# 1 引言

步入信息时代以来，互联网给人们的生活带来了翻天覆地的变化，互联网也不再简单地仅仅通过提供便利快捷的资讯服务来丰富我们的生活。互联网的出现打破了许多传统行业垄断的格局，互联网以其接入面广、信息即使、人人可参与等等性质，迅速融入到了人们的生活中，并且已经成为了整个社会不可缺少的一部分。大约5年以前的互联网是属于PC的时代，那时人们的网络生活包括逛贴吧、看资讯、写博客、下载多媒体文件以及在线网购等等。互联网是一个时刻都在发展的行业，其发展速度之快是大部分传统领域望尘莫及的，一方面是因为互联网虽然已有近30年的念头，但相较于传统行业，这个年份几乎还是跟刚出生的婴儿差不多。因此互联网有极为广阔的发展空间，另一方面由于互联网本身就技术性质而言，属于一门科学领域，大量国内外的高等技术人才都在不断完善互联网领域的科学根基。从小到微机硬件性能一代又一代地提升，再到大型路由交换网络环境的接入覆盖范围越来越广、越来越快，再到现如今机器学习、人工智能、分布式计算、分布式存储的铺开，互联网领域逐渐形成了一个包含计算机科学、软件工程等等领域的超大集合。以前人无法做到的事情已经逐渐慢慢在通过机器来实现。对于互联网领域而言，只有针对用户有意义的应用才是有价值的。随着互联网的发展和移动设备的普及，目前移动端市场正在以突飞猛进的速度渗透到用户群当中去。人们通过随手可得的智能手机、智能平板，甚至电子阅读器等等诸多设备都可以便捷的通过移动4G网络随时接入互联网。移动4G和公共Wifi网络的速度提升和逐步普及为移动互联网深入人们的生活打下了良好的开端。越来越多的人开始使用移动端设备接入互联网丰富日常生活。从在线购物、在线订餐，订房、订票，到日常信息的获取和交流。

移动互联网是传统互联网的一个分支，但抛开其在移动端运行这一点而言，移动互联网跟传统互联网没有任何区别。然而仅仅是在移动端运行这一点而言，就足以体现出移动互联网与传统互联网的巨大区别。首先考虑用户的使用场景，一般而言，传统互联网的接入方式一般是PC浏览器，用户端座位上，面对着电脑屏幕获取互联网资讯。而移动互联网的用户没有这么多时间，他们可能是上班的路上拿出了手机，可能是吃饭时想发一张自拍，最常见的是与好友们的即使通讯，分享自己某一刻的心情等等。这些事情的共同点是操作时间短，使用地点自由。移动互联网因为其便捷的接入性完全满座以上需求。用户在使用智能设备访问移动互联网是没有端坐在桌面上通过PC浏览器访问桌面网页时那么有耐心，移动互联网让用户们得以随时随地享受快速的服务。

然而重新提到移动互联网和传统互联网的共同点时，就又要回归到技术层面上，那就是虽然互联网的接入模式在变，但是基于的技术架构仍然没有任何变化，当用户在移动设备上访问互联网时，和在PC上通过浏览器访问后端服务没有任何差别。因此此前在传统互联网领域所没有克服的技术问题也就顺势带到了移动互联网开发领域。这十分让人苦恼，在之前，由于传统互联网的接入方式通常是宽带接入，网络稳定性和速度有保证，于是开发者们可以牺牲一些新能和速度以换取用户的良好体验。然而当面临移动互联网时，开发者们要面临移动端网络环境不稳定、速度有延迟等无法避免的事情。又由于移动端通常不具备PC端良好的处理性能和存储空间的优势，因此移动端开发时需要对性能有更高的要求。除此之外，还有一个难以调和的矛盾，就是移动端涌入我们的生活了，对于用户而言是好事，他们可以通过更多种途径获取更多种便捷的服务。但是服务和业务规模的扩增对开发者而言却是让人头疼的难事，因为传统互联网一直没能很好地解决前后端耦合的问题。前端由于接入设备种类的增加以及业务规模地愈发复杂化，导致我们强烈需要把前端从后端中解耦出来，让前后端不近从运行平台上解开，更重要的是从代码设计和架构上实现完全的解耦。

在这些基础上，前人为了实现前后端解耦总结了许多实践方法，但每解决一个问题往往会引入新的问题。本课题就是试图引入一种新的模式，从而从根本上实现前后端的彻底解耦。通过微信公众号开发实现一个图书借阅平台的真实例子，从而验证所提出的新模式的可行性与架构方法。

## 1.1项目研究背景

Web软件研发这一行业，从互联网诞生的那一刻开始诞生，开发模式也在不断演进。从最初既做设计图又要写代码，不分前后端端一人通吃的蛮荒时代，到现在逐渐分工明确各司其职的现代化开发时代，Web行业正在以历史上最快的速度向前发展。在这个历史进程中，Web研发模式经历了如下阶段。

最初期可称为Web初期，那时不区分前后端工程师，对于大部分公司而言，所懂得技术的人才并不多，然而当初的业务成都也没有现在这么复杂。通常所有工作由一个小团队即可完成，HTML内容直接由JSP或者PHP等后端语言直接输出，交给浏览器来呈现，所有关于页面的展示逻辑都由浏览器来负责。浏览器端也很少参与业务逻辑，内容完全由服务器端程序来决定。这种模式的方便之处就是业务逻辑十分简单，业务核心依赖服务器端，代码结构简单清晰，利于调试和编写。然而一旦随着业务规模的增长，过多参杂了View层逻辑的服务器端代码就会不断增长，不单影响性能、效率，还严重制约程序开发的进度。服务器端的复杂度越来愈大，为了解决问题，单纯靠填补服务器端开发人员的数量也难以应对庞大的服务器端代码。

这期间遇到的一些典型问题令许多人深有感触，随着服务逐渐增长，API调用关系变复杂，服务器端程序部署成为问题。后端人员对代码做出修改后上线需要进行上线部署，而前端人员对页面的细微调整也需要与后端人员进行沟通，提交代码给后端进行处理，完成本地开发，代码合并，后端部署等诸多环节。这导致前端的任务严重依赖于后端，没有后端的前端程序完全无法独立运行，制约着前后端之间开发进度的衔接。

为了解决这种混乱不堪的开发状况，诞生了一种耦合式的前后端解决方案：即前端人员书写完页面代码之后，交给后端人员处理与数据库的对接，让前端人员给后端人员打下手。并且严格按照这种开发流程来执行。虽然这种方式解开了前后端人员之间的工作任务，但是并没有解开前后端之间耦合的代码。

到了Web2.0时期，浏览器所呈现的内容愈发丰富，各种Web应用层出不穷，开始强调客户端丰富的用户交互。此时再把前后端耦合在一起显然不合适了。所以为了降低复杂度，开始在Web后端领域实践MVC开发模式。这一阶段涌现出了一些非常优秀的后端框架，比如RubyonRails、JavaSpring、Django等等。后端MVC强调模型、视图和控制器职责的分离，而一直以来前端的工作都是与View层紧密相连的，所以后端可以完全专注于Model和Controller的开发工作，而前端则专注于View层页面的开发。当前端编写完页面模板之后，交给后端去渲染Model层的数据。虽然前后端各自的关注点分离了，但是依然没有解决代码耦合的问题。对于后端而言，Controller层和Model层纠缠不清，Model层又和View层纠缠不清。虽然后端MVC是一种非常优秀的开发模式，但是前后端耦合这些根源问题依然没有解决。

随着Google推动Ajax技术的发展，现在几乎所有网站都有Ajax的身影，异步的JavaScript和XML技术给Web开发模式提供了全新的思路。服务器端单纯以服务的模式提供数据，View层完全独立于后端的服务，前端所需的数据都通过Ajax异步请求向服务器端获取，页面内容的渲染工作全部由前端完成。这种模式下，前端与后端实现了完全的分离，交互的关键点就是Ajax接口，但是此时，服务器端的业务的虽然降低了，但是前度的复杂度却增加了。一切总归是平衡的，后端所减轻的负担，现在全部施加到了前端上。这个阶段，是前端工作人员最痛苦的时期，JavaScript走向了之前JSP和PHP走过的路，前端代码陡增，前端尝试引入框架来解决代码逻辑问题，但前端的压力依然很大。

到了最近一个阶段，也是目前业界正在广泛使用的一个模式，即在前端也部署MVC甚至MVVM这种大型软件的构建模式。这一阶段前后端之间的代码耦合已经降低到最低了，几乎解除了Web开发之间的代码耦合问题。后端专注于业务逻辑的开发，通过RESTful接口输出数据，前端只要准从设计模式的标准，也可以应对复杂的交互逻辑了。但是方便的背后，也带来了一些其他问题。比如全部异步的请求模式给前端编码带来难度，全部通过请求接口来渲染数据对搜索引擎也十分不友好，不方便爬取页面数据。而且对于移动设备而言，尤其是手机，在网络状况不好的情况下，前端页面需要等待网络请求返回到结果才能呈现。对于前端SPA应用，最重要的路由问题难以和后端调谐，长此以往路由都是交给后端处理的，但是现实的情况是前端对路由的依赖更加严重。

展望未来，业界正在积极寻找一种新的解决方案，一个新的模式。既能解除Web开发面临的耦合问题，又能让前后端关注自己的职责，并且彼此的功能划分地更加细致。让前端可以脱离于后端，让后端可以专注于业务服务。随着Node.js技术的发展，再一次为Web研发模式带来了新的思路。Node.js与前端共享开发语言，可以与前端实现高度的代码重用，也赋予了JavaScript在服务器端执行的能力。这种新的模式就是在传统后端和前端之间，引入一个Node.js中间层来调和矛盾和彼此的缺失。以往所面临的问题，随着Node.js的引入都可以得到解决，所带来的代价便是需要前端工程师对后端有更深入的理解，需要他们设计更多后端的知识，已经引入一个新层之后的通信效率略微下降。更多的制约可能是面临新的技术解决方案时的担忧，缺乏对新技术的实践经验，无法短期内改善大量的历史代码。

## 1.2 项目研究内容

本课题所要研究的是长久以来Web研发领域前、后端耦合的问题，进而提出一个全新的解决方案，并通过开发一个基于微信的图书借阅系统展示新方案的可行性。在此以前，我们所面临的耦合问题的广义层面含义包含很多中，下面逐一分析这些问题的具体内容。

从代码的耦合来看，Web研发领域在初始时期由于代码规模较小，并不注重代码分离。导致出现了不区分前后端的混合式编程模式。比如页面数据从数据库获取之后直接写入到页面当中输出给浏览器端展示。衍生出的问题包括代码糅杂和混乱不堪，难以区分服务端代码和客户端代码，不容易做分离，代码的灵活性不强，不利于代码复用和拓展。业界要想实现中到大型规模的软件产品，与团队协作的重要性不可分。需要由不同的团队负责不同的模块代码开发，所以解耦的第一步就是要解开代码层面的耦合，以适应大规模的代码量。

从业务的耦合来看，以电商网站为例，用户完成一个完整的购物流程包括浏览商品，搜索商品，购物车下单，支付，等待收货，完成评价。既然作为一个整体的流程，那么每个环节之间必然会出现耦合，比如搜索功能依赖展示功能呈现结果，支付功能依赖购物车功能计算总价，评价系统依赖用户信息系统。安装业务的属性来分，展示功能属于浏览器端，搜索和支付属于服务器端，但脱离前、后任何一端，任何功能都无法完成。传统Web研发领域由于前后端耦合的问题，导致前后端的职责并没有很好地区分开来，虽然从业务职责上各个模块是分离的，但是前后端确实耦合的，这就给开发带来了难度，依旧不能解决随着业务规模扩展带来的开发压力。

从服务的耦合来看，随着目前移动互联网的火热，用户接入互联网的方式不再局限于电脑。使用安卓智能手机、苹果智能手机、各类平板电脑、所占据的用户比例甚至已经超过了传统使用PC浏览器的上网比例。这又给Web研发提出了新的要求，如果统一提供跨平台的服务。传统Web研发由于后端服务与前端服务耦合，不方便做统一地跨平台支持。同样的，随着用户设备多样性的增加，需要提供越来越多的服务需求，开发压力也将陡增。

本课题通过研究Web领域历史以来的演变过程中所诞生的各种技术的优缺，通过探索新的模式，提出新的方案来解决耦合这个历史问题。以达到让前后端的代码更分离、职责更清晰、服务更统一、分工更合理高效的目的。

要解决问题需要首先分析问题出现的原因，传统Web开发带来的制约源于后端没有纯粹地负责数据处理和存取功能，而是过度耦合了前端呈现和用户交互。后端应该不区分端，只负责提供统一地服务，然后各个端之间负责利用后端提供的服务接口完成与用户的交互和业务逻辑。这样就找到了解决问题的方法。

所尝试探索的途径是通过引入Node.js中间层技术，在Web服务器上同时部署服务器程序和Node.js中间层，当用户请求前端页面时，前端首先访问Node.js中间层进行页面关键数据的渲染，当后续需要数据时即可跳过Node.js中间层直接请求后端。当用户进行页面跳转时，使用Node.js中间层和前端同步页面路由，对于用户的状态管理等与业务逻辑不是强相关的服务也由Node.js来提供。依然会依赖Ajax技术来处理大量的数据依赖，除了引入Node.js之外，其他的前后端技术依然保持不变，变的是它们之间的组合关系。本课题所要实现的，也正是这样一种合理利用Node.js中间层解决前后端耦合的研发模式。

## 1.3 项目研究意义

就软件工程领域而言，近些年发展最快的莫过于Web开发。绝大部分互联网公司采用Node.js、PHP、Python、JavaWeb、Ruby、Go、HTML/CSS/JavaScript等等技术为用户提供丰富便捷的互联网服务。这些服务涵盖我们日常生活的方方面面，从衣食住行到社交通讯，到处都需要Web开发提供服务程序。对于一个完整的应用声明周期而言，从开发到测试再到上线再到版本迭代，这期间会不停地伴随着功能增加和业务规模的扩增，因此Web开发对程序设计的要求需要具备灵活的拓展性。程序等于数据结构加算法，这是一条基本常理，无论多么复杂的程序，归根结底就是在做这样两件东西。相比之下，软件架构的设计也离不开另一个基本的常理，那就是高内聚低耦合。无论是命令式编程的独立函数还是面向对象式编程的设计模式，无论是MVC还是MVVM架构，在设计软件架构的时候，都要时刻考虑这两个基本原则。因为只有软件的模块、类、功能、服务彼此之间尽量独立，尽量不要依赖别的部分，自己内部的状态尽量保持封闭这些都能做到时，才能保证模块可以随意增删以及单独测试等等。研究耦合这个问题，就像研究数据结构与算法对于程序设计的意义一样，是Web应用程序架构的基础。对于Web开发这种功能快速变化，版本快速迭代的模式而言，如何实现高度的解耦是功能业务能否快速拓展以及保证前后端开发人员各司其职高效配合的重要基石。

研究Web研发新模式的意义，不是一味地标新立异，而是真正着眼于当下Web开发领域所面临的开发效率低效这个切实问题。契合当今前、后端独立发展，开发人员职责合理分配，让团队之间的协作更加高效，并让产品的开发速度，软件服务的可拓展性、灵活性、独立性提升的需求，解放生产力。

# 2 技术选型与开发环境

本微信图书借阅平台是一个完整的B/S架构的实现，在服务器端通过Nginx监听用户请求并分发请求给StrongLoop进程管理器，StrongLoop创建多个slave进程维护Node.js实例来处理用户请求，Node.js作为中间层可以构建在任何其他语言的后端之上，通过基于RESTful规则设计的接口可以建立Node.js中间层与其他后端语言的数据通信。数据库采用MySQL、通过Jade模板引擎实现服务端HTML预渲染、在客户端通过微信内置的Blink内核的WebView作为与用户交互的入口、UI效果展示采用Bootstrapcss界面库快速制作移动端友好的响应式界面。

## 2.1 技术选型

### 2.1.1Node.js介绍

Node.js是一个基于事件驱动的JavaScript异步运行环境，从一开始设计之初就是针对互联网这种节点灵活、易于扩展、高性能的场景而生的。Node.js处理并发的性能非常好，下面有个简单的实例，展示Node.js如何用简短的代码实现一个HTTP服务器，并且可以承受极高的并发请求量。当有HTTP请求到达服务器端时，Node.js迅速处理调度事件循环处理请求，当没有HTTP请求到达服务器是，Node.js就自动挂起进程，节约CPU和内存资源。

示例：用Node.js创建一个HTTP服务器

consthttp=require('http');

consthostname='127.0.0.1';

constport=3000;

constserver=http.createServer((req,res)=>{

res.statusCode=200;

res.setHeader('Content-Type','text/plain');

res.end('HelloWorld\n');

});

server.listen(port,hostname,()=>{

console.log(`Serverrunningathttp://${hostname}:${port}/`);

});

Node.js在设计上受到了RubyEventMachine和PythonTwisted的影响，但对于事件模型的处理上Node.js更进一步，Node.js不依赖第三方库，在运行时自动创建一个事件循环处理任务。一般在别的时间循环系统当中，需要在代码起始执行的地方显式地通过阻塞方法开启一个事件循环，然而这些在Node.js当中都是不需要的，Node.js在执行代码文件的伊始就自动开启了事件循环。这种模式跟浏览器端JavaScript脚本的执行非常类似，异步操作都是浏览器在幕后完成的。

HTTP是Node.js中非常重要的一部分，因为Node.js的初衷就是针对互联网上大规模可灵活拓展的HTTP服务而设计的。因此Node.js作为提供HTTP服务的基础服务层十分适合。

### 2.1.2 异步编程介绍

如上所述，可以看出Node.js的异步处理模型与常见的基于系统线程实现并发处理的模型有很大反差。基于线程的网络模型唯一的优势就在于用最低的资源开销实现并发处理，但是多线程编程非常不利于控制和管理，相比之下，单线程的编程模式就简单直接多了。虽然Node.js引入的是异步编程，但目前已经有良好的异步编程解决方案，比如Async/await+Promise的解决方案使得异步编程可以像同步编程一般方便直观。反观多线程编程，多个线程之间的任务如果是非相关的，那么影响倒不是很大，然而一旦多个线程依赖同一个数据源，就会导致资源争夺而无法合理分配的情况，需要引入锁机制来防止死锁。而且多线程编程还有一个致命问题在于一旦其中一个线程出现异常，但是这个异常没有被正确处理（这十分常见），就会导致主线程崩溃的情况，最终导致所有线程崩溃。所以为了提升多线程程序的鲁棒性，必须依赖经验丰富的程序员通过巧妙的机制来合理处理资源争夺、数据不同步、异常崩溃等问题。仅仅是为了节省资源而引入多线程编程，在大多数情况下并没有这个必要。

示例：Node.jsAsync/await+Promise解决方案

function\_fetchbooks(){

returnnewpromise((resolve,reject)=>{

//ajaxoperation

if(ajax.success){

resolve(ajax.response);

}else{

reject(ajax.error);

}

});

}

asyncfunctionfetchbooks(){

constbooks=await\_fetchbooks();

console.log(books);

returnbooks;

}

而Node.js的开发者几乎不需要考虑进程死锁的问题，因为单线程的执行环境中并没有锁这个概念。而且Node.js中几乎所有函数都是异步I/O函数，因此Node.js的主进程永远不会阻塞，又因为不会发生阻塞，所以扩展系统业务的规模也十分方便。虽然Node.js被设计成不依赖线程编程，但并不意味着Node.js不能发挥CPU多核心的性能。通过child\_process.fork()API可以灵活地复制出子进程，而且它们彼此之间的通讯也经过了特殊设计，非常方便。基于内置的cluster模块，可以在本地的多进程之间共享sockets，充分发挥多核心的优势。

因为随着计算机的普及和计算机硬件行业的发展，服务器上所采用的硬件在处理能力上正在发生翻天覆地的变化，绝大多数公司在考虑后端技术选型和服务器硬件配置时，都是首先考虑技术的复杂度，服务器能承受的并发压力，未来产品迭代的速度等等。除非是顶尖的互联网软件行业对性能要求极高的公司，对于大部分公司而言，Node.js所提供的这种简洁轻快、高效方便的优势都会对快速产品迭代开发带来极大的优势。

### 2.1.3 阻塞和非阻塞介绍

阻塞是指当执行JavaScript代码时，Node.js进程必须等待非JavaScript操作结束，之所以发生这种事情是因为事件循环在阻塞操作发生时无法继续执行。比如说读取一个大文件，当Node.js执行读取文件的代码时，必须等待操作系统将整个文件读取完毕，否则Node.js的事件循环将无法继续工作。因此在Node.js中面临CPU密集型任务时，JavaScript通常会表现出很差的性能，但是在处理I/O密集型任务时，则表现出优异的性能。Node.js标准库中基于libuv的同步方法一般都调用阻塞操作，而异步方则调用非阻塞操作并且接收回调函数用来处理异步的结果。通常Node.js会提供两个版本的函数，默认是异步方法，方法名后面加上sync的是同步方法。

示例：Node.js阻塞方法

constfs=require('fs');

constdata=fs.readFileSync('/file.md');//blockshere

示例：Node.js非阻塞方法

constfs=require('fs');

fs.readFile('/file.md',(err,data)=>{

if(err)throwerr;

});

在上面的阻塞方法示例中，Node.js会等待文件全部读取完毕才能继续往下执行，这将导致Node.js进程阻塞无法继续执行，等待非JavaScript操作也就是操作系统读取完毕后，方可继续执行。而在非阻塞方法的示例中，当执行readFile之后程序会立即结束，继续往下执行，但是读取文件的任务并没有终止而是放在了事件循环中去做，当文件读取完毕后回调函数会被出发。

### 2.1.4 MySQL数据库介绍

MySQL从最初开始就被设计为一款支持SQL查询语言的关系型数据库管理系统，最初MySQL以开源的形式向外界发放。它的兼容性非常好，支持Linux系统、Windows系统、Unix系统。由于其广泛的支持度，MySQL被用于各种类型的应用程序的开发之中，当然MySQL的最大用途还是作为在线Web应用的数据库，也就是我们所常用的网络应用数据库服务器。MySQL不光以其强大的兼容性风靡各大平台，其中广为熟知的LAMP一站式打包服务更是为MySQL在Linux上的统治地位奠定了强有力的基础。配合Linux内核操作系统、Apache应用服务器、MySQL数据库服务器、PHP（Python或Perl）网络开发语言，无数中小企业通过这些技术向用户提供各种丰富的网络服务。

虽然MySQL最初以开源的方式向外界授权，后来被瑞典MySQLAB公司收购，后者2008年又被甲骨文公司收购，甲骨文也因此获得了MySQL的版权，收购MySQL之后的甲骨文公司宣称MySQL面向普通用户将永远免费，只有企业用户需要获得授权才能使用。也正因此导致MySQL发展出继续开源的MariaDB分支，MariaDB保持跟MySQL完全的兼容性，但仍会以开源的性质继续发布。

MySQL的特点：

a）MySQL是一款完善的关系型数据库管理系统。

b）MySQL基于Client/Server架构，只需要一个Server端，就可以支持大量的Client端接入。对于集群式应用开发，只需要部署一台MySQLServer服务器即可。

c）相比于Oracle、DB2、SQLServer等商业服务器而言，MySQL对个人用户完全免费，也正因为如此吸引了大量的用户。

4）MySQL性能优异，安全性、跨平台兼容性都非常优异。

5）MySQL对SQL的支持度很好，并且支持SQL的最新标准。

6）MySQL功能完善，支持外键约束、索引、子查询、视图、过程、触发器、Unicode字符、全文搜索等实用功能。

7）MySQL支持事务回滚，数据操作失败时根据数据库读写日志恢复到出错前的状态。

5）MySQL社区用户活跃，社区对于一款开源软件的影响将起到主导作用。

MySQL操作：

MySQL可以在Windows、Linux和Unix下很好地工作，推荐在Linux命令行中安装MySQL，过程快捷便利。以Ubuntu操作系统为例，执行如下命令即可安装MySQL服务器端和客户端：

$sudoapt-getinstallmysql

执行如下命令即可进入MySQL数据库进行数据库操作：

$mysql-uroot-p

mysql>CREATETABLEmytable;

除了命令行方式以外，MySQL支持多种语言的API接入，MySQL服务端默认监听服务器的3306端口，等待来自外界的TCP连接，常见的Web开发语言通过各自版本的MySQL连接驱动即可成功连接MySQL数据库执行数据库操作。

### 2.1.5 Nginx服务器介绍

Nginx是一个高性能的Web和反向代理服务器,它具有有很多非常优越的特性:

作为Web服务器：相比Apache，Nginx使用更少的资源，支持更多的并发连接，体现更高的效率，这点使Nginx尤其受到虚拟主机提供商的欢迎。能够支持高达50,000个并发连接数的响应，Nginx支持epoll、kqueue等多种开发模型.

作为负载均衡服务器：Nginx既可以在内部直接支持Rails和PHP，也可以支持作为HTTP代理服务器对外进行服务。Nginx用C编写,不论是系统资源开销还是CPU使用效率都比Apache等其他服务器要好很多。

作为邮件代理服务器:Nginx同时也是一个非常优秀的邮件代理服务器。

Nginx安装非常的简单，配置文件非常简洁（还能够支持perl语法），Bugs非常少的服务器:Nginx启动特别容易，并且几乎可以做到7\*24不间断运行，即使运行数个月也不需要重新启动。还能够在不间断服务的情况下进行软件版本的升级。

### 2.1.6StrongLoop进程管理器介绍

StongLoop是Node.js集群工作环境下最完善的运行时管理控制和监控程序，StrongLoop在吸取了Node.js社区一些现有的进程管理器（forever、pm2）的基础上又扩增了独有的功能，StrongLoop支持的功能如下：

a)图形化操作界面；

b)集成nginx负载均衡；

c)支持远程管理；

d)自动化多站点部署；

e)集成运行监控和性能评估；

f)多机器、多进程分布式实时控制。

### 2.1.7微信公众号平台介绍

微信公众号是腾讯公司所推出的即时聊天工具“微信”中，面向非普通用户即具备推广能力的个人用户和企业用户所推出的推广平台。通过开放授权和实名审查，吸引有开发和推广能力的人员使用。普通用户可以关注微信公号，获取资讯和服务。微信公众号的本质是提供一个第三方平台打通内容推广者与大众用户的桥梁。

申请个人公众订阅号和商业公众企业号的用户，首先需要首先到微信官网注册账号，个人公众订阅号用户需要提供证件申请实名认证，商业公众企业号用户需要提供商业资质证明。等待申请结果成功后，用户们便可以通过微信搜索功能搜索公众号名字，查看详情后关注该微信公众号。开发者在微信后台配置微信与自己的服务器之间的连接，从而打通从用户到微信服务器再到个人服务器之间的消息传递，并将处理后的结果回传给微信服务器，最终转发到用户的微信客户端上。开发者有两种方式与用户进行交互，一是通过聊天信息和推送小时与用户进行信息交互，二是通过微信公号底部的自定义菜单栏引导用户触发指定的功能，与用户完成交互。微信提供OAuth2.0认证机制，帮助开发者在经过用户的授权之后获取用户的一些非隐私性个人信息，比如用户id、头像、昵称、地域等等信息。帮助开发者给用户提供更加精准、有效的服务。

# 3 需求分析

自从移动互联网兴起以来，开发人员面临了更大的挑战，要求也更加严格。传统的仅基于PC浏览器平台的Web应用模式无法满足现如今各大移动终端平台的需求。面临更多的平台，需要提供针对不同平台的服务来满足用户需求，Web应用的定制性越来越强，代码规模越来越大。为了满足新的业务需求，提升开发效率，节约开发成本。需要将前、后端开发的人员独立开来，让彼此各司其职。在这个需求下，我们需要达到如下几点：对于后端而言，仅负责业务逻辑和数据接口的开发，一套服务和接口的独立性尽量高，从而可以被多套业务重用。而对于前端而言，负责保持用户的状态，为用户展示界面效果，与用户进行逻辑交互，一套界面组件的独立性尽量高，从而可以被多套业务重用。

## 3.1 技术需求

其实Web开发解耦这个话题的讨论从若干年前就早已开始了，只是随着Web研发一个阶段一个阶段的推进，业界虽然探索出了一些比较实用的方案，但是没有从根本上解决问题。另一方面因为Web领域的受众用户基础量太大，一旦采用新模式重构旧代码，会导致伤筋动骨，之前大量的业务和逻辑需要重写。所有很多时候，历史问题比技术问题更难解决。

当我们讨论前、后端解耦，我们在讨论什么？对于前、后端解耦，大家对于其认知不尽相同。对于大部分初步接触解耦的人而言，比较认同的解耦模型是SPA（Single Page Application）模式，这种模式的技术特点是，后端处理所有的业务逻辑和数据，但不处理任何与视图界面和用户交互方面的问题。后端仅通过RESTFul、SOAP或普通的HTTP请求对前端发出的请求给予响应。到了用户这一端都是前端在进行处理，除了页面上硬编码的HTML结构，所有的动态数据都通过接口请求后端获得，然后动态渲染视图展示给用户，所有与用户交互的操作都有前端完成。这种架构是一种非常彻底的解耦模型，但是它太过于彻底了，以至于前后端之间充斥着大量的接口请求操作，以完成视图界面的创建。对于前端而言，又由于请求都是异步操作，会带来编码上的复杂性。

除了SPA模式之外，目前业界没有其他更成熟的前、后端解耦方案。在PC浏览器时代，由于物理带宽得到稳定保证，因此浏览器发出的HTTP请求通常可以得到快速响应，而且上网花销比较低。然后随着移动互联网的普及，流量费用是一笔不小的开销。而且移动网络的稳定性并没有传统物理宽带那样稳定，SPA架构的Web应用被这两个问题频繁困扰，不通过请求获取数据，页面上将会呈现大片空白区域，网络速度不稳定，加载的数据不完整，频发发送请求，对用户的流量也是一个比较大的负担。再之，SPA的可用范围比较窄，对于以内容为主的Web应用，比如新闻类和分享类的网站，需要被搜索所收录，SEO对于此类网站而言尤为重要。而SPA应用的网页内容都是通过JavaScript异步请求后端接口获得数据后动态插入到页面当中的，搜索引擎无法执行JavaScript，因此无法收录网站的内容。这将会对企业造成极大损失，因此SPA只能在有限的场景下发挥作用。以上便是我们所熟知的前后端解耦。

为什么要探索前、后端解耦的新模式？对于目前现有的几种Web研发模式而言，它们都有各自适合的开发场景，没有谁能够完全取代别彼此。这就造成了开发人员为了兼容多套不同的平台业务，需要采用多种方案来实现，学习成本高，维护的代价也大，需要大量的开发人员来维护多套解决方案代码。基于Node中间层的解耦方案，可以实现前、后端解耦架构的大一统，可拓展性、灵活性都非常强。不管是垂直业务深度拓展，还是水平平台种类覆盖方面，Node中间层都能以一个灵巧的胶水的角色拟合它们的差异，实现极高的重用和复用。

对于现今Web研发的开发效率而言，前、后段人员的分工，团队之间的配合，代码的整合与部署，新产品的迭代，出错后的容灾回滚等等都受到现有开发模式的限制。因为前、后端的工作内容没有分离开，为了解决分分合合的代码问题，将会浪费大量的时间处理非技术性问题。

另一方面，前端人员所能做的事情一直受到后端的制约，前端所开发出来的应用无法再脱离后端的情况下运行，难以调试、部署。过渡依赖于后端环境提供前端运行环境的支撑。又因为前端需要耦合到后端当中，也就意味着后端参与了很多本应该前端做的事情，为了给后端做出让步，前端不能充分发挥自身的能力，需要把功能做得尽量简单，转而通过后端来实现复杂的逻辑。后端又因为需要处理与用户交互先关的内容而叫苦连天。

还有十分重要的一点，就Web应用的性能而言，服务器在承受着巨大并发压力的情况下，如果参与了过多关于视图界面相关的工作，会影响CPU的处理能力和内存资源的占用。而将这些工作搬到前端之后，后端的职责更加清晰，仅需要执行纯粹的数据处理的请求处理就可以了。

以上这些问题是Web研发领域面临的迫切需求，只有进行良好的设计解除前、后端之间的耦合关系，充分发乎前后端开发人员的能力，才能更加高效地推动行业的发展。

## 3.2 功能需求

微信图书借阅平台提供添加图书、借阅图书、归还图书、浏览图书等功能。用户首次向书架贡献自己的图书时，使用添加图书功能；当用户需要查看当前书架图书信息时，使用浏览图书功能；当用户需要从书架上借阅图书时，使用图书借阅功能；当用户阅读完毕图书时，使用归还图书功能。用户用例图如图3.1所示。

关注公众号

归还图书

添加图书

浏览图书

借阅图书

图3.1 用户用例图

## 3.3 系统操作流程

微信图书借阅平台的使用流程有三个步骤。第一步，用户关注微信图书借阅平台公众号。第二步，用户使用微信图书借阅平台的各个功能，包括添加图书、浏览图书、借阅图书、归还图书。第三步，扫描图书条形码，完成图书信息读取，并确认图书的添加、借阅、归还等功能。系统流程图如图3.2所示。

### 3.3.1 关注微信图书借阅平台

目前微信平台使用的广泛度已经家喻户晓，作为即时通讯软件，它还提供了开发平台共开发者使用。因此微信图书借阅平台基于微信，不需要下载任何软件，只要用户关注“微信图书借阅平台”公众号，就可以使用平台的图书相关功能。“微信图书借阅平台”作为一个公众号，可供任何拥有微信号的用户搜索并关注，点击微信界面右上角“+”，在搜索框输入“微信图书借阅平台”，点击进入平台主页，点击关注按钮，便完成了关注过程。

### 3.3.2 功能菜单

微信图书借阅平台公众号的主界面是由三个一级菜单组成，包括借阅图书、归还图书、管理图书。其中管理图书菜单下还有两个二级菜单，包括添加图书、浏览图书。用户第一次使用本平台可能是要贡献图书的贡献者用户或者是借阅图书的借阅者用户，对于贡献者，点击管理图书菜单下的添加图书，并将图书放入书架；对于需要借阅书籍的用户，点击借阅图书菜单。用户需要借阅书籍时，可以点击浏览图书，查看当前书架上有哪些图书，点击具体的某本图书，查看图书相关信息。

### 3.3.3 扫码

微信图书借阅平台并没有图书馆手持款图书条形码扫描仪，但是，平台使用微信的“扫一扫”功能，完成对图书条形码的扫描，从而获取实体图书的图书信息。添加图书、借阅图书、归还图书的功能都需要获取图书信息，通过点击功能菜单会自动转到扫一扫界面。

### 3.3.4 确认操作

微信图书借阅平台的操作流程是关注“微信图书借阅平台”公众号、选择功能菜单、进入扫码、完成操作确认。系统操作流程图如图3.2所示。

开始

关注公众号

浏览图书

归还图书

借阅图书

添加图书

扫码

确认操作

结束

图3.2 微信图书借阅平台操作流程图

# 4.系统设计

概要设计是建立在需求分析的基础上的，概要设计的主要任务是，通过仔细分析软件规格说明，适当的对软件功能进行分解，从而把软件划分为模块，并设计出能完成预订功能的模块结构。

## 4.1概要设计

概要设计是软件设计具体实施阶段的第一个环节，通过给出核心功能点和设计要素的概括性描述，指导设计开发的方向。在这个阶段的主要任务一方面是概括地描述出整个软件的整体设计。另一方面是将工程概括的拆分出主要的核心模块描述各个模块大致的实现方案。

a）整体设计

1)微信公众号自定义菜单

微信公众号是用户使用该图书分享平台的入口，公众号底部的自定义菜单是引导用户进行功能操作的入口，需要设置简洁直观的入口让用户快速了解菜单对应的功能。

2)前端交互

图书分享和借阅应该是一个非常便捷的操作，不应当涉及过多的交互步骤，使得各个功能一触即达。

3)Node.js中间层

中间层的职责是前端与后端的桥梁，负责从后端获取数据并渲染出页面或进一步处理数据后传递给前端。

b）模块设计

1）借阅图书模块

图书借阅模块实现了图书的借阅操作，读者从书架上拿到图书，通过扫描图书背面的条形码确认借阅完成借书操作，取走图书。

2）归还图书模块

图书归还模块实现了图书的归还操作，当读者阅读完毕后，扫描图书条形码后确认图书归还，然后将图书归回原位。

3）添加图书模块

图书添加模块实现了图书的入库操作，读者拿出自己的图书，扫描图书条形码后确认图书归还，然后将图书放入书架。

4）浏览图书模块

浏览图书模块实现了浏览图书信息的功能，用户可以远程查看书架上的书籍信息，包括书籍名等基本信息、书籍数量、书籍可借阅数量。

## 4.2详细设计

用户进入微信后，关注公号进入公号界面，通过公众号主页底部的自定义菜单进入相关操作界面。点击“借阅图书”自动弹出二维码让用户扫描书籍自动借阅。点击“归还图书”自动弹出二维码让用户扫面书籍自动归还，点击“添加图书”自动弹出二维码让用户扫描书籍完成添加，点击“浏览图书”跳转到设定的网页浏览当前书库的所有书目。

### 4.2.1 微信公众号设置

申请个人公众订阅号和商业公众企业号的用户，首先需要首先到微信官网注册账号，个人公众订阅号用户需要提供证件申请实名认证，商业公众企业号用户需要提供商业资质证明。等待申请结果成功后，用户们便可以通过微信搜索功能搜索公众号名字，查看详情后关注该微信公众号。

在微信后台配置微信与自己的服务器之间的连接，从而打通从用户到微信服务器再到个人服务器之间的消息传递，并将处理后的结果回传给微信服务器，最终转发到用户的微信客户端上。

微信图书借阅平台通过企业公众号底部的自定义菜单栏，分别引导用户至不同的功能，与用户完成交互。菜单栏分为一级菜单和二级菜单，本平台使用借阅图书、归还图书、管理图书三个一级菜单，管理图书下有添加图书、浏览图书两个二级菜单。

### 4.2.2 获取图书信息

实现一个图书数据库的工作量非常庞大，设计到市面上所有已有图书的信息采集入库操作。为了便利地获取图书信息，采用豆瓣开放的公共图书API接口实现图书信息查询功能。用户首先打开微信进入公众号界面，通过公众号底部菜单中的“添加图书”菜单，点击后自动转至图书条形码扫描界面，用户扫描图书背面的条形码后，微信将识别出条形码中包含的ISBN号并回传给前端程序。前端程序根据将ISBN回传给Node.js中间层服务器，Node.js向豆瓣开放API服务器发送RESTful请求，附带着图书的ISBN信息，取得图书信息（图书名、图书封面图片链接、图书内容摘要、作者、出版社等）。

### 4.2.3 存储图书信息

所有图书的信息都通过自豆瓣的开放API接口服务获得，每当采集到一本图书的信息，便会执行一次图书信息的存储任务，将图书信息存入服务器本地数据库中。这样做的原因是，一方面便于再次获取图书信息时，可以快速查询图书结果。另一方面将初次获取后的信息按照关系存入本地数据库有利于信息的统计和变更。

在取得图书信息后，用户点击确认按钮，图书的书名、ISBN号、作者、出版社、封面、摘要等信息会存储到数据库中，并附加与借阅相关的图书信息，比如图书库存量，借阅次数，当前借阅者的信息等等。

### 4.2.4 展示图书信息

Node.js中间层在取得图书信息后，根据预先定义的Jade模板渲染出图书界面的HTML内容。微信用户内置的Webview控件在收到HTML内容后，渲染内容，加载图书封面等静态资源，经过CSS样式表处理后渲染出图书展示界面。图书信息的展示工作完全由Node.js中间层负责，当用户在图书页面之间跳转时，均直接请求Node.js中间层服务器，交由Node.js选择指定的Jade模板渲染内容。

系统功能模块图

前端

后端

Node.js中间层

微信Webview浏览器

RESTful over HTTP/HTTPS

RESTful over HTTP/HTTPS

业务逻辑/数据接口服务器

RESTful over HTTP/HTTPS

# 5数据库设计

微信图书借阅系统的正常使用需要有正确的数据库读取操作，数据库设计的优劣与否关系到系统能否运行流畅。微信图书借阅系统的数据库设计主要分为概念结构设计和物理结构设计，概念结构设计主要分析实体关系，以E-R图形象的展示实体关系。物理结构设计将实体关系转化成数据库中表的关系，以及表结构中各个字段的设置。

## 5.1概念结构设计

微信图书借阅系统的实体是各个用户，用户之间的地位平等，每个用户都可以贡献图书、借阅图书、查看图书借阅情况。本系统的数据库E-R图如图5.1所示。

摘要

总数

图书id

图书名称

封面

作者

出版社

贡献者

可借阅数

图书

借阅id

n

用户id

查看

借阅

图书id

m

借阅时间

用户

用户标识

用户名

用户id

图5.1系统E-R图

## 5.2物理结构设计

本系统使用的数据库是MySQL数据库，设计的充分考虑了程序处理的要求、应用环境的需求，斟酌数据库表结构和字段设置，考虑数据库各表中的字段类型和长度，尽可能提前预估到可能出现的问题，避免程序设计过程中带来不必要的麻烦。MySQL数据库中，凡字段为varchar或char类型时，需要指定编码方式为UTF-8，本系统对以上两种类型数据，全部设置为UTF\_8编码方式。

微信图书借阅系统共设计了3张表，分别是图书表（Books）、用户表（Users）、借阅表（Borrowing）。

1）图书表

图书表存储图书的相关信息，包括字段：图书id、ISBN号、名称、封面、作者、出版社、摘要、总数、可借阅数、贡献者。图书表结构如表5.1所示。

表5.1图书表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 是否可空 | 说明 |
| id | integer |  | 否 | 图书id，主键，自增 |
| ISBN | varchar | 255 | 否 | ISBN号 |
| title | varchar | 255 | 否 | 名称 |
| cover | varchar | 255 | 是 | 封面 |
| author | varchar | 255 | 是 | 作者 |
| press | varchar | 255 | 是 | 出版社 |
| excerpt | text |  |  | 摘要 |
| total\_count | integer |  | 否 | 总数 |
| avail\_count | integer |  | 否 | 可借阅数 |
| owner\_id | integer |  | 否 | 贡献者 |

2）用户表

用户表存储系统用户的相关信息，包括字段：用户id、用户标识、用户名。用户表结构如表5.2所示。

表5.2用户表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 是否可空 | 说明 |
| id | integer |  |  | 用户id，主键，自增 |
| uuid | integer |  | 否 | 用户标识 |
| name | integer |  | 否 | 用户名 |

3）借阅表

借阅表存储用户借阅信息，包括字段：借阅id、图书id、用户id、借阅时间。借阅表结构如表5.3所示。

表5.3借阅表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 宽度 | 是否可空 | 说明 |
| id | integer |  |  | 借阅id，主键，自增 |
| book\_id | integer |  | 否 | 图书id |
| user\_id | integer |  | 否 | 用户id |
| borrow\_time | date |  | 否 | 借阅时间 |

# 6.图书借阅平台的实现

微信图书借阅平台是一个完整的B/S架构的实现，在服务器端通过Nginx监听用户请求并分发请求给StrongLoop进程管理器，StrongLoop创建多个slave进程维护Node.js实例来处理用户请求，Node.js作为中间层会可以构建在任何其他语言的后端之上，通过基于RESTful规则设计的接口可以建立Node.js中间层与其他后端语言的数据通信。数据库采用MySQL、通过Jade模板引擎实现服务端HTML预渲染、在客户端通过微信内置的Blink内核的WebView作为与用户交互的入口、UI效果展示采用Bootstrap、CSS界面库快速制作移动端友好的响应式界面。

## 6.1配置

微信图书借阅平台是一个微信公众号，用户使用之前需要先关注公众号。用户登录微信，点击右上角的添加按钮，点击添加朋友菜单，输入“微信图书借阅平台”，关注公众号。成功关注公众号以后，用户便可以正常使用平台的各项功能。

## 6.2主界面

主界面是微信图书借阅平台的功能入口界面，提供借阅图书、归还图书、管理图书等三个一级菜单，其中管理图书分为添加图书、浏览图书两个二级菜单。主界面如图6.1所示。



图6.1主界面

各个功能大概是怎么实现的。

## 6.2添加图书

微信图书借阅平台是一个提供用户共享书籍的平台，用户有闲时不看的图书便可以自己的图书添加到公共书架上。在平台上，即点击图书管理下的二级菜单添加图书，便进入“扫一扫”界面。通过扫码图书，平台读取图书ISBN号，获取图书信息，最后跳转到贡献图书确认界面。

添加图书菜单按钮提供了用户添加图书的功能入口，点击添加图书，便进入扫一扫界面，可收集图书的信息，如图6.2所示。



图6.2扫一扫ISBN

获取图书ISBN号后，经过处理，获取图书信息并展示出来，用户可以对图书贡献进行确认，如图6.3所示。



图6.3添加图书

## 6.3借阅图书

微信图书借阅平台是一个提供用户共享书籍的平台，用户可以通过扫码借阅公共书架上的书籍。在平台上，即点击借阅图书菜单，进入“扫一扫”界面。通过扫码图书，平台获取图书ISBN号，获取图书信息，最后跳转到借阅图书确认界面。

借阅图书界面显示当前书籍信息（若要凑字，可以多描述一下），图书总量、已借阅、剩余等信息，如果当前书架上有可以借阅的书籍，确认借阅按钮可用，如图6.4所示；否则，确认按钮不可用，如图6.5所示。

 

图6.4借阅图书（可借）图6.5借阅图书（不可借）

## 6.4归还图书

微信图书借阅平台是一个提供用户共享书籍的平台，用户阅读完书籍可以通过扫码图书，归还公共书架上的书籍。在平台上，即点击归还图书菜单，进入“扫一扫”界面。通过扫码图书，平台获取图书ISBN号，获取图书信息，最后跳转到归还图书确认界面。

归还图书菜单按钮提供了用户归还图书的功能入口，点击归还图书，便进入扫一扫界面，可收集图书的信息。点击确认归还按钮，可以归还图书，如图6.6所示。



图6.6归还图书

## 6.5浏览图书

微信图书借阅平台是一个提供用户共享书籍的平台，用户可以添加书籍到公共书架、借阅书籍、归还书籍、浏览书籍。浏览书籍功能菜单直接跳转到书籍列表，展示当前书架上都有哪些书籍，点击书记项进入书籍详情页，查看书籍详细信息。

在平台上，点击浏览图书菜单，进入图书列表界面，如图6.7所示。通过点击书籍项目，进入详情页。详情页展示书籍详细信息，包括书籍名称、作者、图书ISBN号等书籍信息，还包括书籍当前借阅情况，例如总量、已借阅、剩余，如图6.8所示。

 

图6.7图书列表图6.8图书详情

结论

该毕业设计历经几个月时间，在这段时间首先确定了题目，查询了相关资料，经过需求分析、总体设计、系统功能设计及模块划分、系统数据库设计、编码实现与测试等过程，最终完成了河北科技大学网上教学评估系统设计。

本次毕业设计基本完成了所需的功能，实现了。。。。功能。基于作者的设计水平的有限性和经验的不完备，有部分功能不尽完美，如：。。。。这些都是我未来学习及工作中应该学习的地方。

参考文献

1

2