Rational Software

校园拼客

Version 2.0

修订历史

日期	版本	描述	作者
16/7/2015	V1.0	初始版本	李健华
17/7/2015	V2.0	补充完善	孔方圆

小组成员

学号	姓名
12330144	姜岚
12330150	孔方圆
12330151	孔晓彤
12330153	邝媛媛
12330165	李健华

目录

技术选型

项目	Web app	备注
1 终端支持	PC	
1.1 开发语言框架	HTML5 CSS3 JavaScript	
2 服务器端支持		
2.1 语言	JavaScript	
2.2 web 框架	Meteor	
2.3 数据库	mongodb	

技术简介：

HTML5——万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言（HTML），HTML5 本身是由 W3C 推荐出来的，它的开发是通过谷歌、苹果，诺基亚、中国移动等几百家公司一起酝酿的技术，这个技术最大的好处在于它是一个公开的技术。它还有如下优点：多设备跨平台、自适应网页设计、即时更新、提高可用性和改进用户的友好体验。

JavaScript——JavaScript 是脚本语言，是一种轻量级的编程语言。JavaScript 是可插入 HTML 页面的编程代码。插入 HTML 页面后，可由所有的现代浏览器执行。

Meteor——Meteor 是一个新鲜出炉的现代网站开发平台，基础构架是 Node.JS + MongoDB，它把这个基础构架同时延伸到了浏览器端，如果 App 用纯 JavaScript 写成，JS APIs 和 DB APIs 就可以同时在服务器端和客户端无差异地调用，本地和远程数据通过 DDP（Distributed Data Protocol）协议传输。页面 CSS 样式和 HTML 结构更改时可自动刷新浏览器实现代码的热部署，方便查看运行效果；访客浏览网站，服务器端和每一个浏览器端的数据增删查改都将自动同步推送至服务器和每一个会话终端，不需要刷新页面来查看新内容，新版本代码和数据推送过程也不会打断当前用户的正常浏览

Mongodb——一种非关系型数据库。Mongo DB 很好的实现了面向对象的思想(OO 思想),在 Mongo DB 中 每一条记录都是一个 Document 对象。Mongo DB 最大的优势在于所有的数据持久操作都无需开发人员手动编写 SQL 语句,直接调用方法就可以轻松的实现 CRUD 操作。

技术选型理由：

随着互联网技术的发展，人们对网页浏览的需求日益增加，人们需要的是既能快速浏览， 又炫丽的页面，在这样的背景下，传统的 web 技术显然不满足现代人们的新需求，因此单页面网页应用技术就油然而生，然而又在百家争鸣当中，angularjs 和 meteorjs 这两种技术尤为突出，但是我们觉得 angularjs 的 url 里面的 #号有点不堪入目，所以就打算选 meteorjs 作为开发框架。meteorjs 有两方面的优点：

1.把后台的数据库按照一定的需求发布到前台， 虽然刚开始加载的时候会稍微慢一点点，但是一旦加载进来了，就能让前台使用本地的数据库一样，极其的快，能让用户完全忘记这是在通过网络浏览的页面，这就解决了用户们的第一个需求：快速浏览。

2.前台用 template 的形式模块化，变成 js 代码动态的与 html 交互，这使得各种大神用这个机制做出了极其华丽的前台页面，这就满足了用户的第二个需求：炫丽。

架构设计

软件架构图

表示层	用户界面
业务层	功能实现（具体功能见模块划分功能模块）
中间层	数据处理
数据库、操作系统	数据库：Mongodb；操作系统：Mac

模块划分

1. 程序结构图

客户端程序	用户界面					
功能模块	发起活动	取消活动	加入活动	退出活动	查看已发起的活动	查看已加入的活动
存储结构	用户信息			活动信息		

2. 目录结构说明

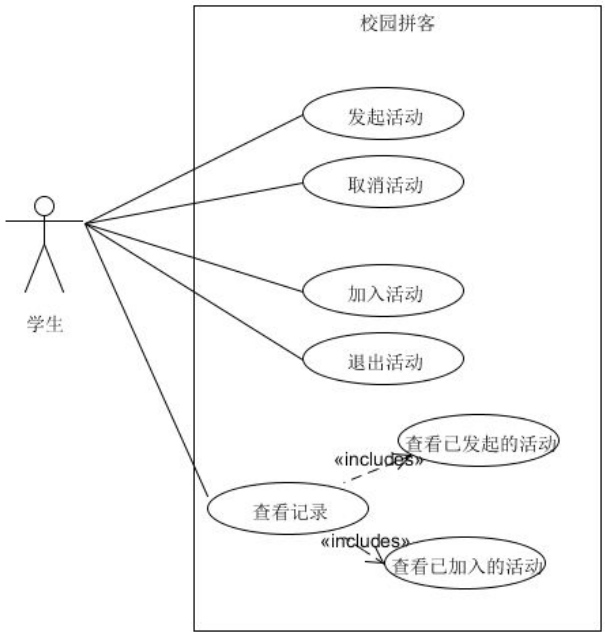
client　定义一些前台用到的组建： 如 html, css, javascript
　　helpers
　　　　onfig.js　定义用户注册时候需要输入的信息
　　　　errors.js　定义 error 的前台数据库模型
　　　　stylesheets　定义 css
　　templates　前台各种 templates 以及对应的 js 文件
　　　　activity　有关发布活动页面有关的所有 html 和 js
　　　　…
　　　　application
　　　　　　layout.html　整个框架组建
　　　　　　personalpage.html　使用 material design 的时候用到的已经全部注释
　　　　　　sponsorActivity.html　使用 material design 的时候用到的已经全部注释
　　　　events　使用 material design 的时候用到的已经全部注释
　　　　home　定义了主页面用到的 html 以及 js 文件
　　　　　　…
　　　　includes　一些功用的组建用到的 html 以及 js 文件
　　　　　　…
　　　　users　有关查询用户历史纪录的组建用到的 html 以及 js 文件
　　　　　　…
lib　定义一些前后台公用的组建： 例如 路径分配 router.js
　　collections
　　　　activities.js 定义了有关活动的前台 minimongo 和 后台 mongodb 各种调用方法
　　permissions.js　判断一个活动是否是这个人物发起的
　　router.js　定义各种路径以及加载更多的方法
public　放一些图片之类的
　　resources
　　　　ali.jpg　只是一个图片
server　定义了后台 publification 的方法还有一些调试时候用到的初始数据库信息
　　fixtures.js　刚开始调试时候用到的初始数据库数据
　　publification.js　定义了把哪些后台数据发布到前台

软件设计技术

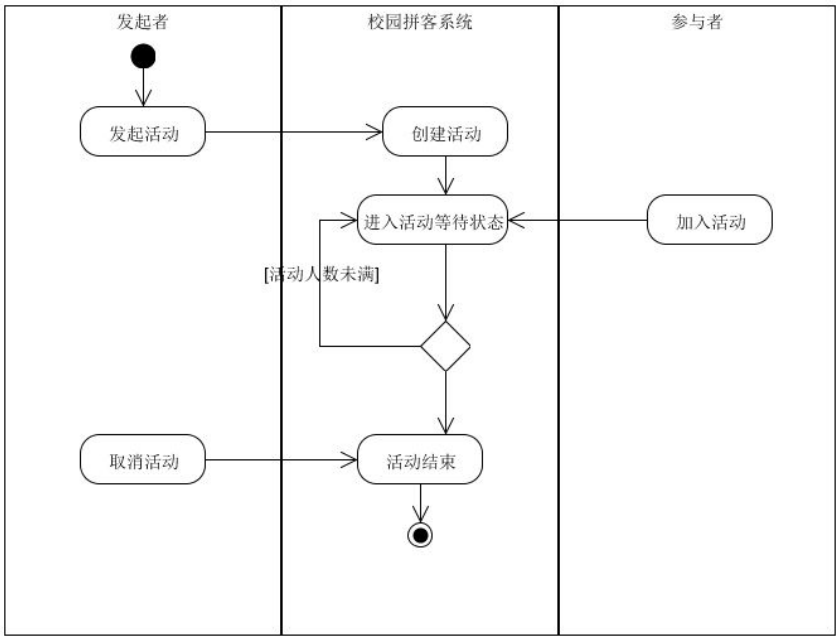
1.OOAD

该项目在设计时使用的面向对象的分析设计方法（Object-Oriented Programming），设计中用到的部分 UML 图如下：（更多详细内容见需求文档）

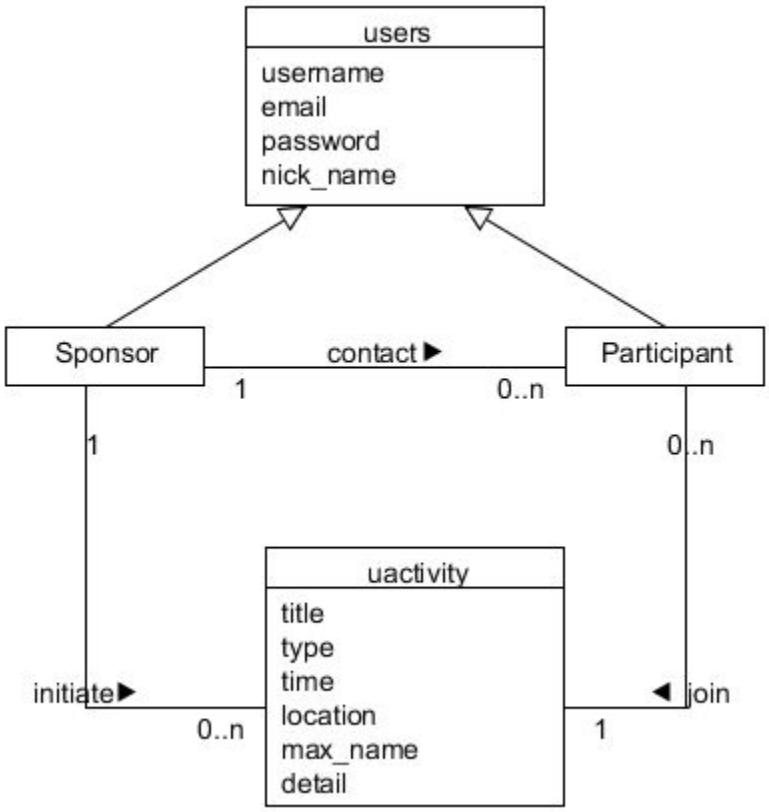
系统用例图



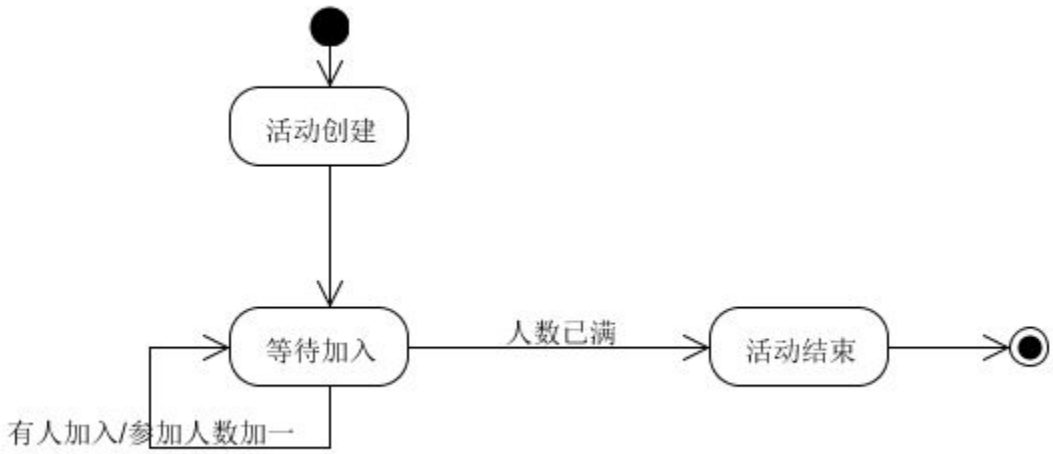
活动图



领域模型


users
username
email
password
nick_name
Sponsor
contact
Participant
1
0..n
1
0..n
uactivity
title
type
time
location
max_name
detail
initiate
0..n
join
1

活动状态图


活动创建
等待加入
人数已满
活动结束
有人加入/参加人数加一

2. Service Oriented Architecture

一个用户可以作为发起者发起活动，也可以作为参与者加入活动。

1）对于发起者身份，提供发起活动、取消活动、查看已发起的活动的服务；

发起活动——activity_create.js

```
activityInsert: function(activityAttributes) {
    check(Meteor.userId(), String);
    check(activityAttributes, {
        title: String,
        type: String,
        time: String,
        location: String,
        max_number: Number,
        contact_way: String,
        detail: String
    });

    var errors = validateActivity(activityAttributes);

    if (errors.title) {...}

    if (errors.type) {...}

    if (errors.time) {...}

    if (errors.location) {...}

    if (errors.max_number) {...}

    if (errors.contact_way) {...}

    if (errors.detail) {...}

    var user = Meteor.user();
    var activity = _.extend(activityAttributes, {
        sponsor: Meteor.userId(),
        sponsorNickName: user.profile.nickname,
        current_number: 0,
        participator: []
    });

    var activityId = Activity.insert(activity);

    return {
        _id : activityId
    };
},
```

取消活动——activity_create.js

```
deleteActivity: function(activityId) {
    check(this.userId, String);
    check(activityId, String);

    var activity = Activity.findOne(activityId);
    if (!activity) {
        throw new Meteor.Error('invalid', "Activity not found");
    }

    if (activity.sponsor != this.userId) {
        throw new Meteor.Error('invalid', "You are not the sponsor of this activity");
    }

    var removed = Activity.remove(activityId);

    if (!removed) {
        throw new Meteor.Error('invalid', "You weren't able to delete this Activity");
    }

}
```

查看已发起的活动——router.js

```
SponsoredController = RouteController.extend({
    template: 'sponsored',
    increment: 4,
    activitiesLimit: function() {
        return parseInt(this.params.sponsoredActivityCounter) || this.increment;
    },
    findOptions: function() {
        return {sort: {time: -1}, limit: this.activitiesLimit()};
    },
    activitiesSub: function() {
        return Meteor.subscribe('activity', this.findOptions());
    },
    activities: function() {
        var userId = Meteor.userId();
        return Activity.find({sponsor: userId}, this.findOptions());
    },
    data: function() {
        var hasMore = this.activities().count() === this.activitiesLimit();
        var nextPath = this.route.path({sponsoredActivityCounter: this.activitiesLimit() + this.increment});
        return {
            activities: this.activities(),
            ready: this.activitiesSub().ready(),
            nextPath: hasMore ? nextPath : null
        }
    }
});

Router.route('/user/log/sponsored/:sponsoredActivityCounter?', {
    name: 'sponsored'
});
```

2）对于参与者身份，提供加入活动、退出活动、查看已加入的活动的服务；

加入活动——activity_create.js

```
  joinActivity: function(activityId) {
      check(activityId, String);
      check(this.userId, String);

      var activity = Activity.findOne(activityId);
      if (!activity) {
          throw new Meteor.Error('invalid', "Activity not found");
      }

      if (_.include(activity.participator, this.userId)) {
          throw new Meteor.Error('invalid', "Already participated");
      }

      if (activity.current_number >= activity.max_number) {
          throw new Meteor.Error('invalid', "人数已满!");
      }

      if (activity.sponsor == this.userId) {
          throw new Meteor.Error('invalid', "You are the sponsor of this activity");
      }

      var affected = Activity.update({
          _id: activityId,
          participator: {$ne: this.userId}
      }, {
          $addToSet: {participator: this.userId},
          $inc: {current_number: 1}
      });

      if (! affected) {
          throw new Meteor.Error('invalid', "You weren't able to join this Activity")
      }

  },
```

退出活动——activity_create.js

```
quitActivity: function(activityId) {
    check(activityId, String);
    check(this.userId, String);

    var activity = Activity.findOne(activityId);
    if (!activity) {
        throw new Meteor.Error('invalid', "Activity not found");
    }

    if (!_.include(activity.participator, this.userId)) {
        throw new Meteor.Error('invalid', "Not participated");
    }

    if (activity.sponsor == this.userId) {
        throw new Meteor.Error('invalid', "You are the sponsor of this activity");
    }

    var affected = Activity.update({
        _id: activityId,
        participator: {$in: [this.userId]}
    }, {
        $pull: {participator: this.userId},
        $inc: {current_number: -1}
    });

    if (! affected) {
        throw new Meteor.Error('invalid', "You weren't able to quit this Activity")
    }
},
```

查看已加入的活动——router.js

```
JoinedController = RouteController.extend({
    template: 'joined',
    increment: 4,
    activitiesLimit: function() {
        return parseInt(this.params.joinedActivityCounter) || this.increment;
    },
    findOptions: function() {
        return {sort: {time: -1}, limit: this.activitiesLimit()};
    },
    activitiesSub: function() {
        return Meteor.subscribe('activity', this.findOptions());
    },
    activities: function() {
        var userId = Meteor.userId();
        return Activity.find({participator: {$in: [userId]}}, this.findOptions());
    },
    data: function() {
        var hasMore = this.activities().count() === this.activitiesLimit();
        var nextPath = this.route.path({joinedActivityCounter: this.activitiesLimit() + this.increment});
        return {
            activities: this.activities(),
            ready: this.activitiesSub().ready(),
            nextPath: hasMore ? nextPath : null
        }
    }
});

Router.route('/user/log/joined/:joinedActivityCounter?', {
    name: 'joined'
});
```