软件设计文档

技术选型

Web App

1. 通用且跨平台。Web App 基于 html+css+javascript，能够运行在所有装有现代浏览器的设备上，且对于不同的设备，适配难度较低，工作量小，开发成本低。

2. 升级简单，可快速迭代。Web App 的升级只需要更新服务端代码，用户无需进行任何操作，也不会有太大的感知。对于一些小错误，可以快速修复并更新。

3. 用户体验好。近年来，智能手机快速发展，通讯技术的提升也非常明显，几乎每个用户人手一台高性能的手机并连接着告诉的 4G 或 WiFi 网络，这样的环境给 Web App 的发展打下了良好的基础。在这样的环境下，Web App 可以提供接近甚至打平原生应用的体验。在 PWA 出现后，Web App 和原生应用之间的界线进一步模糊，体验更进一步。

Node.js

1. 统一前后端开发语言，降低开发难度。

2. 解决高并发，特别适用于前后端分离，后端用于提供 API 获取数据的情况。

3. 社区完善，支持全面，有大量成熟稳定、功能强大的第三方包，提高开发效率，降低开发难度，同时遇到问题也更容易找到解决方案。

Vue.js

1. 虚拟 DOM，减少重复渲染，降低资源开销。

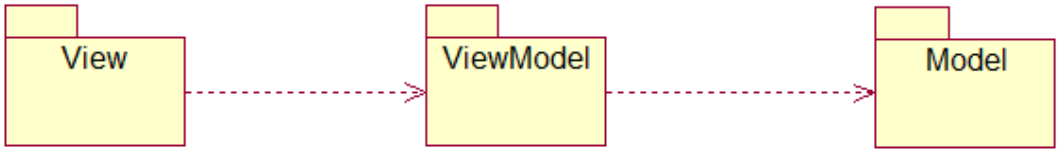
2. 数据双向绑定，视图层与视图模型层中任意一者发生改变，另一者会随之改变，简化开发过程中对于逻辑的处理。

3. 组件化、低耦合，可以将具有一定功能的模块封装成组件，再由组件拼凑起来，构成具有不同功能的页面。这种特性使得维护简单，且代码可高度重用。

架构设计

架构描述

系统采用 MVVM（Model-View-ViewModel）三层架构，该架构的使用实现了应用程序的分层管理，简化后续对程序的修改和扩展，并且使程序某一部分的重复利用成为可能。模型层（Model）根据需求从数据库中获取所需要的数据；视图层（View）负责显示用户界面及相关数据并对用户输入进行反馈；视图模型层(ViewModel)是模型层与视图层的连接层，从模型层获取数据后形成视图层所需要的数据结构与视图层双向绑定。当视图层/视图模型层数据发生改变，视图模型层/视图层也会同步改变，当数据改变后，视图模型层则操作模型层对数据库进行更改。



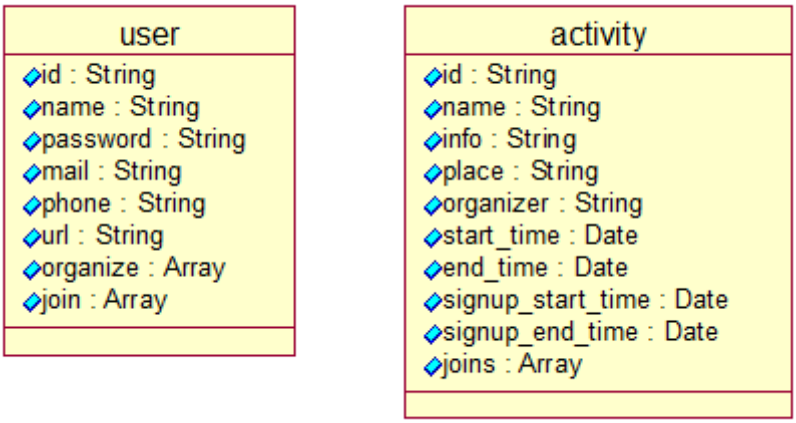
系统架构图

关键抽象

经过分析，本系统有两个实体类，分别为 user、activity。

user：储存用户基本信息，包括用户名、密码(加密)、邮箱、头像等个人账号信息。

activitiy：储存活动相关信息，包括标题、时间、地点、人数等。

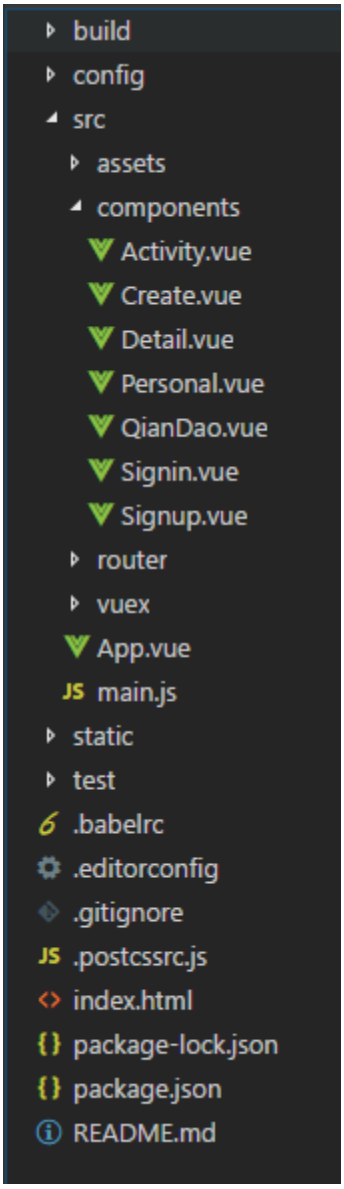


实体类

模块划分

前端模块

代码结构：



前端模块划分主要是根据网站页面进行划分，主要分为以下模块：

- 登录注册
- 个人主页
- 活动管理
- 活动详情
- 活动创建
- 扫码签到

后端模块

主要逻辑集中于前端，后端仅用于接收前端请求并对数据库进行增删查改，且数据类较少，因此后端并未根据不同的数据类或不同的处理逻辑进行模块划分。

软件设计技术

结构化编程

在项目的设计技术选择中，主体使用了结构化编程方式。采用按功能来分析系统需求，自顶向下，逐步求精，模块化。项目小组首先采用结构化分析(SA)方法对系统进行需求分析，然后采用结构化设计(SD)方法对系统进行概要设计、详细设计，最后采用结构化编程(SP)方法实现系统。在项目的实现过程中通过采用子程序、程式码区块、for 循环以及 while 循环等结构来取代传统的 goto，让程序更易于理解。

在项目的编程实践过程中，主要以函数的形式封装各个小功能程序，对外提供一个统一的入口。例如后台的程序代码模块:

```
//是否参加
app.get("/isSignup", (req, res) => {
    var aid = req.query.aid;
    var uid = req.query.uid;
    activityCollction.findOne({id:aid, "joins.id": uid}, function(err, data) {
        if(err) {res.send({"ok":false, "code":404, "err":err});}
        else if(data === null) {
            res.send({"ok": true, "code":400, "data":"Not_signup"});
        }
        else {
            res.send({"ok": true, "code":401, "data":"signup"});
        }
    })
})
```

面向对象编程

面向对象的程序设计具有易维护，质量高，效率高，易扩展等优点。在项目代码的实现中，结合了面向对象的思想。对具有多个属性的一个整体封装成一个对象。例如：在用户登录注册时，会将用户的信息（用户名，密码，邮箱，学号，手机号码等）封装成一个整体进行传输。

```
    var doc = {
        id: req.query.id,
        name: req.query.name,
        password: req.query.password,
        mail: req.query.mail,
        phone: req.query.phone,
        url: req.query.url,
        organize:[],
        join:[]
    }
```

使用 Web API 提供服务

在提供后台接口服务时，使用的是通过 http 公开的 Web API，并使用 json 作为数据格式。

```
app.get("/test", (req, res) => {
    activityCollction.findOne({id:100}, function(err, data) {
        var a = [];
        for(var i = 0; i < data.array.length; i++) {
            console.log(JSON.stringify(data.array[i]));
            a.push(data.array[i]["id"]);
        }
        res.send(a);
    })
});
```

数据库支持

使用 MongoDB 来提供后台数据库支持，保证了后台数据的长期存储。支持非结构化数据的存储，很大程度上提高了数据库访问的性能。

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27000";
```

设计模式

在程序的设计过程中，也使用到了一些常见的设计模式，例如单一职责原则。保证作为一个整体的部分只提供唯一的一个功能，有利于使得程序保持松耦合。