会议管理系统

黄灿铭[1]
(中山大学计算机科学与数据学院)[1]

1 选题背景

在当今互联网+的浪潮中,传统行业与互联网的融合能够提高传统行业的工作效率,节省其运营成本,使其焕发出新的活力。对于公司来说,业务量的增加会导致的会议的增加,采用传统的人工会议管理会带来许多问题。如在会议之前,传统的会议管理存在着预约流程冗长,不能及时地通知和修改会议时间困难等问题,在会议期间,则存在着不能够对会议及时地控制等问题,在会议结束之后,则存在着不能全面的对会议情况进行总结和分析等问题,因此需要开发一个会议管理系统来解决这些问题引起的公司成本上升这一问题。

2 软件工程过程

2.1 需求分析

由于开发时间较短，因此本系统主要实现了三个功能——登录注册，添加会议和查询会议。项目使用 vue-cli + element ui 作为前端框架，koa 作为后端框架编写。

1. 登录注册：

用户可以根据自己的姓名，部门注册一个账号并登录，登录的用户可以查询和添加会议。已注册的用户会出现在别的用户的会议可选参与人列表中。

2. 添加会议：

用户可以自定义会议主题，参与人员，会议时间，来发起一个会议。参与人员包括不同部门的人员，会议时间精确到分钟（年月日小时分钟）

3. 查询会议：

已创建的会议会出现在会议查询的页面中，查询内容包括会议主题，会议发起人，会议参与人，会议时间。用户还可以通过指定会议主题和时间范围查询指定的会议。

2.2 设计

1 登录注册页面

UI：

使用 element ui 进行布局，登录注册的表单填写部分均使用 el-form。

在登录页面点击注册后可跳转到注册页面，

点击登录，若登录成功则跳转到会议管理系统的主页面，登录失败则弹出提示消息。

注册页面的表单填写，用户名要求是 2-4 个字，密码无限制，但需重复输入以确认密码，部门可在管理部，生产部，业务部，工程部中任选其一，此四项均为必填项。点击重置可以清空输入的用户名和密码信息，点击注册后，若注册成功则跳转到登录页面，注册失败则弹出提示消息。

2 查询页面

UI:

用户成功登陆后，会进入主页面，主页面默认是查询页面，左边导航栏有查询会议，添加会议，退出登陆三个功能可选项。导航栏由 el-menu 实现。

查询页面分为两个部分，一部分是用户查询条件，可以通过指定会议名称和会议时间来更新会议表格，这部分由 el-form 和 el-date-picker，el-button 实现。第二部分是会议表格，默认状态下为列出所有会议，一个页面列出五条会议，列出的一条会议包括五项，会议时间，主题，主持人，参与人。该部分由 el-table 和 el-pagination 实现。

3 添加会议页面

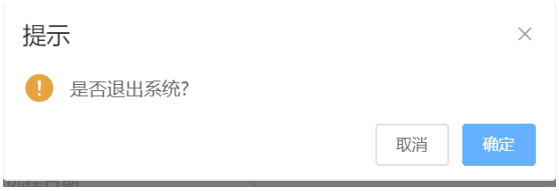
UI:

点击导航栏的添加会议可以跳转到添加会议页面。

添加会议页面需要填写会议主题，选择参与人员，会议日期，会议时间。参与人员分为四个部门，每个部门对应一个多选下拉框，可以选择该部门的人员，会议时间选择区域为 8:30-22:30，间隔为十五分钟。该部分由 el-form，el-select，el-date-picker，el-time-select 实现。添加会议成功后，会弹出提示消息，之后便可在查询页面中查看。

退出登陆页面：

UI:



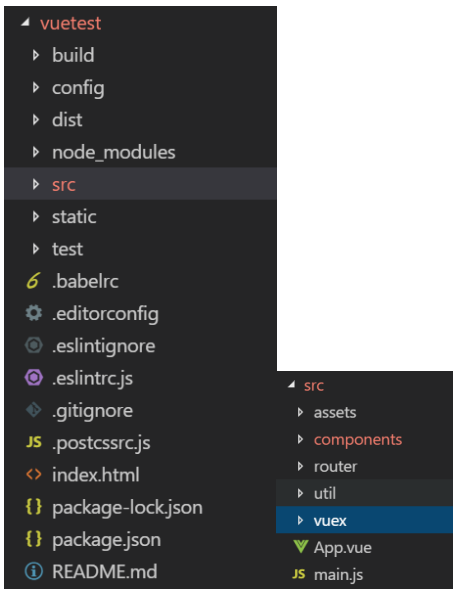
点击导航栏的退出会弹出退出提示框，点击确定即返回登录界面

2.3 实现

本系统采用 vue-cli+element ui 作为前端框架，采用 koa 作为后端框架进行开发，以下简要说明登陆注册和添加会议，查询会议功能的原理。

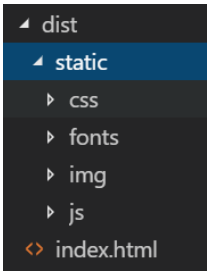
1. 使用 vue-cli 快速原型开发

首先使用 npm 全局安装 vue-cli，接着进入项目空间，使用命令创建项目，vue init webpack vuetest。接着 vue-cli 会自动生成一个项目，此为项目目录。



我们只需要在 src 中编写我们的前端页面，即组件和路由以及某些配置。

在项目完成后，执行 npm run dev 可将项目打包，生成 dist 文件夹，这就是我们打包好的前端页面，点击文件夹中的 index.html 即可打开。



现在我们就只剩下简单的 index.html 和对应的图片，css 和 js 文件。

2. 前后端交互

在页面填写完表单信息或者点击相应的按钮后，前端页面的 8080 端口会向后端的 3000 端口发送 get/post 请求（request），post 请求的 body 字段中会包含表单中的信息，后端接受到 get/post 请求后，根据请求中包含的 url，匹配相应的路由，并调用相应的匹配函数，进行相应的处理后后端给前端发回响应（response）。

登录时对应的路由是 POST /login

注册时对应的路由是 POST /register

查询会议时对应的路由是 POST /getTableData

创建会议时对应的路由是 GET /getdept 和 POST /createMeeting 前者是获取各部门的人员，后者是创建会议

```
module.exports = {
    'GET /api/': async (ctx, next) => {…
    },

    'POST /register': async (ctx, next) => {…
    },

    'POST /login': async (ctx, next) => {…
    },


    'GET /getdept': async (ctx, next) => {…
    },

    'POST /createMeeting': async (ctx, next) => {…
    },

    'POST /getTableData': async (ctx, next) => {…
    }
};
```

3. 状态存储

在确认登录之后，前端会把登录的用户信息储存在前端页面中，以维持登录状态。该功能通过使用 vuex 实现。

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。

```
const state = {
  user: {username: null, password: null, department: null},
}

const mutations = {

  SETUSER(state, user) {
    state.user = user;
  },

  DELETEUSER(state) {
    state.user = {username: null, password: null, department: null}
  },

}
```

3 感想

本次是第一次独立开发一个完整的 web 应用，尽管由于开发时间短，功能十分不健全，单是查资料和前端的 ui 设计和构建就花了很多时间，导致后端没时间实现更多的 api 供前端使用，而且没有使用数据库来进行数据持久性存储，总之还是有很多遗憾。但是整个应用的基本框架已经搭建好，只是需要更多的开发时间去完善一些功能，并不困难。最初开发会议管理系统的灵感是经常遇到部门，课程小组安排会议后，由于事情太多，忘了一些，而且也有系统地整理每一次会议内容的想法，虽然并没有开发出来，理论上需要多设计一个页面来显示每一次的会议记录。还有后端的很多验证和限制都没有完善，这都是边边角角的基础功能，但做起来十分花时间。总之学到了很多，也还需要学习更多。