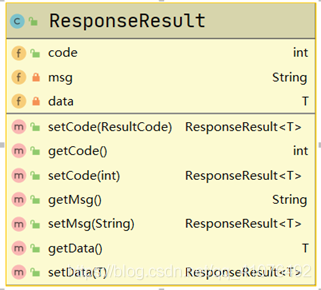
2. ResponseResult

封装向前端返回的数据。Code为具体的标识码，与下面Resultcode中的枚举类一一对应。Msg为具体的提示信息，data为数据类，模板



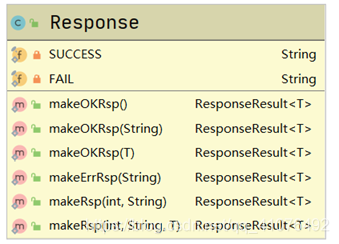
2. ResponseResult

封装向前端返回的数据。Code为具体的标识码，与下面Resultcode中的枚举类一一对应。Msg为具体的提示信息，data为数据类，模板



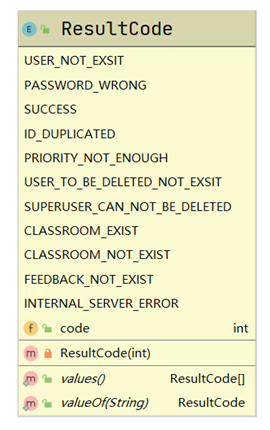
3. Response

使用不同的函数生成不同的响应信息类



4. ResultCode

枚举类，每一个枚举变量对应查询时的一种状态



### Controller层

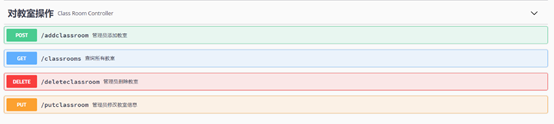
提供用于与前端进行交互的API

1、User Controller

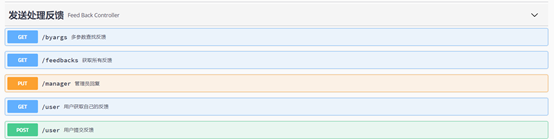
用户操作



2、Classroom Controller

教室信息操作 

3、FeedBack Controller

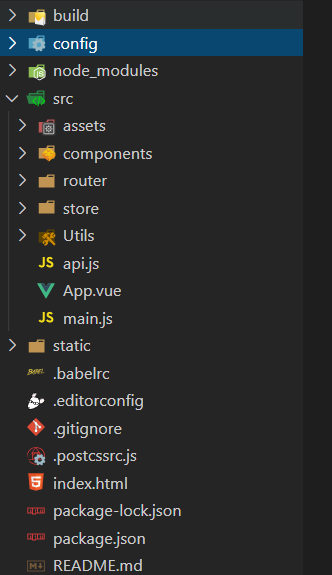


处理反馈

## 2.2 前端

前端主要建立在Vue.js框架之上编写而成，前端部分由于熟练度不够仅实现了登陆和展示教室功能，但是基本已经完成了了整体框架。其中包括了项目的路由框架，前端拦截器，token刷新和验证，部分组件的实现。

其中前端项目框架如下：



### Main.js 入口文件编写

Main.js文件主要完成了项目依赖的注册和根vue元素的实例化，如下，主要包括导入了router文件，axios的封装文件，store（即vuex的主要文件）以及通过render来渲染了根vue文件。

// The Vue build version to load with the `import` command

// (runtime-only or standalone) has been set in webpack.base.conf with an alias.

import Vue from 'vue'

import App from './App.vue'

import router from './router'

import axios from './Utils/https'

import store from "./store/store"

Vue.prototype.axios=axios;

/\* eslint-disable no-new \*/

new Vue({

  el:'#app',

  axios,

  router,

  store,

  render:h=>h(App)

}).$mount('#app');

### Router 路由文件编写

Router文件夹下的index.js路由文件主要完成了对前端页面的路由转发，其中也完成了在home等页面中对localstorage中token的验证检测。

import Vue from 'vue'

import Router from 'vue-router'

import Login from '@/components/Login'

import Home from '@/components/Home'

import Register from '@/components/Register'

import store from '../store/store'

import \* as types from '../store/types'

import ElementUi from 'element-ui'

import 'element-ui/lib/theme-chalk/index.css'

Vue.use(ElementUi)

Vue.use(Router)

const

  routes= [

    {

      path: '/',

      name: 'Login',

      component: Login

    },

    {

      path:'/register',

      name:"Register",

      component:Register

    },

    {

      path:'/home',

      name:'Home',

      component:Home,

      meta: {

        requireAuth: true,

    },

    }

  ]

// 页面刷新时，重新赋值token

if (window.localStorage.getItem('token')) {

  store.commit(types.LOGIN, window.localStorage.getItem('token'))

}

const router = new Router({

  routes

});

router.beforeEach((to, from, next) => {

  if (to.matched.some(r => r.meta.requireAuth)) {

      if (store.state.token) {

          next();

      }

      else {

          next({

              path: '/login',

              query: {redirect: to.fullPath}

          })

      }

  }

  else {

      next();

  }

})

export default router;

### store 文件（vuex部分）

引用官网中的话来说：

每一个 Vuex 应用的核心就是 store（仓库）。“store”基本上就是一个容器，它包含着你的应用中大部分的状态 (state)。Vuex 和单纯的全局对象有以下两点不同

Vuex 的状态存储是响应式的。当 Vue 组件从 store 中读取状态的时候，若 store 中的状态发生变化，那么相应的组件也会相应地得到高效更新。

针对本项目中使用到的状态，主要是账号密码，用户token，文件代码如下

import Vuex from 'vuex'

import Vue from 'vue'

import \* as types from './types'

Vue.use(Vuex);

export default new Vuex.Store({

    state: {

        id: null,

        password:null,

        token: null,

        title: ''

    },

    mutations: {

        [types.LOGIN]: (state, data) => {

            localStorage.token = data.id+data.password;

            state.token=data.id+data.password;

            //后端同学入土，目前没实现token 下次一定

            state.id = data.id;

            state.password=data.password;

        },

        [types.LOGOUT]: (state) => {

            localStorage.removeItem('token');

            state.token = null;

            state.id=null;

            state.password=null;

        },

        [types.TITLE]: (state, data) => {

            state.title = data;

        },

        [types.TOKEN]: (state, data) => {

            state.Token = data;

        }

    }

})

### axios封装

在前端请求中需要对后端的域名，端口等进行限制以及设置对应的拦截器，一旦报错，不至于前端界面出现卡顿情况

代码如下：

import axios from 'axios'

import store from '.././store/store'

import \* as types from '.././store/types'

import router from '.././router'

// axios 配置

axios.defaults.timeout = 5000

axios.defaults.baseURL = 'http://localhost:8080'

// http request 拦截器

axios.interceptors.request.use(

  config => {

    if (store.state.token) {

      config.headers.Authorization = `token ${store.state.token}`

    }

    return config

  },

  err => {

    return Promise.reject(err)

  },

)

// http response 拦截器

axios.interceptors.response.use(

  response => {

    return response

  },

  error => {

    if (error.response) {

      switch (error.response.status) {

        case 401:

          // 401 清除token信息并跳转到登录页面

          store.commit(types.LOGOUT)

          // 只有在当前路由不是登录页面才跳转

          router.currentRoute.path !== 'login' &&

            router.replace({

              path: 'login',

              query: { redirect: router.currentRoute.path },

            })

      }

    }

    // console.log(JSON.stringify(error));//console : Error: Request failed with status code 402

    return Promise.reject(error.response.data)

  },

)

export default axios

### 组件编写

组件编写由于能力和时间原因主要只编写了登陆和教室的查看界面，下面为login登陆界面编写

<template>

  <div >

    用户名:<input type="text" v-model="id" placeholder="请输入账号" />

    <br><br>

    密码:<input type="password" v-model="password" placeholder="请输入密码" />

    <br><br>

    <button @click="login">登录</button>

    <br><br>

    <router-link to='/Register'>

      <span style="text-align:center;font-size: 14px;">账号注册</span>

    </router-link>

    <router-link to='/ChangePwd'>

      <span style="text-align:center;font-size: 14px; color:red;">修改密码</span>

    </router-link>

  </div>

</template>

<script  type="application/ecmascript">

  import \* as types from '../store/types'

  import api from '../api'

  export default {

    name:"login",

    data() {

      return {

        id: "",

        password: "",

      }

    },

    methods:{

       login(){

           let data1={id:this.id,password:this.password}

           this.$store.commit(types.LOGIN, data1);

            this.axios.get(api.login+"?id="+this.id+"&password="+this.password)

            .then(response => {

            if (response.data.code === 102)

            {

            this.$router.push({path: '/Home'})

            }

          else {

              console.log("login fail")

          }

        })

    }

 }

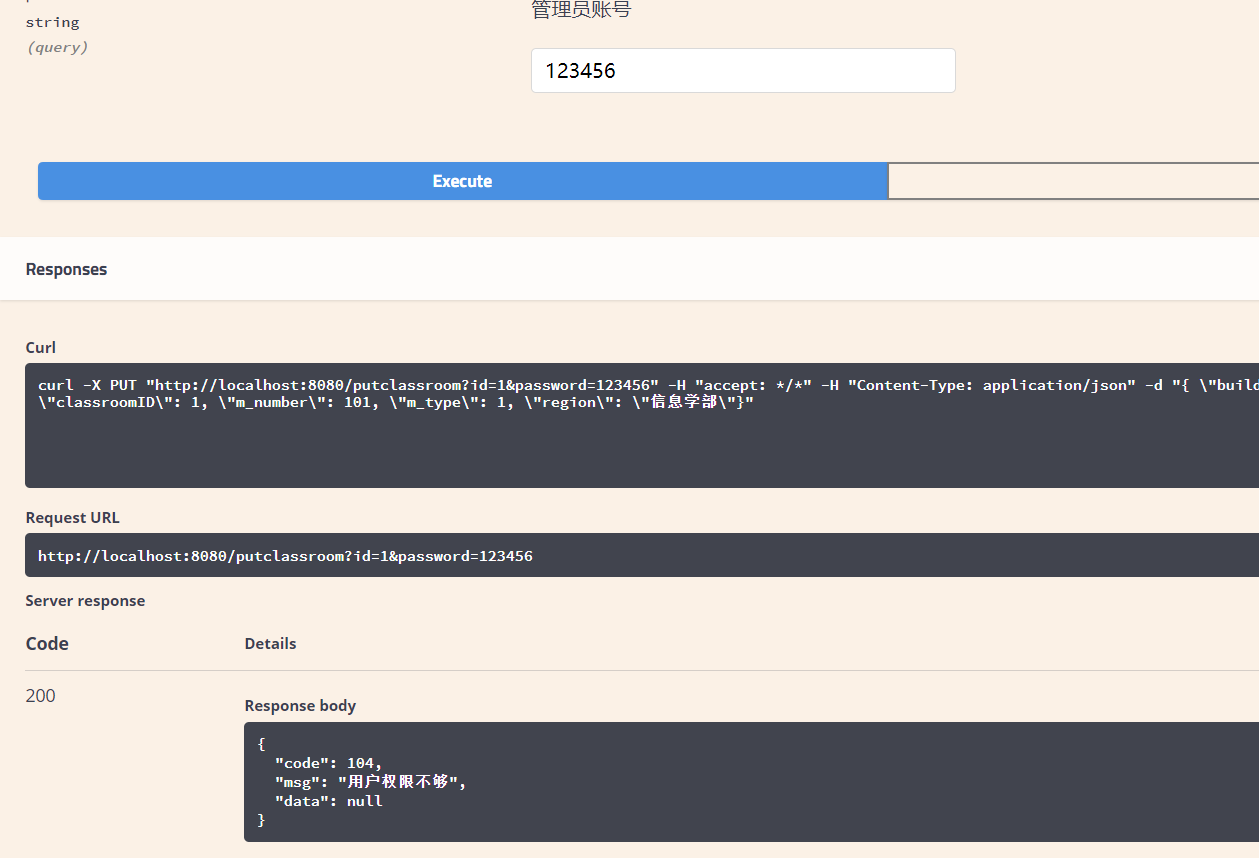
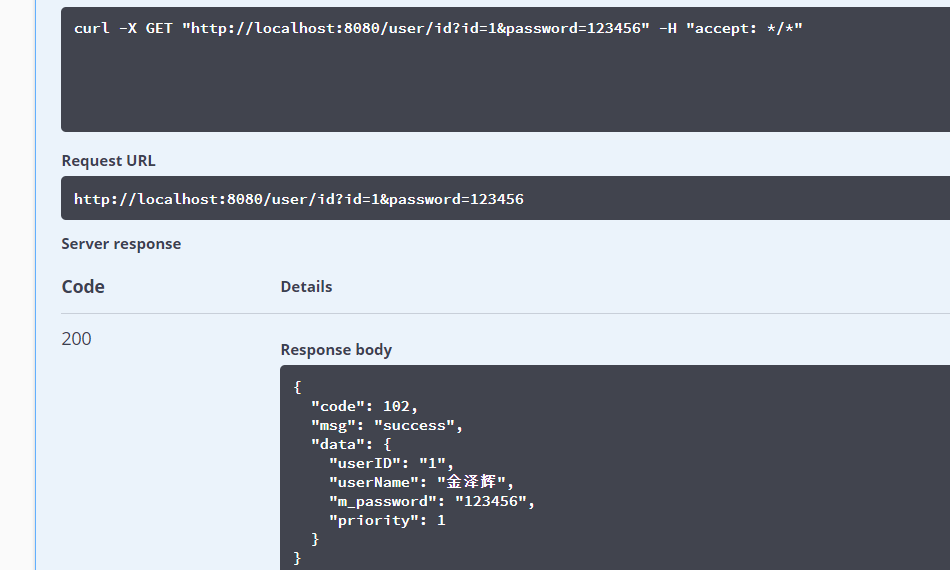
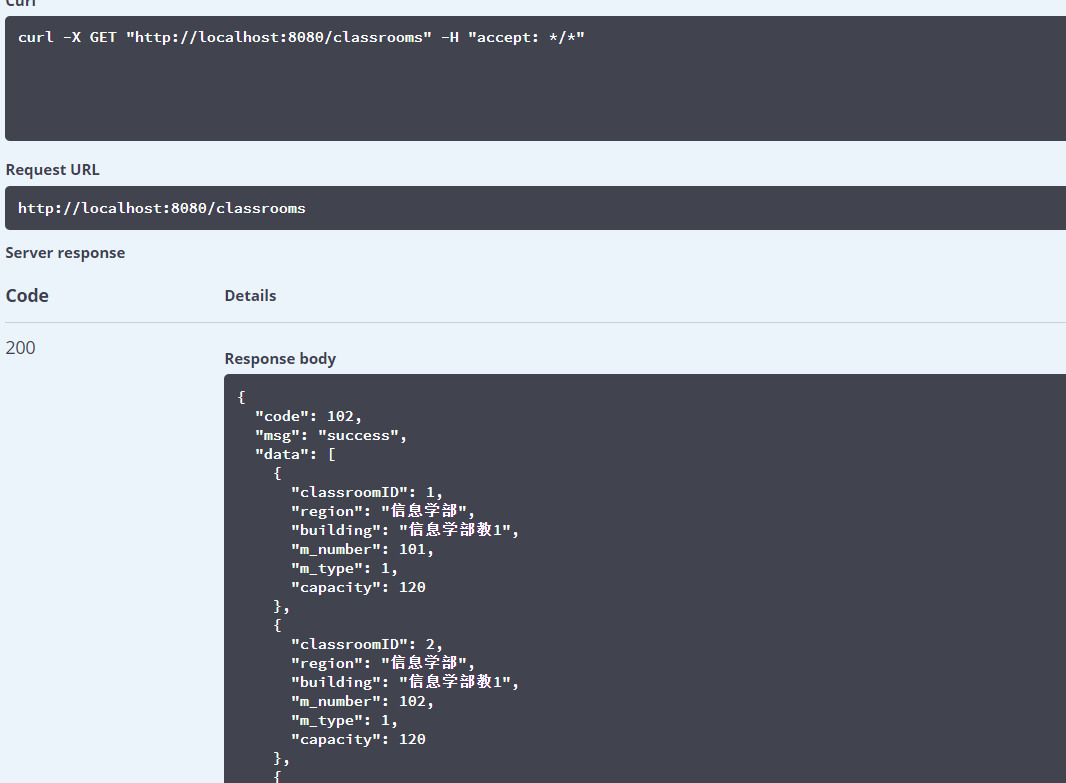
}

</script>

# 3 实验结果

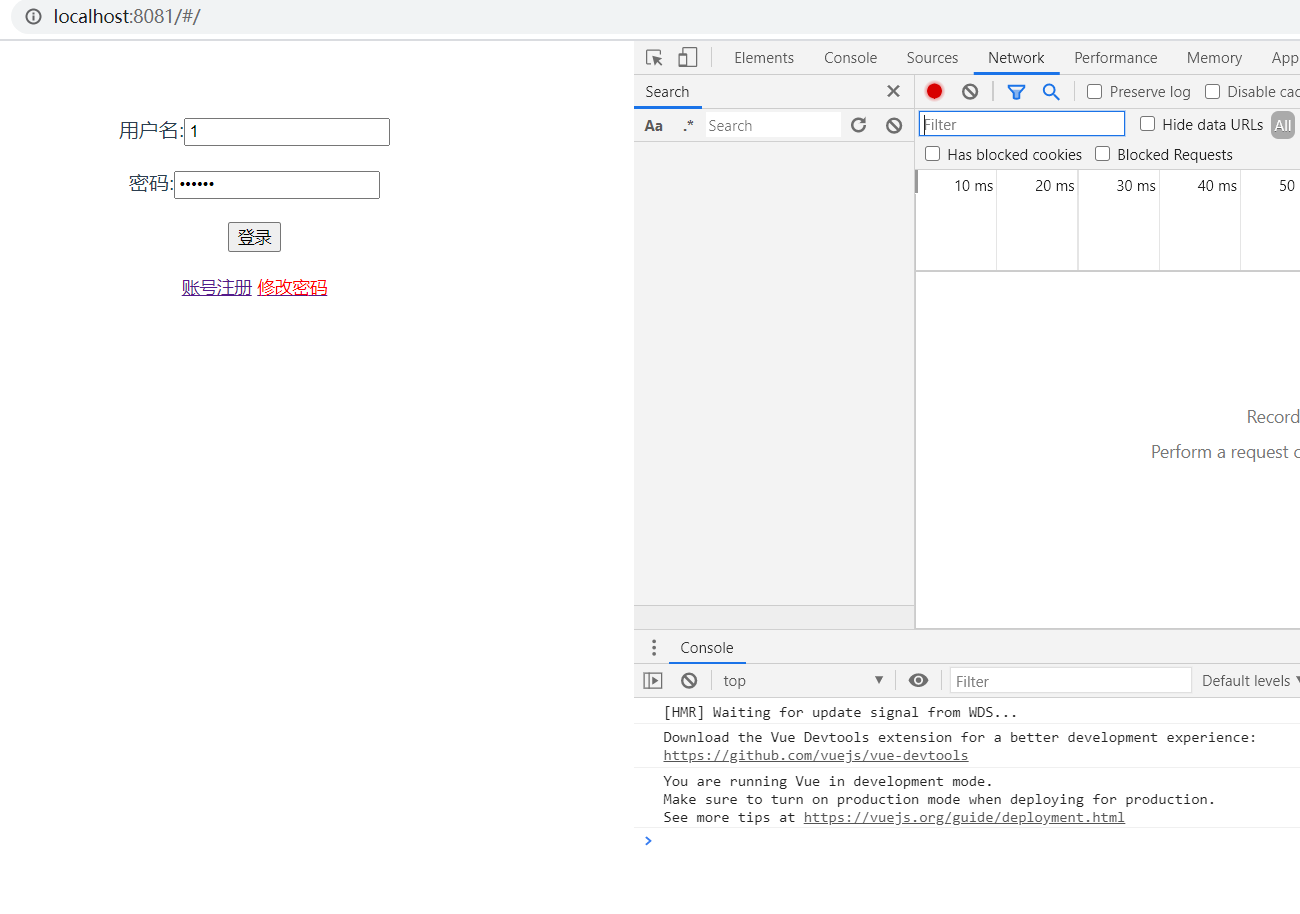
## 3.1 后端Restful api文档展示

在运行本项目的后端springboot 项目之后，进入[http://localhost:8080/swagger-ui.html#/](http://localhost:8080/swagger-ui.html" \l "/)界面，即可进入项目的swagger文档部分，结果呈现如下：

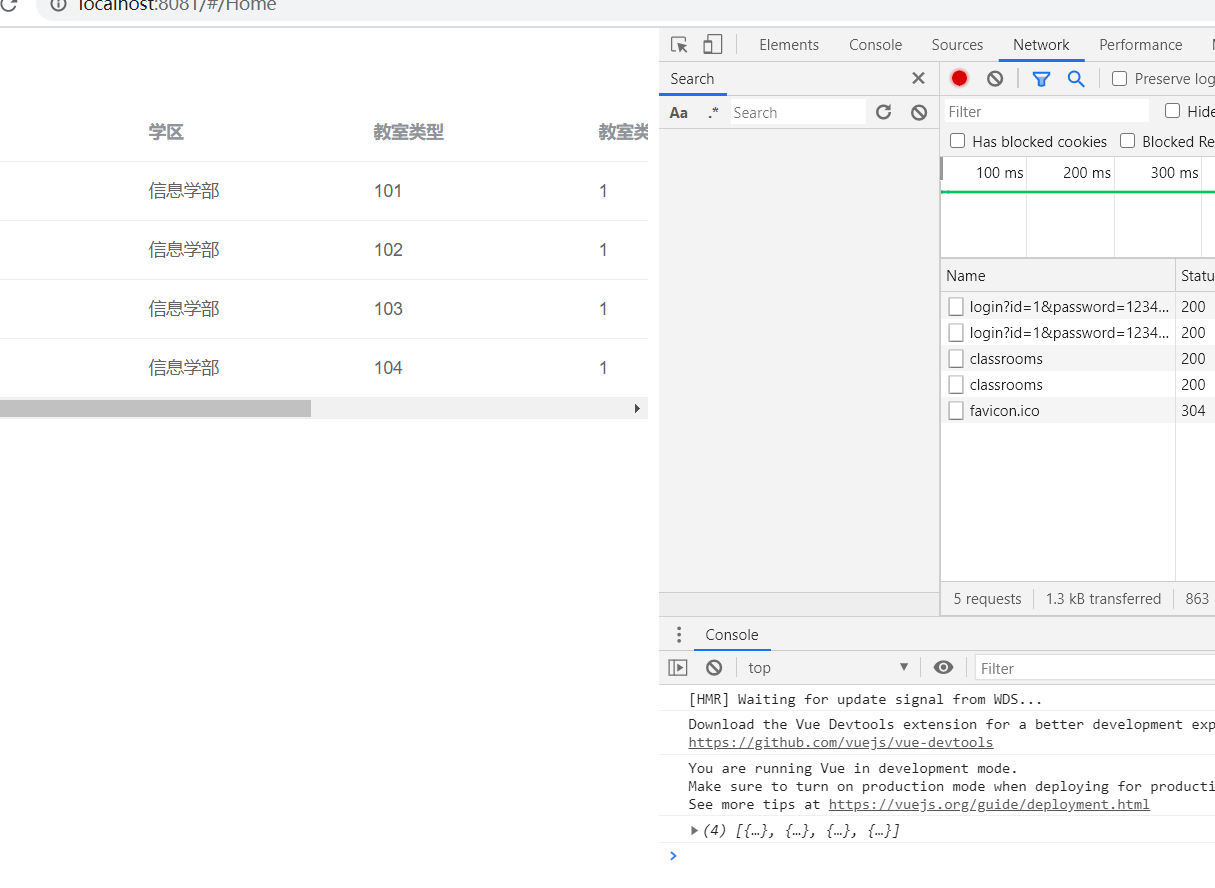


## 3.2 前端结果展示

登陆界面：



以及登陆完成后对于用户在对应界面中的教室查询展示



# 4成员分工与贡献

|  |  |  |
| --- | --- | --- |
| **姓名** | **贡献分** | **贡献** |
| 金泽辉 | 20 | 数据库合作设计，用例图主体设计， 流程图主体设计，  前端vue.js框架的运用300行，后端部分controller编写，项目部署，参与后端api的调试与补充。项目整体文档撰写与修改。 |
| 朱家伟 | 20 | 数据库合作设计，用例图辅助设计，  流程图辅助设计。  后端springboot架构主体代码编写800行，api文档注释编写，后端测试文件编写。后端以及项目部分文档撰写。 |

总体来说，这次对于我和朱家伟同学都是一次很不错的尝试，起初我们都是想使用java ee课堂中学习到的spring mvc来进行一个简单mvc架构的尝试，后来随着javaee课程深度提升，我们就转换成为springboot mybatis +vue element-ui项目的编写。由于我比较熟悉后端操作，所以我这次去尝试了前端知识，朱家伟同学则熟悉了后端api设计的整体流程。这次实验，我们从数据库的编写到api接口设计，再到前后端分离，再到跨域访问请求，本地存储用户信息等都是一些不属于自身熟悉领域的尝试，感觉收获挺大。美中不足的就是，由于我个人在前端vue框架的使用方面着实不够了解，未能完整地展现我们本次的前后端交互功能和系统设计，希望在这学期后的java ee中，可以合作出更完整与系统的项目。