高效

以下简单的展示一些登录用户后能够查看的一些数据画面。

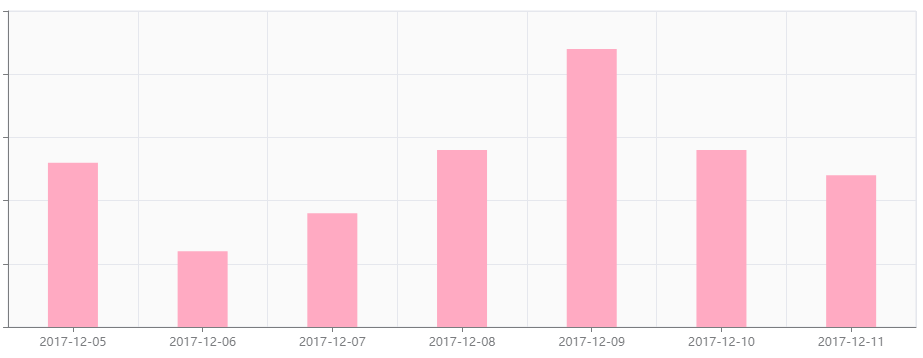


图 1-1-3-2 在线直方图每日盖章统计

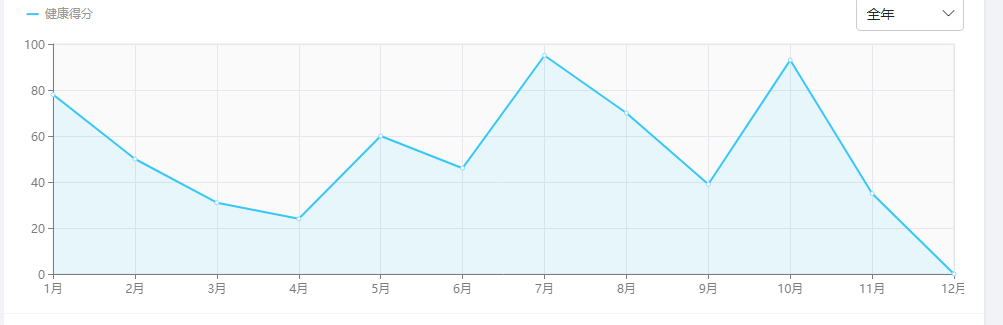


图 1-1-3-3 网页安全性评估

根据反馈的数据，用户可以简单且及时地查看单日盖章数目，一旦发生异常-----盖章数目大于预估盖章数目,用户可及时查看系统日志,进行错误查找,防止一些问题由于早期显现的不明显而被忽略。

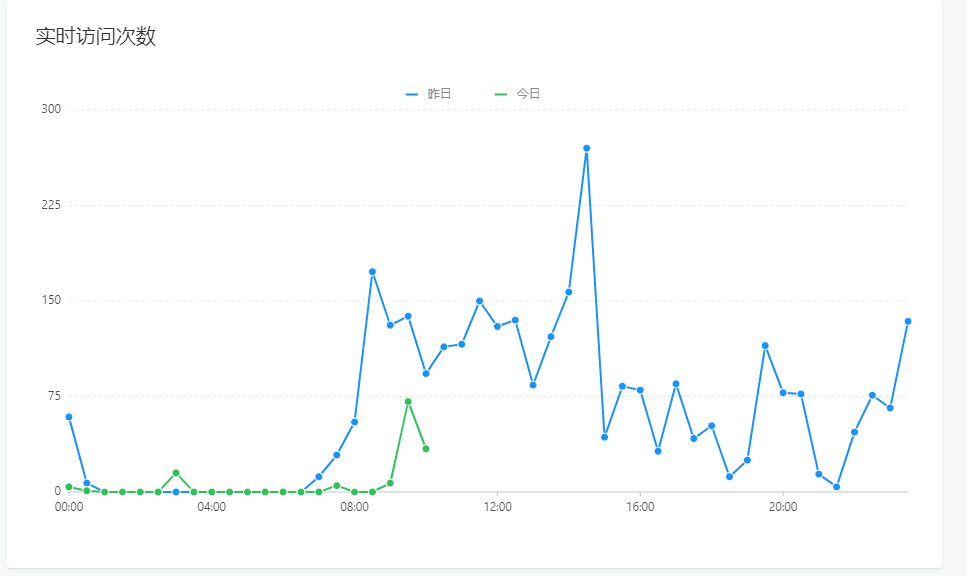


图1-1-3-4 实时访问人数

### **2.1前后端分离技术**

**前后端分离**

本项目开发使用了前后端分离技术，即前端专注于数据展示与交互，后端专注业务逻辑的处理。在架构上分离解耦，逐渐摆脱前后端在架构上的依赖，前后端单独开发，通过 Restful接口传递数据。减轻后端服务器的压力，后端服务器不再负责页面渲染，只负责输入数据，提升吞吐量。

**2.1.1 Git团队开发**

**git项目管理**

Git是一个开源的分布式版本控制系统，用以有效、高速的处理从很小到非常大的项目版本管理。git对源代码进行管理,使本地机器与远程服务器代码保持同步,让多人进行项目开发,对同一份代码进行更改,也可根据项目需求同步他人代码。当项目遇到未预期错误,使用git版本管理可以清晰了解不同版本之间区别,对代码进行回滚处理。



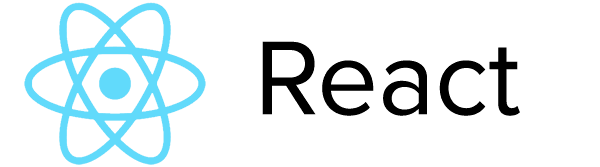


**2.4网页技术介绍:**

为了能够更直观的展示后台数据，进行数据的批量处理，提高效率，我们开发了后台管理系统，对各个系统模块进行统一管理。

**技术实现：**

**React**



后台管理系统采用了React框架，提高开发效率。在Web开发中，我们总需要将变化的数据实时反应到UI上，这时就需要对DOM进行操作。而复杂或频繁的DOM操作通常是性能瓶颈产生的原因。React为此引入了虚拟DOM（Virtual DOM）的机制，每当数据变化时，React都会重新构建整个DOM树，然后React将当前整个DOM树和上一次的DOM树进行对比，然后仅仅将需要变化的部分进行实际的浏览器DOM更新。不同于Jquery等框架的直接操作DOM，性能有了很大提升。

虚拟DOM(virtual-dom)不仅带来了简单的UI开发逻辑，同时也带来了组件化开发的思想， UI功能模块之间相互分离，每个组件都只关注自己部分的逻辑，彼此独立。

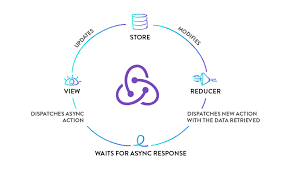
在本项目的构建中，考虑到系统的实时性能，以及实时的数据监测，部分模块具有一定的复用性，因此选择了React进行开发。

**Redux**



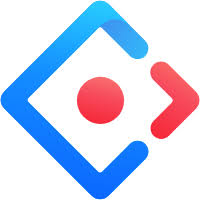
React中的状态管理十分的庞大，react中的状态可能来自服务器、缓存数据、UI状态，或者激活路由等，管理不断变化的状态是十分困难的，这个时候，我们引用redux来对react中的状态进行管理，让状态的管理更加的简洁和方便。

**Redux-thunk**



Redux可以很方便的利用Action，Reducer函数更新state来重新渲染View。但是其中的异步问题没有办法解决。这个时候我们就利用了redux的中间件--redux-thunk来解决这个问题，redux-thunk通过使得action操作脱离正常的流程，来到中间件进行执行，让页面的异步请求的问题得到解决。

**Antd**



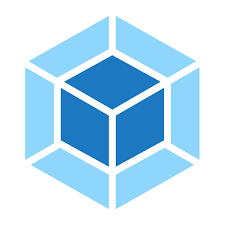
后端页面的实现功能的同时也要注重美化和方便，为此antd-react成为了不二之选，antd提供了大量丰富的组件，使得页面的搭建更加便捷，增加了页面的美观度、加快了页面框架的搭建以。

**Axios**



Axios是一个基于promise的HTTP库，可以用在浏览器和node.js。Axios的现在流行的网络库之一。在项目中，我们将axios进行二次封装，使得axios的功能更加强大，也让我们的网络请求变得更加的简便。以及解决前后端分离中的常见问题--跨域

**Webpack**



Webpack作为现在最流行的项目打包程序，自然是我们项目架构中的最优选择。我们利用webpack自定义了开发框架和流程，不再依赖react的create-react-app命令，实行了按需添加。并且使用热加载技术，使得开发效率得到极大的提升

**设计要点：**



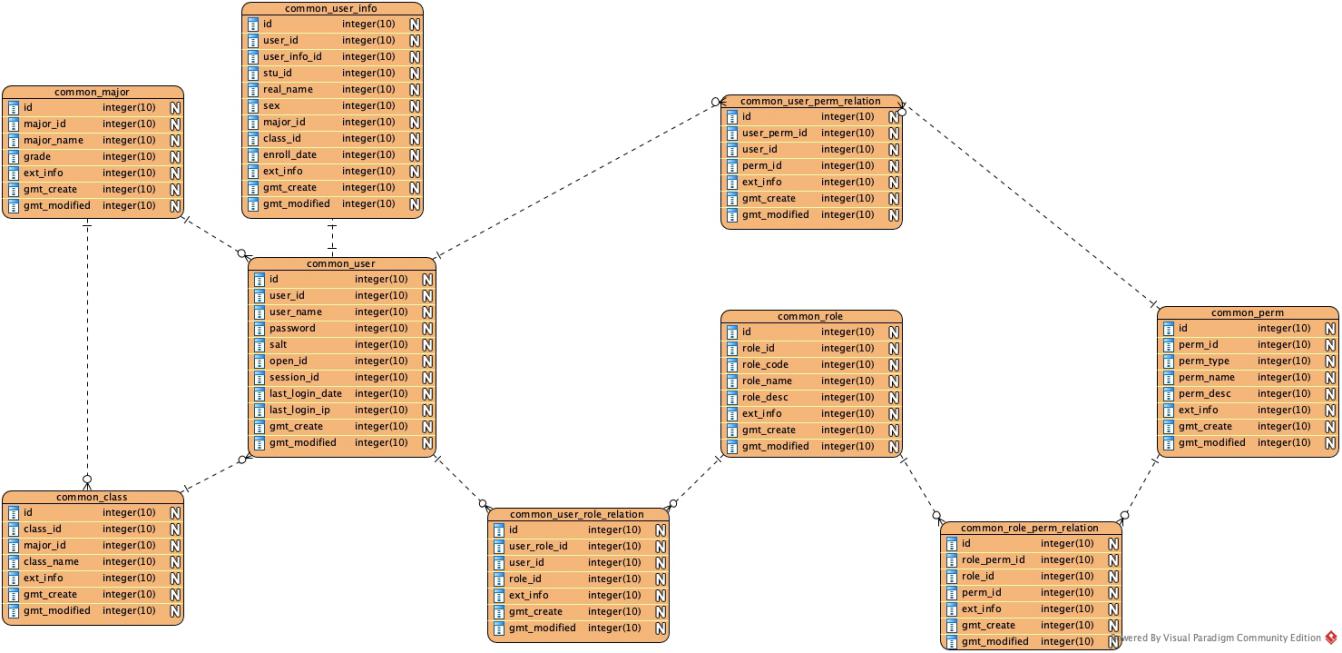
微信小程序采用了Hybrid app模式，由上图可见三种开发模式的优劣势，鉴于此，选择了微信小程序。小程序即点即用，用户使用成本低。在速度方面，主要样式代码都封装在微信小程序里，打开速度比H5快，慢于原生。同时，小程序可以调用更多的手机硬件功能，例如GPS，录音。由于支持微信的在线版本更新，开发、维护成本较低，同时维护几个版本的可能性较小

领域层

领域层(core层)以用户领域、活动领域、物资领域、财务领域、组织领域、证书领域为核心构成核心模块,biz层依赖core层.实现领域交际功能.

用户领域

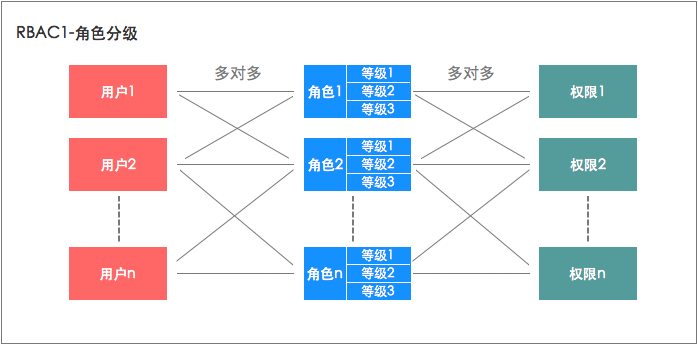
用户领域是第二课堂第一个搭建模块,小程序用户以学生为主体,面向学生属性进行数据库设计.



Rbac模型

RBAC是一套成熟的权限模型。在传统权限模型中，我们直接把权限赋予用户。而在RBAC中，增加了“角色”的概念，我们首先把权限赋予角色，再把角色赋予用户。这样，由于增加了角色，授权会更加灵活方便。在RBAC中，根据权限的复杂程度，又可分为RBAC0、RBAC1、RBAC2、RBAC3。其中，RBAC0是基础，RBAC1、RBAC2、RBAC3

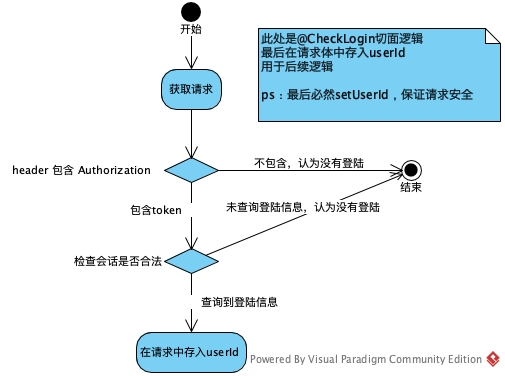
都是以RBAC0为基础的升级。对此第二课堂以RBAC1为蓝本建立完善的权限管理模型以保证最小权限原则，责任分离原则和数据抽象原则。



Jwt请求认证

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准（[(RFC 7519](https://link.jianshu.com/?t=https://tools.ietf.org/html/rfc7519" \t "https://www.yuque.com/cpwen/pyigqn/_blank)).该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

传统身份验证使用HTTP协议,用户发送请求到服务器,服务器创建唯一session记录存储到Cookie中.用户再次提交请求时,服务器对Cookie中session进行身份验证.使用session存储方式相对cookie来说更安全,但由于"第二课堂"使用nigix利用反向代理来实现负载均衡,导致下一个操作请求提交到另外一台服务器的时候session会丢失.故为解决该难点,系统使用JWT认证,服务端无需保存用户的认证信息或者会话信息,在提交登陆或者操作请求时,将token信息放入Header请求头Authorization中并使用Https对请求链接进行加密传输,后端对token参数进行验证后提供后续服务.最后会话合法,在请求中获取userId,保证请求安全.



Validations数据检验器

Hibernate中不仅有ORM一个框架，它的另一个框架Validator也是被经常使用，用来验证实体类是否满足需求。Validator实现了Java的一项标准Bean Validation。

"第二课堂"小程序,不仅实现数据导入,根据学生名单自动创建学号账号,并且用户可以创建账号.

数据校验是用户注册部分不可或缺的功能,小程序前端可对数据进行第一层校验,如账号密码合理性,但是为了避免用户绕过浏览器,使用postman等工具向后端传入违法数据,服务端的数据校验也是必须的.对此引入hibernate validation (JSR303).

后端继承validation实现openid校验器,密码长度校验器,密码强度校验器,用户账号校验器.

密码强度校验器:对密码进行等级计算,当密码含有数字时密码强度+1,对密码中是否含有大小写字符进行判断,密码长度分别+1,对特殊符号判断,密码强度+1,密码强度逐层累加,当密码强度小于2时,强度校验器返回异常信息,用户创建账号失败.

用户账号校验器:当用户提交创建账号请求,获取提交账号参数中用户账号参数,数据库验证是否有相同账号若有则返回异常信息,用户创建账号失败.