

湖 南 科 技 大 学

毕 业 设 计（ 论 文 ）

题 目	基于 WEB 平台的阅读 APP 设计与实现
作 者	曹慧琳
学 院	计算机科学与工程
专 业	物联网工程
学 号	1305040226
指导教师	符琦

二〇一七年 五月 二十七日

湖南科技大学

毕业设计（论文）任务书

____ 计算机科学与工程 ____ 院 ____ 物联网工程 ____ 系（教研室）

系（教研室）主任：____（签名） ____ 年 ____ 月 ____ 日

学生姓名：____ 曹慧琳 ____ 学号：____ 1305040226 ____ 专业：____ 物联网工程 ____

1 设计（论文）题目及专题：____ 基于 WEB 平台的阅读 APP 设计与实现 ____

2 学生设计（论文）时间：自 ____ 2016 ____ 年 ____ 12 ____ 月 ____ 30 ____ 日开始至 ____ 2017 ____ 年 ____ 5 ____ 月 ____ 20 ____ 日止

3 设计（论文）所用资源和参考资料：

[1]Remo, H, Jansen. Learning TypeScript[M]. The United States:Packt Publishing, 2015.

[2]张轩、杨寒星. React 全栈：Redux+Flux+webpack+Babel 整合开发[M]. 北京：电子工业出版社，2016.

[3]许式伟. Go 语言编程(图灵原创 6) [M]. 北京：人民邮电出版社，2012.

4 设计（论文）应完成的主要内容：

1) 学习基于 React+Redux 的单页应用的搭建

2) 学习基于 Go 语言的 RESTful API 的实现原理

3) 设计并完成阅读 App 应用

4) 完成 App 的部署和测试

5 提交设计（论文）形式（设计说明与图纸或论文等）及要求：

1. 论文思路清晰，条理清楚，目的明确，内容充实，格式规范，层次分明，图表清晰，结论正确；按《湖南科技大学本科生毕业设计（论文）要求与规范》编排、打印。

2. 上交论文（纸质）打印文稿和系统源码（电子版）。

6 发题时间：____ 2016 ____ 年 ____ 12 ____ 月 ____ 30 ____ 日

指导教师：____（签名）

学 生：____（签名）

湖 南 科 技 大 学

毕业设计（论文）指导人评语

指导人： (签名)

年 月 日

指导人评定成绩： _____

湖 南 科 技 大 学

毕业设计（论文）评阅人评语

评阅人： (签名)

年 月 日

评阅人评定成绩： _____

湖 南 科 技 大 学

毕业设计（论文）答辩记录

日期： 2017 年 6 月 4 日

学生： 曹慧琳 学号： 1305040226 班级： 物联网工程二班

题目： 基于 WEB 平台的阅读 APP 设计与实现

提交毕业设计（论文）答辩委员会下列材料：

- 1 设计（论文）说明书 共 57 页
- 2 设计（论文）图 纸 共 0 页
- 3 指导人、评阅人评语 共 2 页

毕业设计（论文）答辩委员会评语：

答辩委员会主任：	（签名）
委员：	（签名）
	（签名）
	（签名）
	（签名）

答辩成绩： _____

总评成绩： _____

摘 要

书籍是承载着人类文明和智慧为载体，阅读自古以来都是中华民族的优良传统。在全面建设小康社会的今天，智能手机已经变得必不可少，手机阅读也逐渐成为了人们学习、工作、生活中的一部分。本文介绍了一款基于 WEB 平台的阅读 APP 的设计与实现，该 APP 采用简洁护眼颜色搭配，简洁舒适的设计风格，集合了现有市场 APP 的众多功能，采用 React、Redux、Webpack、Babel 等最新的 WEB 开发技术，拥有流畅真实的翻书动画、自动记录阅读历史、多平台同步、收藏和分享图书等功能，同时支持智能分析用户阅读和搜索历史，并自动匹配、推荐用户可能喜爱的书籍，给用户一种全新的阅读体验。

关键词：图书阅读；WEB；手机移动应用；智能推荐

ABSTRACT

Books are the carrier of human civilization and wisdom. Reading has been a fine tradition of the Chinese race since ancient times. With the rapid development of science and technology and the widespread of smart mobile phones, mobile phone has gradually entered people's life, and online reading has become a new way of reading. This paper introduces design and implementation of a reading APP, which is based on the WEB platform. The App is styled in minimalist style by using comfortable color and simple page layout. It achieves a number of functions which are in existing market. React, Redux, Webpack, Babel and WEB development of the latest technology are used. It also achieves smooth animation about reading, self-recording reading history, multi-platform synchronization, collection and sharing of books and other functions. Also, intelligent analysis for users' reading and searching history, automatic book recommendation are considered. This App gives users a new reading experience.

Keywords: Book Reading; WEB; Mobile Phone Application; Intelligent Recommendation

目 录

第一章 前言	1 -
第二章 开发工具及技术	2 -
2.1 Visual Studio Code	2 -
2.2 TypeScript	2 -
2.3 React	2 -
2.4 Redux	2 -
2.5 Go 语言	3 -
2.6 MongoDB	3 -
2.7 REST API	3 -
第三章 需求分析	5 -
3.1 客户端功能需求	5 -
3.1.1 登录注册功能	6 -
3.1.2 图书搜索功能	6 -
3.1.3 图书分类功能	6 -
3.1.4 图书阅读功能	6 -
3.1.5 图书收藏功能	7 -
3.1.6 图书分享功能	7 -
3.1.7 图书推荐功能	7 -
3.1.8 阅读历史查看功能	7 -
3.2 服务端功能需求	7 -
3.3 性能需求	9 -
3.4 运行需求	9 -
第四章 数据库设计	10 -
4.1 数据库的需求分析	10 -
4.2 数据库的概念设计	10 -
4.3 数据库的逻辑设计	11 -
表 4.3.1 Tag 的逻辑设计表	11 -
表 4.3.2 Book 的逻辑设计表	12 -
表 4.3.3 Catalogue 的逻辑设计表	12 -
表 4.3.4 Content 的逻辑设计表	13 -
表 4.3.5 User 的逻辑设计表	13 -
表 4.3.6 Collect 的逻辑设计表	13 -
表 4.3.7 Recommend 的逻辑设计表	14 -
表 4.3.8 History 的逻辑设计表	14 -
第五章 详细设计	15 -
5.1 后端 API 接口设计	15 -
5.1.1 公共 API 接口设置	15 -
5.1.2 用户权限 API 接口设计	17 -
5.2 前端架构设计	21 -
5.3 前端交互设计	21 -

5.3.1 路由设计	21 -
5.3.2 主界面布局设计	21 -
5.3.3 主界面交互设计	22 -
5.3.4 图书列表布局设计	22 -
5.3.5 搜索框设计	23 -
5.3.6 登录界面交互设计	23 -
5.3.7 注册页面布局设计	24 -
5.3.8 注册界面交互设计	25 -
5.3.9 图书信息页面布局设计	25 -
5.3.10 阅读图书布局设计	25 -
5.3.11 阅读图书页面交互设计	26 -
5.3.12 收藏页面布局设计	26 -
5.3.13 推荐页面布局设计	26 -
5.3.14 用户信息页布局设计	27 -
第六章 系统实现	28 -
6.1 后端 API 接口的实现	28 -
6.1.1 公共 API 接口实现	28 -
(1) 获取图书所有分类的 API 实现	28 -
(2) 获取特定分类下的图书列表的 API 实现	29 -
(3) 获取特定图书的详细信息的 API 实现	29 -
(4) 获取图书特定章节的图书内容的 API 实现	29 -
(5) 获取特定关键词搜索图书列表的 API 实现	29 -
6.1.2 用户权限 API 接口设计	29 -
(1) 注册的 API 实现	29 -
(2) 登录的 API 实现	30 -
(3) 获取用户信息的 API 实现	31 -
(4) 收藏和取消收藏图书的 API 实现	31 -
(6) 获取推荐图书列表的 API 实现	32 -
(7) 上传用户读书记录的 API 实现	33 -
(8) 获取用户特定图书的最近记录的 API 实现	33 -
(9) 修改用户名的 API 实现	33 -
(10) 修改用户登录密码的 API 实现	33 -
(11) 获取短信或邮箱验证码的 API 实现	34 -
(12) 验证短信或邮箱验证码的 API 实现	34 -
6.2 前端组件的实现	35 -
(1) Router 组件的实现	35 -
(2) Home 组件的实现	36 -
(3) Header 组件的实现	37 -
(4) Footer 组件的实现	38 -
(5) SerchInput 组件的实现	39 -
(6) CategoryList 组件的实现	40 -
(7) RemindPopup 组件的实现	41 -
(8) BookList 组件的实现	41 -
(9) Loading 组件的实现	42 -

（10）Share 组件的实现	- 42 -
（11）BookContent 组件的实现	- 43 -
（12）BookContentHeader 组件的实现	- 44 -
（13）BookContentFooter 组件的实现	- 45 -
（14）BookItem 组件的实现	- 46 -
（15）FlipAnimation 组件的实现	- 47 -
（17）Battery 组件的实现	- 48 -
（18）Article 组件的实现	- 49 -
（19）LoginForm 组件的实现	- 50 -
（20）RegisterForm 组件的实现	- 51 -
（21）Collection 组件的实现	- 52 -
（22）User 组件的实现	- 53 -
第七章 总结	- 54 -
参考文献	- 56 -
致谢	- 57 -

第一章 前言

千年的风霜磨砺掉的是君权人治的封建旧制，磨不灭的是读书启智的文化底蕴。古有悬梁刺股，凿壁借光，囊萤映雪之典故，近代有周总理“为中华之崛起而读书”的豪情，如今更有许许多多读书人用成功诠释着读书的魅力。勤于读书，善于读书，不仅能够增加你的知识储备，职业技能，还能提高你的精神气质，言谈举止之间透露出那种文雅脱俗，不卑不亢的境界。

目前，中国经济已经发展到较高水平，综合实力显著提升。但文化软实力明显滞后甚至脱离了经济的发展步伐。调查显示，我国国民阅读的时间在逐步下降，人们更多的是为了考试、证书、养家糊口而读书，现代人生活压力越来越大，却忘了读书真正的意义。在全面实现小康社会的新时代，继承传统的读书文化，发扬中华民族的民族精神将会是我们面临的新的挑战。在发扬中国传统的读书文化，寻找新的读书方式的过程中，很多人把目光投向了智能手机。当今社会，生活节奏越来越快、阅读时间越来越碎片化，使用手机阅读开始成为主流，手机新闻，手机电子书，微信公众号等手机阅读方式也开始融入人们的生活。这种方便，快捷，海量的读书方式也逐渐得到大家的认可。当然如今的手机阅读还存在着大量的问题，阅读的体验，长时间对身体的消耗，经典阅读微乎其微。

本项目在阅读数据采集上，该项目以传统文化为核心，更多的去承载经典的文学作品；在设计上，采用简洁的设计风格，给读者一种清新的阅读心境；在阅读方式上，项目给予用户更真实的读书体验，智能适应用户的读书习惯；在技术上，采用全新的 WEB 技术，实现了多平台同步使用，同步更新；在资源利用上，实现了按需加载，更多的从用户的角度考虑，节省内存，节省流量。这种全新的阅读体验将会为全民阅读的新时代贡献出不可或缺的力量。

第二章 开发工具及技术

2.1 Visual Studio Code

Visual Studio Code 是微软在 Build 2015 大会上发布的一款免费的现代化轻量级代码编辑器，对 Go 语言开发和前端开发有非常好的插件支持，提供了 Git、代码补全、代码重构、代码调试等高级功能，支持 Linux, OS X, and Windows 等多个平台，可以说是程序员的开发利器。

Visual Studio Code 对开发本项目所使用到的 TypeScript 和 Go 语言有很好的支持，可以在一个编辑器中通过编写和调试前后端的代码而无需安装多个 IDE 以及在多个 IDE 中切换，大大提高了开发效率。

2.2 TypeScript

TypeScript 是一门开源的编程语言，由微软开发，可以看做是添加了静态类型和面向对象支持的 JavaScript。TypeScript 完美的兼容了 ES6, ES7 的新特性，可编译成可读的，标准的 JavaScript。TypeScript 支持几乎所有的浏览器和操作系统并且免费开源给大家使用。

和 JavaScript 相比，Typescript 编写的代码由于带有类型信息，更为健壮更具有可维护性。同时编辑器可以利用 Typescript 的类型信息提供更好的代码补全、代码检查、代码重构功能，有助于提供工作效率。

2.3 React

React，一个颠覆前端 UI 开发的框架。React 是由 Facebook 开发的用于开发数据不断变化的大型应用程序的 UI 框架。React 采用了 Virtual DOM 的新思想，从根本上解决了频繁 DOM 操作的性能瓶颈问题。React 采用组件化的构建思想，使用 React，你唯一需要关心的就是编写组件，组件具有良好的封装性，可以简化代码的复用、测试和协作开发流程。基于组件的开发让项目内部组件可以很方便的复用，也可以很方便的从 github 等网站寻找第三方组件来满足自己需求，提高了代码重用率和开发效率。

本项目着重于手机端的使用，页面不重复刷新，流量的合理控制，代码的简洁重复使用成为了很重要的问题，选用 React 框架可以说几乎完美的解决这些问题。

2.4 Redux

Redux 是 Facebook Flux 架构的一种简化实现，它是前端数据的一个状态容器，提供了可预测的状态管理。Redux 采用了 reduce 和纯函数的概念，每个新的 state 都是由旧的 state 在 action 的作用下生成的。state 集中在单一树状对象

上，即 store，是整个项目的数据层。Redux 通过这样统一的数据管理方式有效的解决了 React 在数据处理方面比较混乱的问题。

Redux 自带了非常优秀的调试工具，可以看到应用的历史状态和所有的状态变更，支持像视频播放器一样对应用的快进快退操作，支持代码热加载，修改代码的同时就可以看到新代码生效的结果，而不需要重新刷新页面，大大提高了前端开发效率。

本项目采用了 Redux-Immutable 的形式实现前端的数据管理，Immutable 内部使用了 Trie 数据结构来存储，两个对象的区别取决于 hashCode 的值，这给 React 的性能带来的很大的提升。

2.5 Go 语言

Go 语言于 2007 年诞生，2009 年正式发布，如今已经有 10 年历史。Go 语言语法比大部分编程语言都更加简单更加灵活，它在易于开发、快速编译，高性能执行之间找到了最佳平衡。它的并发特性可以方便地用于高并发网络程序开发，可以很好的利用现代的多核处理器，同时没有 Java 那样繁琐的继承机制、基于接口的类型系统可以非常方便地开发模块化的系统。Go 语言有非常快的编译速度，同时自带垃圾回收，降低了程序员的心智负担，并且还支持程序运行时反射，在高并发网络编程方面起到了不可替代的作用。

本项目在后端服务器搭建和 API 接口实现方面均使用 Go 语言实现，使得服务部署，开发效率方面都得到了明显的提高。

2.6 MongoDB

MongoDB 是 NoSQL 中一个非常流行的分布式开源文档数据库，它使用 C++ 编写而成，有非常高的性能。Mongodb 属于 NoSQL 中的文档数据库这一类，数据库由一个个独立的 bson 文档组成，而不是像传统数据库一样的二维表格结构。对于存储在数据库里面的文档，我们可以方便的添加字段而无需像传统数据库一样需要经历缓慢的表结构变更过程。

本项目数据结构不算太复杂，系统设计也相对简单，数据源都采用 JSON 格式，使用 MongoDB 这个数据库，可减少多余的数据转换的步骤，便于快速使用和操作数据，是实现系统的不二之选。

2.7 REST API

REST, Representational State Transfer, 表示资源在网络中的状态转移。REST 是以资源核心，通过 URL 来标识不同的资源，用 HTTP 协议的 GET、POST、PUT、DELETE 分别来表示对资源的增删查改操作。随着近年来移动互联网的发展，客户端种类逐渐增加，前后端分离成为了迫切需求，RESTful API 使得 Web 端和 Server 统一 API 来传递数据和改变数据状态成为可能。Web，

iOS, Android 和第三方开发者可通过一套 API 来共同消费 Server 提供的服务。

本项目在需求分析之后拟定了一套 RESTful 规则，统一了对资源的操作流程，使得项目在开发过程中可以前后端同时进行，并解决了不同操作系统对于资源访问的兼容性问题。

第三章 需求分析

本章详细描述了本项目的详细需求，为整个项目开发和技术实现提供指南，同时确定项目的内容和范围，为评价和测试该项目提供依据。

下面将分别对该项目的功能模块进行需求分析的详细说明。

3.1 客户端功能需求

客户端系统包含游客和会员两种角色，游客用户拥有基本的图书阅读权限，会员则包含很多会员专享权限。

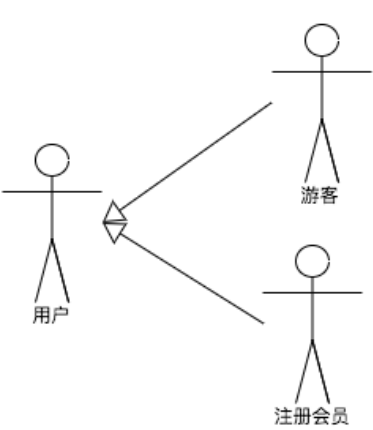


图 3.1.1 系统角色图

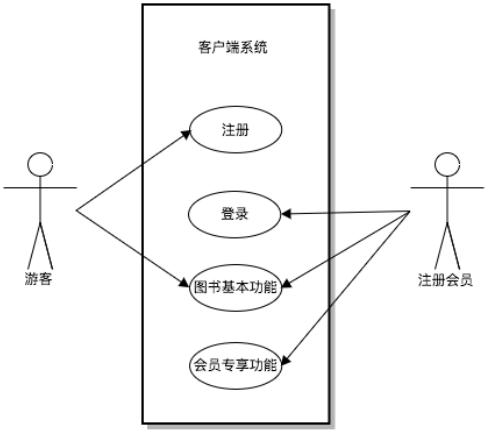


图 3.1.2 客户端系统用例图

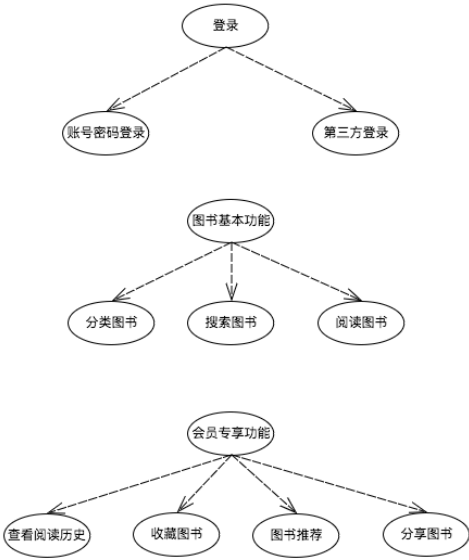


图 3.1.3 客户端系统用例分解图

3.1.1 登录注册功能

为了方便用户的使用，数据同步等，软件需要登录注册，该模块设计的主要需求为：

- （1）用户可以采用游客的方式试用软件，在没有登录时可以体验应用的部分功能。
- （2）用户可使用已经注册的账号密码登录，也可使用微信，微博，QQ 等第三方应用账号登录。
- （3）用户若没有账号，可进入注册页面进行注册。
- （4）用户首次注册成功后，将进行阅读兴趣的选择，阅读方式的配置等，将用户的配置信息存储到服务器进行同步。
- （5）用户可在 user 页面退出登录

3.1.2 图书搜索功能

用户可通过关键词搜索全站图书，该模块设计的主要需求为：

- （1）用户进入主页可在页面中看到搜索栏。
- （2）在搜索栏中输入关键词即可搜索想看的图书。
- （3）实时显示搜索结果。

3.1.3 图书分类功能

用户可筛选特定类型的书籍，该模块设计的主要需求为：

- （1）用户进入主页可看到图书的所有分类。
- （2）点击类型，显示对应类型的图书列表，列表中包含图书名称及介绍信息。
- （3）点击列表中图书，进入图书的详细介绍页。

3.1.4 图书阅读功能

用户可阅读选择的图书，该模块设计的主要需求为：

- （1）用户点击图书详细介绍页的去读或者目录栏的特定章节，即可进入阅读模式。
- （2）阅读模式主页显示对应章节的书的内容，并实现逼着的翻页动画。
- （3）页面左下角显示目前的读书进度。
- （4）页面的右下角显示当前时间和当前设备的电量，以及是否处于充电模式。
- （5）单击页面中的任何位置，弹出设置页面。

（6）设置页面可设置屏幕亮度，字体大小，白天夜间模式等。

3.1.5 图书收藏功能

用户可对自己喜欢的书籍进行收藏，该模块设计的主要需求为：

- （1）单击阅读页面,弹出收藏按钮。
- （2）如果用户喜欢该书籍，可点击收藏按钮进行收藏，收藏按钮变为已收藏按钮。
- （3）如果用户再次点击则取消收藏该图书。
- （4）用户可进入收藏页面，查看自己已经收藏图书的图书列表。

3.1.6 图书分享功能

用户可对自己喜欢的书籍进行分享，该模块设计的主要需求为：

- （1）单击阅读页面,弹出分享按钮。
- （2）如果用户喜欢该书籍，可点击分享按钮，弹出选择分享方式的页面。
- （3）单击分享方式即可分享对应书籍。

3.1.7 图书推荐功能

服务器根据用户的兴趣和读书种类，给用户推荐其可能喜爱的图书，该模块设计的主要需求为：

- （1）用户在进入收藏页面时，可在页面尾部看到系统推荐图书的图书列表。
- （2）点击感兴趣的图书进入图书详情页面。

3.1.8 阅读历史查看功能

用户可查看自己的阅读历史，该模块设计的主要需求为：

- （1）用户再次进入应用，可收到是否回到最新阅读的提示信息。
- （2）用户查看某本已经看过的书籍时，应用会提醒用户是否回到该书的上次阅读，用户点击是可直接进入该书的上次阅读的页面。
- （3）用户可在 User 页面查看读过书的数量及近期读书的时间。

3.2 服务端功能需求

服务器端主要给客户端提供 RESTful API 支持。下面针对具体服务进行需求上的详细说明。

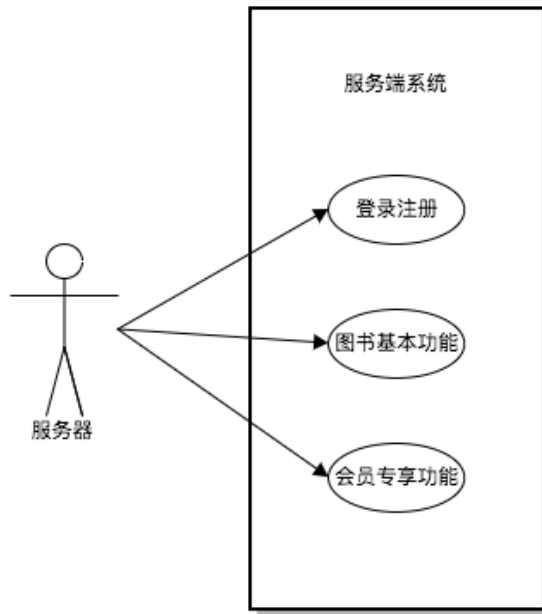


图 3.2.1 服务端系统用例图

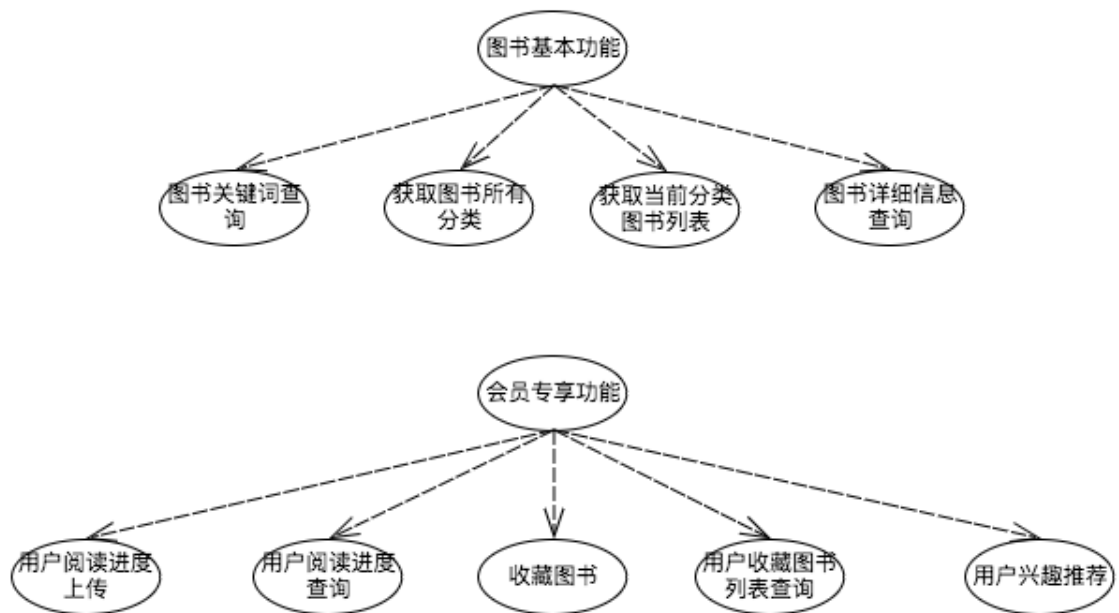


图 3.2.2 服务端系统用例分解图

- **用户登录注册服务：**用户登录注册服务主要响应于用户的请求，并将用户信息存于数据库。
- **图书关键词查询服务：**图书关键词查询服务主要响应客户端关键词查询请求。
- **获取所有分类服务：**获取所有分类服务主要响应客户端获取所有分类的请求。

- **获取当前分类图书列表服务：**获取当前分类图书列表服务主要响应客户端特定类型图书列表的请求，将数据库中属于该类型的图书列表返回给客户端。
- **图书详细信息查询服务：**图书详细信息查询服务主要响应客户端获取特定图书详细信息的请求。
- **用户阅读进度上传服务：**用户阅读进度上传服务主要响应客户端存储用户进度的请求，将用户阅读进度及历史的相关信息存储到数据库中。
- **用户阅读进度查询服务：**用户阅读进度查询服务主要响应客户端获取用户阅读进度的请求，从数据库中读取用户最近阅读进度，返回给客户端。
- **收藏图书服务：**用户阅读进度上传服务主要响应用户收藏图书的请求。
- **用户收藏图书查询服务：**用户收藏图书查询服务主要响应客户端获取用户收藏图书列表的请求，从数据库中获取用户收藏图书列表，返回给客户端。
- **用户兴趣推荐服务：**用户兴趣推荐服务主要根据用户阅读历史，收藏数据等信息进行用户行为分析，生成推荐列表，向客户端推送。

3.3 性能需求

（1）数据准确：图书信息，用户信息，用户收藏等 API 必须保证准确返回，保证良好的用户体验。

（2）响应时间：图书信息，用户信息，用户收藏等 API 保证在 500ms 之内返回。

（3）响应信息准确详细：用户请求 API 时尽可能返回详细的错误信息。

3.4 运行需求

（1）客户端支持主流的 Android 手机，支持 Android4.0 以上、iOS 8 以上系统。

（2）在用户没有网络或者网络极差的条件下，客户端需保证基本功能可以流畅使用。

第四章 数据库设计

4.1 数据库的需求分析

通过对项目的需求分析，数据库数据存储的基本信息如下：

表 4.1 数据库存储信息表

存储信息	字段
图书分类信息	类型 ID、类型名称、数目
图书详细信息	ID、名称、作者、介绍、封面、章节列表（章节 ID、章节名称）、类别
图书内容	图书 ID，章节 ID，图书内容
用户信息	用户 ID、用户昵称、账号、密码、阅读详情（今日读书时间、累计读书时间、累计读书数目）
收藏图书信息	用户 ID，收藏图书列表（图书 ID，分类标签）
推荐图书信息	用户 ID，推荐图书列表（图书 ID）
历史记录信息	用户 ID，图书 ID，章节 ID，页数，历史创建时间

4.2 数据库的概念设计

根据数据库需求分析，数据库应该包含八个主要的数据实体：

数据库名称	字段
Tag	tagId, category, booksCnt
Book	bookId, name, author, introduce, picture, catalogueId, tagId
Catalogue	categoryId, bookId, chapter
Content	bookId, categoryId, text
User	userId, nickName, account, password, readStatus

Collect	userId, collectList
Recommend	userId, recommendList
History	userId, bookId, categoryId, page, time

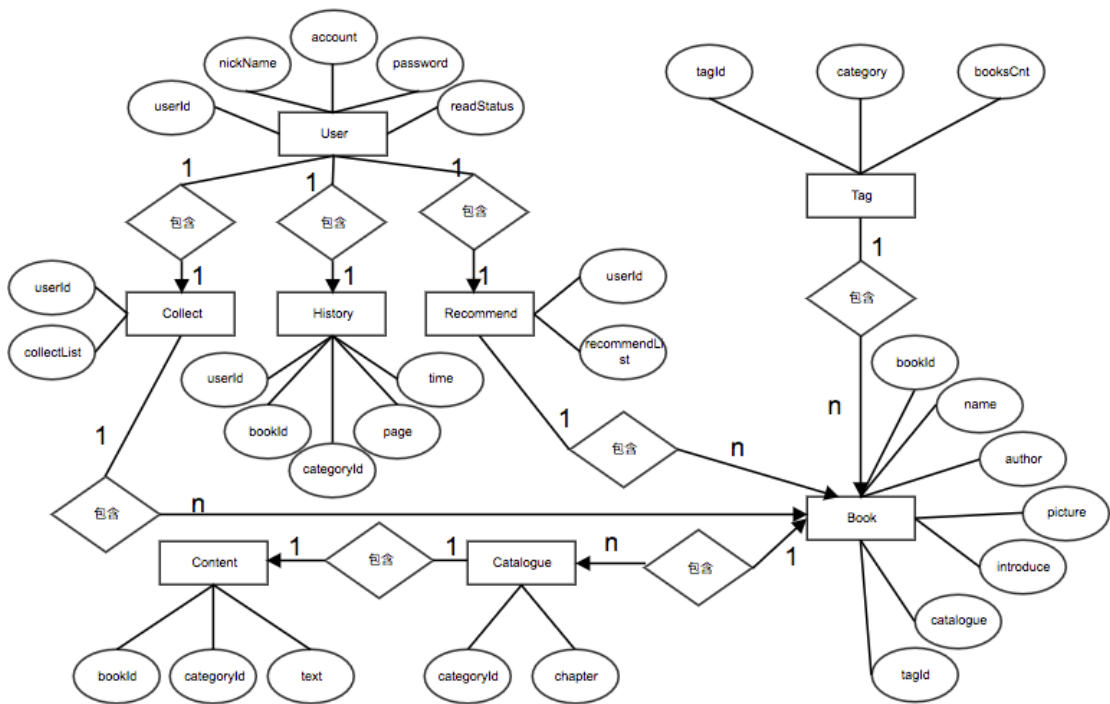


图 4.1 数据库实体关系图

4.3 数据库的逻辑设计

表 4.3.1 Tag 的逻辑设计表

序号	字段名	数据类型	
1	tagId	string	主键
2	category	string	
3	booksCnt	number	

表 4.3.1 描述图书类别信息的数据表，其中 tagId 为主键，category 为类别名称，bookCnt 为当前该类别所有图书的数量，每一次后端更新或者添加图书时，更新数据表。

表 4.3.2 Book 的逻辑设计表

序号	字段名	数据类型	
1	bookId	string	主键
2	name	string	
3	author	string	
4	introduce	string	
5	picture	string	
6	catalogue	引用数组	
7	tagId	string	外键

表 4.3.2 描述图书信息的数据表，其中 bookId 为主键，tagId 为所属分类的外键，catalogue 为该书所有章节的列表，书籍更新到数据库后信息基本不更新。

表 4.3.3 Catalogue 的逻辑设计表

序号	字段名	数据类型	
1	catalogueId	string	主键
2	bookId	string	外键
3	chapter	string	

表 4.3.3 描述章节具体信息的数据表。其中 catalogueId 为主键，chapter 为章节标题。该数据为图书 catalogueList 的元素。

表 4.3.4 Content 的逻辑设计表

序号	字段名	数据类型	
1	bookId	string	外键
2	categoryId	string	外键
3	text	引用数组	

表 4.3.4 描述书籍详细内容的数据库表。其中 bookId, categoryId 为主键。text 为文章内容的数组，按段落分组。

表 4.3.5 User 的逻辑设计表

序号	字段名	数据类型	
1	userId	string	主键
2	nickName	string	
3	account	string	
4	password	string	
5	readStatus	引用数组	

表 4.3.5 描述用户详细信息的数据表。其中 userId 为主键，readStatus 为用户历史阅读状态的统计，包括今日读书时间，累计读书时间和累计读书的数量，每日读书时间每晚 24 点清零。

表 4.3.6 Collect 的逻辑设计表

序号	字段名	数据类型	
1	userId	string	主键，外键
2	collectList	引用数组	

表 4.3.6 描述用户收藏列表的数据表，其中 `userId` 为主键，`collectList` 为用户收藏图书列表，用户收藏图书时，会把图书对应的 `bookId` 存储到 `collectList` 中。

表 4.3.7 Recommend 的逻辑设计表

序号	字段名	数据类型	
1	<code>userId</code>	<code>string</code>	主键，外键
2	<code>recommendList</code>	引用数组	

表 4.3.7 描述用户推荐列表的数据表，其中 `userId` 为主键，系统会定时进行用户行为分析，得出可能用户感兴趣的图书列表，将图书列表的 `id` 存储到 `recommendList` 中。

表 4.3.8 History 的逻辑设计表

序号	字段名	数据类型	
1	<code>userId</code>	<code>string</code>	主键，外键
2	<code>bookId</code>	<code>string</code>	外键
3	<code>categoryId</code>	<code>string</code>	外键
4	<code>page</code>	<code>string</code>	

表 4.3.8 描述用户阅读历史的数据表。其中 `userId` 为主键，用户阅读图书时会不断上传当前读书的状态，将这些状态记录到该表中，用于分析用户行为和提醒用户快速回到上次阅读的状态。

第五章 详细设计

本章将介绍该项目后端 API 接口和前端页面流程的详细设计过程。

5.1 后端 API 接口设计

后端 API 分为公共 API 和用户权限的 API，用户权限 API 只有在用户登录成功之后才可发送成功，否则为无效请求。这种分离保证用户在没有登录或者以游客的身份进入应用时仍可正常使用该应用。下面对 API 的设计进行详细介绍。

5.1.1 公共 API 接口设置

（1）获取图书所有分类的 API

Request: GET /tags

Response: 200 OK

```
{
  "tags": [
    {
      "id": "string",
      "category": "string",
      "books_cnt": "number"
    }
  ]
}
```

注意：books_cnt 为当前该分类图书中书的数目，方便前端用户用户排序展示。

（2）获取特定分类下的图书列表的 API

Request: GET /tags/:tagId

Response: 200 OK

```
{
  "books": [
    {
      "id": "string",
      "name": "string",
      "picture": "string",
      "author": "string",
      "description": "string",
    }
  ]
}
```

```
    "category": "string"
  }
]
}
```

（3）获取特定图书的详细信息的 API

Request: GET /books/:bookId

Response: 200 OK

```
{
  "id": "string",
  "name": "string",
  "picture": "string",
  "author": "string",
  "description": "string",
  "category": "string",
  "catalog": [
    {
      "id": "string",
      "chapter": "string"
    }
  ]
}
```

（4）获取图书特定章节的图书内容的 API

Request: GET /books/:bookId/content/:categoryId

Response: 200 OK

```
{
  "contents": [
    "string"
  ]
}
```

（5）获取特定关键词搜索图书列表的 API

Request: GET /search/books?q={query}

Response: 200 OK

```
{
  "books": [
    {
```

```
    "id": "string",
    "name": "string",
    "picture": "string",
    "author": "string",
    "description": "string",
    "category": "string"
  }
]
```

5.1.2 用户权限 API 接口设计

（1）注册的 API

Request: POST /register

```
{
  "account": "string",
  "nickname": "string",
  "password": "string"
}
```

Response: 200 OK

```
{
  "jwt": "string"
}
```

（2）登录的 API

Request: POST /login

```
{
  "account": "string",
  "password": "string"
}
```

Response: 200 OK

```
{
  "jwt": "string"
}
```

（3）获取用户信息的 API

Request: GET /user/info

Response: 200 OK

```
{
  "username": "string",
  "nickname": "string",
  "readStatus": {
    "today": "number",
    "allTime": "number",
    "allBooks": "number"
  }
  "current": {
    "book_id": "string",
    "menu_id": "string",
    "page": "string",
    "has_history": true,
    "last_time": "string"
  }
}
```

（4）收藏和取消收藏图书的 API

Request: POST /user/collect

```
{
  "book_id": "string",
  "tag": "string",
  "action": "string"
}
```

Response: 200 OK

（5）获取收藏图书列表的 API

Request: GET /user/collect

Response: 200 OK

```
{
  "books": [
    {
      "id": "string",
      "name": "string",
      "picture": "string",
      "author": "string",
      "description": "string",

```

```
    "category": "string"
  }
],
"tag": [
  {
    "book_id": "string",
    "tag": "string"
  }
]
```

(6) 获取推荐图书列表的 API

Request: GET /user/recommend

Response: 200 OK

```
{
  "books": [
    {
      "id": "string",
      "name": "string",
      "picture": "string",
      "author": "string",
      "description": "string",
      "category": "string"
    }
  ]
}
```

(7) 上传用户读书记录的 API

Request: POST /user/status

```
{
  "book_id": "string",
  "menu_id": "string",
  "page": "string"
}
```

Response: 200 OK

(8) 获取用户特定图书的最近记录的 API

Request: GET /user/status/:bookId

Response: 200 OK

```
{
  "book_id": "string",
  "menu_id": "string",
  "page": "string",
  "has_history": "boolean",
  "last_time": "string"
}
```

(9) 修改用户名的 API

Request: POST /user/nickname

```
{
  "nickName": "string"
}
```

Response: 200 OK

(10) 修改用户登录密码的 API

Request: POST /user/password

```
{
  "password": "string"
}
```

Response: 200 OK

(11) 获取短信或邮箱验证码的 API

Request: GET /user/verifycode

Response: 200 OK

```
{
  "codeKey": "string"
}
```

(12) 验证短信或邮箱验证码的 API

Request: POST /user/verifycode

```
{
  "codeKey": "string",
  "code": "string"
}
```

Response: 200 OK

5.2 前端架构设计

本项目前端采用 React + Redux + Immutable + Webpack 的组合开发框架，React 用于构建前端页面，即 MVC 的 View 层。Redux 用于数据处理，实现了单项数据流，将项目中使用到的数据做集中化处理。Immutable 完美的实现了不可变数据的流程，使数据在使用中不能被随便修改，也优化了 React 页面刷新机制中存在的一些问题。Webpack 用于项目打包，将项目在生产环境和开发环境以不同的方式打包，将开发体验和用户体验都提升到最佳。

5.3 前端交互设计

5.3.1 路由设计

表 5.3 项目路由设计表

URL	模块	描述
/	Home	应用主界面
/home/:tagId	Tag	指定分类的图书列表
/home/book/:bookId	Book	指定图书的介绍信息
/home/book/:bookId/content/: categoryId	Content	指定图书对应章节的内容
/collection	Collection	用户收藏图书列表
/user	User	用户信息
/login	Login	登录页面
/register	Register	注册页面

5.3.2 主界面布局设计

用户进入软件首先看到的是主界面模块。

主界面模块由四部分组成

（1）导航栏：左侧显示 APP 的图标，在页面任何地方点击该图标可回到应用主页，右侧为用户登录状态显示，如果用户没有登录，显示登录按钮，如果登录成功，显示用户昵称。点击登录按钮，即可进入登录页面，点击用户昵称可进入用户信息页面。

（2）图书搜索框：图书全局搜索设计在页面内容的上方，显眼的位置可以使用户使用更加方便。

（3）图书列表：页面显示所有分类，分类采用显示一部分的方式。点击显示按钮，即可显示全部分类。点击感兴趣的分类，即可看到该分类下的图书列表。

（4）主菜单栏：书城、收藏，用户页面的切换按钮，方便用户在项目中快速找到自己需要的页面。



图 5.3.1 主界面布局原型设计

5.3.3 主界面交互设计

用户进入应用时会自动获取用户最近一次阅读的状态，并在页面下方提示用户。如果用户选择继续阅读，可直接跳到上次阅读的位置。



图 5.3.2 主界面交互设计

5.3.4 图书列表布局设计

主页图书列表采用上下排列，具体图书信息采用图片加简要介绍的方式。上下精密排列可使用户在进入软件时尽可能看到多的图书，图片加介绍的方式

更加形象的介绍图书的基本信息，让用户在第一次看到图书时，对图书的理解已经了然于胸。



图 5.3.3 图书列表布局原型设计

5.3.5 搜索框设计

搜索框采用横向沾满整个屏幕的设计，让搜索功能更加醒目。搜索采用实时搜索。



图 5.3.4 搜索框原型设计

5.3.6 登录界面交互设计

登录表单实现实时给用户提供错误信息，当输入框中输入的信息不符合对应的格式时，页面会随时提供给用户提示信息，直到用户输入正确，这样既可以减少请求次数，又可以提高用户体验。



图 5.3.5 登录界面布局设计



图 5.3.6 登录界面交互设计

5.3.7 注册页面布局设计

注册页面除了填写表单的方式，也提醒用户可用社交账号直接登录，这样既不用填写表单，又不需要记忆多个账号密码，实用，方便。



图 5.3.7 注册界面布局原型设计



图 5.3.8 注册界面交互设计

5.3.8 注册界面交互设计

注册表单也实现了实时给用户提示提示信息，同时在验证账户时，提供给用户倒计时发送验证码的方式，既减少用户多次 发送验证码造成服务器压力过大，又平和用户着急等待的心态。

5.3.9 图书信息页面布局设计

用户点击图书列表，即可进入图书详细页面。该页面分为三部分：

（1）图书基本信息：页面上部显示图书封面，作者，类型等信息以及一个醒目的去读按钮。

（2）图书简介：页面中间为该图书的简介，让用户简单了解该图书。

（3）图书目录：页面下部为图书目录页，用户可看到该图书的目录结构及标题，点击目录，即可进入对应目录的阅读页面，方便用户跳读。

5.3.10 阅读图书布局设计

用户点击去读获取目录即可进入阅读页面。

阅读页面由以下几部分组成

（1）标题：阅读页面的顶部为对应章节的标题，提醒用户当前所在的章节

（2）内容：中间为内容部分，显示书籍的详细信息

（3）信息展示：左侧为阅读进度，右侧为当前时间和设备的电量，如果设备处于充电模式，显示充电提醒。



图 5.3.9 图书详细页面布局原型设计

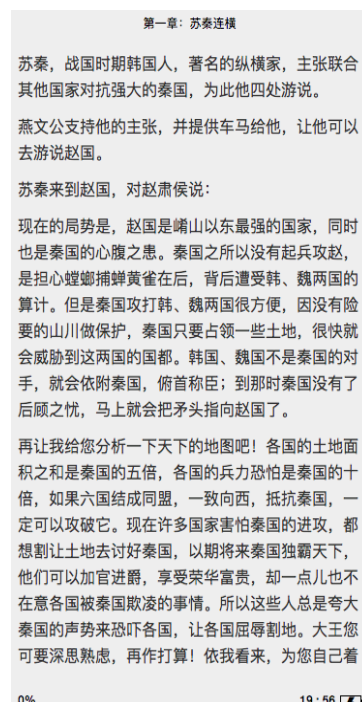


图 5.3.10 阅读图书网页布局设计

5.3.11 阅读图书页面交互设计

阅读页面实现流畅的手势翻页，跟随动画等功能，提高用户体验。

单击页面中部，可进入阅读设置模式，可设置屏幕亮度，字体大小，白天夜间模式等，让用户阅读更加舒服。

如果用户喜欢可点击页面右上角的收藏按钮收藏图书。也可点击分享按钮进行分享图书。

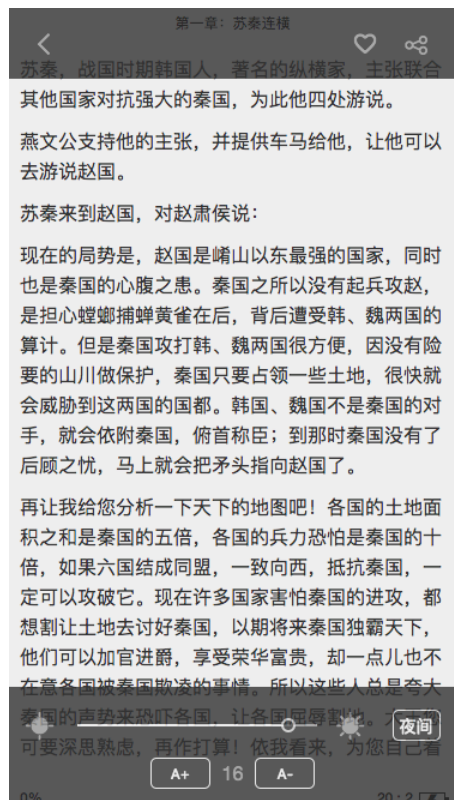


图 5.3.11 阅读图书页交互设计



图 5.3.12 分享布局设计

5.3.12 收藏页面布局设计

点击收藏按钮，即可进入收藏页面。

收藏页面采用封面显示书藏图书的方式。因为用户收藏图书，即用户已经充分了解该图书，不需要详细的图书介绍，只需通过封面，用户即可知图书信息。

点击图书可进入详细页面。

5.3.13 推荐页面布局设计

收藏页面下方为推荐图书列表。

推荐图书同样采用了封面列表的方式，和收藏列表保持相同的布局风格。



图 5.3.13 收藏推荐图书布局设计



图 5.3.1 用户信息界面布局设计

5.3.14 用户信息页布局设计

点击‘我’按钮，即可进入用户信息页面。用户信息页面分为三个部分：

- （1）基本用户信息：页面上部显示用户头像，用户昵称，用户等级等信息，以及用户累积读书历史的信息。
- （2）用户喜欢阅读的类型：根据用户读书历史和读书习惯，列出用户喜欢读书的类型标签。
- （3）用户喜欢的作者：根据用户读书历史和收藏列表，列出用户喜欢的作者。

点击右上角退出按钮，用户即可退出登录。

第六章 系统实现

6.1 后端 API 接口的实现

6.1.1 公共 API 接口实现

公共 API 接口实现基本流程为：

- a. 获取用户请求参数信息；
- b. 在数据库中查找相关数据；
- c. 将数据进行格式化整理；
- d. 将整理后的数据返回给用户。

下面将对具体 API 接口实现进行详细描述：

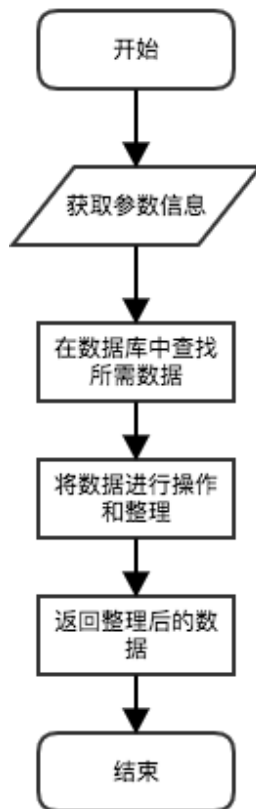


图 6.1.1 公共 API 接口实现流程

（1）获取图书所有分类的 API 实现

- a. 查找数据库中所有图书的类型；
- b. 将获取的数据进行格式化；
- c. 返回格式化后的数据。

（2）获取特定分类下的图书列表的 API 实现

- a. 获取请求参数中 tagId;
- b. 在数据库中查找该类型的图书;
- c. 将图书列表进行整理;
- d. 返回格式化后的图书列表。

（3）获取特定图书的详细信息的 API 实现

- a. 获取请求参数中的 bookId;
- b. 在数据库中查找该图书;
- c. 将图书信息进行整理;
- d. 返回格式化后的图书信息。

（4）获取图书特定章节的图书内容的 API 实现

- a. 获取请求参数中的 bookId 和 categoryId;
- b. 在数据库中找到对应图书;
- c. 在图书的章节列表中找到对应的章节;
- d. 获取对应章节的内容;
- e. 将内容信息进行整理;
- f. 返回整理后的图书内容。

（5）获取特定关键词搜索图书列表的 API 实现

- a. 获取请求参数中用户查找的关键词;
- b. 在数据库中查找含有该关键词的图书;
- c. 将找到的图书按照出现的次数进行排序;
- d. 返回排序后的图书列表。

6.1.2 用户权限 API 接口设计

（1）注册的 API 实现

注册 API 的实现流程:

- a. 获取用户注册参数信息;
- b. 判断参数格式是否正确;
- c. 如果参数格式错误, 执行下一步, 否则跳到步骤 e;
- d. 返回参数格式错误提示信息;
- e. 在数据库中查找该账户是否存在;

- f. 如果该账户存在，执行下一步，否则跳到步骤 h；
- g. 返回账户已经注册错误信息；
- h. 将用户密码加密；
- i. 将用户信息存入数据库；
- j. 计算 jwt；
- k. 返回 jwt。

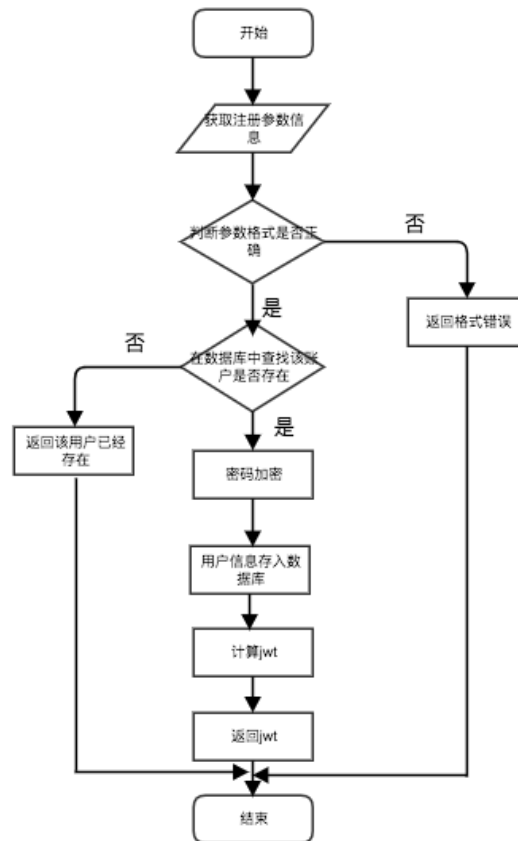


图 6.1.2 注册 API 接口实现流程图

(2) 登录的 API 实现

登录 API 的实现流程：

- a. 获取用户登录参数信息；
- b. 判断参数格式是否正确；
- c. 如果不正确，执行下一步，否则跳到步骤 e；
- d. 返回格式错误；
- e. 在数据库查找该用户是否存在；
- f. 如果用户不存在，执行下一步，否则跳到步骤 h；
- g. 返回用户不存在错误信息；
- h. 将登录密码进行加密；
- i. 判断密码是否相等；
- j. 如果不相等，执行下一步，否则，跳到步骤 l；
- k. 返回密码错误提示信息；
- l. 计算 jwt；
- m. 返回 jwt。

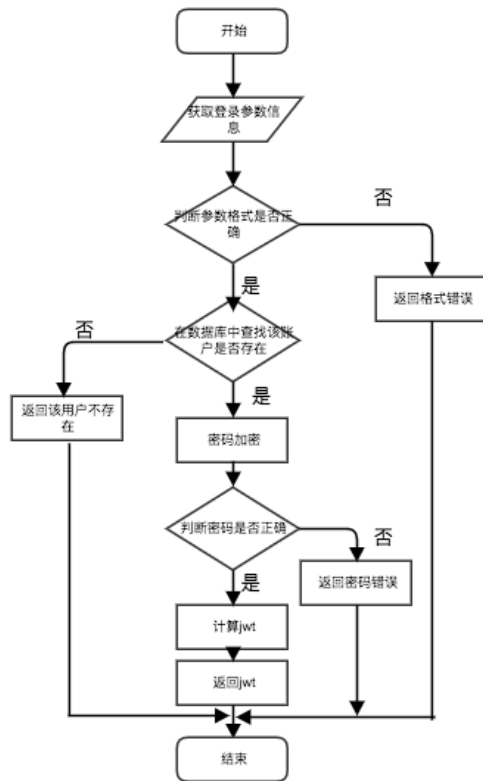


图 6.1.3 登录 API 接口实现流程图

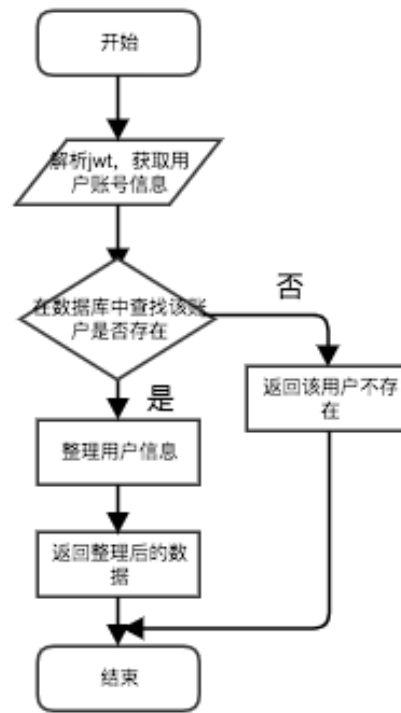


图 6.1.4 获取用户信息 API 接口实现流程图

（3）获取用户信息的 API 实现

获取用户信息 API 实现流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 整理用户信息；
- f. 返回整理后数据。

（4）收藏和取消收藏图书的 API 实现

收藏和取消收藏图书 API 实现流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 获取用户收藏 bookId；
- f. 在数据库中查找图书是否存在；
- g. 如果图书不存在，执行下一步，否则跳到步骤 i；
- h. 返回图书不存在提示信息；
- i. 判断用户操作是否为收藏；
- j. 如果为收藏，执行下一步，否则跳到步骤 l；
- k. 将 bookId 加入用户收藏列表；

- l. 将 bookId 从用户收藏列表删除；
- m. 返回操作成功。

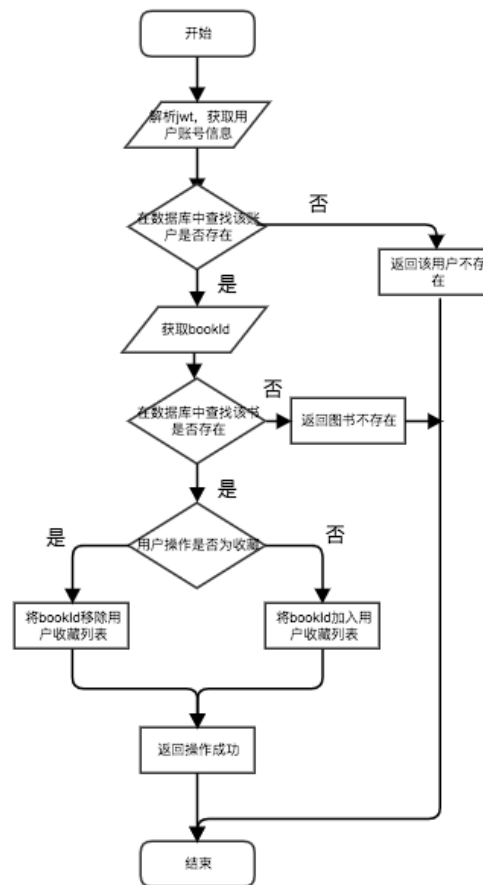


图 6.1.4 收藏和取消收藏图书 API 接口实现流程图

（5）获取收藏图书列表的 API 实现

获取收藏图书列表 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 查找并返回收藏列表。

（6）获取推荐图书列表的 API 实现

获取推荐图书列表 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 查找并返回推荐列表。

（7）上传用户读书记录的 API 实现

上传用户读书记录 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 在获取参数 bookId
- f. 在数据库中查找图书是否存在
- g. 如果不存在，执行下一步，否则，跳到步骤 i；
- h. 返回图书不存在
- i. 将读书信息和当前时间存入数据库
- j. 返回操作成功

（8）获取用户特定图书的最近记录的 API 实现

获取用户特定图书历史记录 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 在获取参数 bookId
- f. 在数据库中查找图书是否存在
- g. 如果不存在，执行下一步，否则，跳到步骤 i；
- h. 返回图书不存在
- i. 查找数据库获取该图书最近一条历史记录
- j. 返回操作成功

（9）修改用户名的 API 实现

修改用户名 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 获取参数 nickName
- f. 修改数据库中用户 nickName
- g. 返回操作成功

（10）修改用户登录密码的 API 实现

修改用户登录密码 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；

- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 获取参数 newPassword
- f. 修改数据库中用户 password
- g. 返回操作成功

（11）获取短信或邮箱验证码的 API 实现

获取短信或邮箱验证码 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 调用发送验证码服务
- f. 给用户发送验证码，并将验证 key 返回客户端

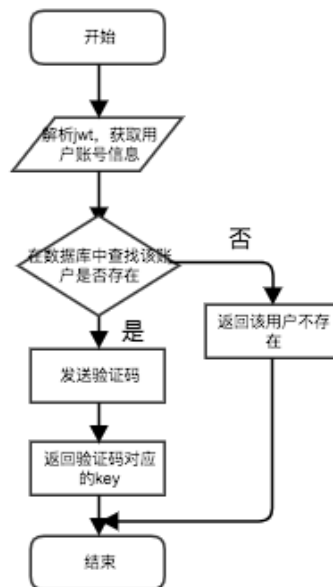


图 6.1.5 发送验证码 API 接口实现流程图

（12）验证短信或邮箱验证码的 API 实现

验证短信或邮箱验证码 API 流程：

- a. 解析 jwt，获取用户账户信息；
- b. 在数据库中查找用户是否存在；
- c. 如果不存在，执行下一步，否则跳到步骤 e；
- d. 返回用户不存在提示信息；
- e. 获取验证 key 和验证码
- f. 验证 key 与验证码是否对应
- g. 如果不对应，执行下一步，否则跳到步骤 i；
- h. 返回验证码错误
- i. 验证成功

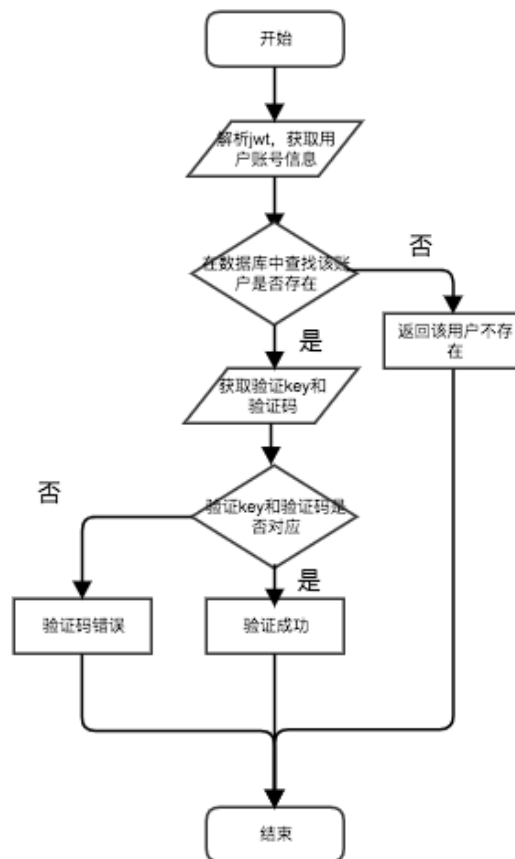


图 6.1.6 验证验证码 API 接口实现流程图

6.2 前端组件的实现

(1) Router 组件的实现

定义了整个应用的路由，使用了 react-router-redux 工具实现，绑定了整个应用的 store，与应用数据层关联起来。

```
export default React.createClass({
  render() {
    return (
      <Provider store={store}>
        <Router history={history}>
          <Route path="/" component={App}>
            <IndexRoute component={Home} />
            <Route path='home' component={Home} />
            <Route path='home/:id' component={Home} />
            <Route path='home/book/:id' component={Article} />
            <Route path='home/book/:id/content/:id'
              component={Content} />
            <Route path='collection' component={Collection} />
            <Route path='user' component={User} />
            <Route path='login' component={Login} />
            <Route path='register' component={Register} />
          </Route>
        </Router>
      </Provider>
    );
  }
});
```

图 6.2.1 Router 组件实现部分代码

(2) Home 组件的实现

Home 组件为应用主页的基本架构，当用户进入应用时，首先渲染该组件。它是由 Header、Footer、SearchInput、CategoryList、RemindPopup 组件组成的，将应用的主要功能和数据列表醒目的展示在用户面前。

```
class Home extends React.Component<IHomeProps, IHomeState> {
  constructor(props: IHomeProps) {
    super(props);
    this.state = {
      showRemind: false,
    };
  }
  handlePopup = () => {...};
  renderBookList() {...}
  componentWillReceiveProps(nextProps: IHomeProps) {...}
  render() {
    const { account } = this.props;
    const current = account.get('data') ?
      account.get('data').current : account.get('data').current : null : null;
    const bookId = current ? current.book_id : null;
    const menuId = current ? current.menu_id : null;
    return (
      <div className='app'>
        <Header />
        <div className='home'>
          <Search />
          <div className='content'>
            <CategoryList />
            {this.renderBookList()}
          </div>
        </div>
        {this.state.showRemind &&
          <RemindPopup menuId={menuId} bookId={bookId}
            handleClick={this.handlePopup} />
        <Footer />
      </div>
    );
  }
  componentDidMount() {...}
}
```

图 6.2.2 Home 组件实现部分代码

（3）Header 组件的实现

Header 组件为整个项目的 Header，header 组件的 logo 采用字体图标的方式，不仅使得加载图标速度加快，而且解决了图片加载失败影响用户体验的问题。右边显示用户的登录状态，当用户登录时，显示用户的用户名，否则显示登录按钮，提醒用户登录。用户在任何界面点击左上角的 logo 都将回到应用的主界面，点击登录按钮，将进入登录页面，点击用户名，进入用户信息详细介绍界面。将 Header 组件封装为单独的组件可以方便该组件在不同的页面公用，方便开发。

```
class Header extends React.Component<IHeaderProps, IHeaderState> {
  constructor(props: IHeaderProps) {
    super(props);
    this.loginOut = this.loginOut.bind(this);
  }
  componentDidMount() {
    if (cookie.load('bookmanage_jwt')) {
      this.props.AuthActions.getUserInfo();
    }
  }
  loginOut() {
    cookie.remove('bookmanage_jwt');
    this.props.AuthActions.deleteUserInfo();
  }
  render() {
    const { account, routing } = this.props;
    const pathname = routing.locationBeforeTransitions.pathname;
    return (
      <div className='header'>
        <div className='container'>
          <Link className='logo' to='/'>
            <i className='iconfont weui-tabbar__icon'>&#xe60a;</i>
            <p>书海</p>
          </Link>
          <div className='loginButton'>
            {account.get('data') ?
              (pathname === '/user' ?
                <a onClick={this.loginOut}>退出</a> :
                <Link to='/user'>{account.get('data').nickname}</Link>)
              :
                <Link to='/login'>登录</Link>}
          </div>
        </div>
      </div>
    );
  }
}
```

图 6.2.3 Header 组件实现部分代码

（4）Footer 组件的实现

Footer 组件为整个应用的主路由切换组件，即主菜单实现。将主菜单切换按钮放在页面醒目的位置可方便用户快速找到想要进入的页面。单独封装 Footer 组件和 header 组件作用一样，可多个页面共用同一个组件。

```
class Footer extends React.Component<IFooterProps, IFooterState> {
  render() {
    const { routing } = this.props;
    const path = routing.locationBeforeTransitions.pathname;
    return (
      <div className='weui-tabbar'>
        <Link to='home' className={ path.startsWith('/home') || path === '/'
          ?
          'weui-navbar__item weui-bar__item_on' : 'weui-navbar__item'}>
          <i className='iconfont weui-tabbar__icon'>&#xe605;</i>
          <p className='weui-tabbar__label'>书城</p>
        </Link>
        <Link to='collection' className={ path.startsWith('/collection')
          ?
          'weui-navbar__item weui-bar__item_on' : 'weui-navbar__item' }>
          <i className='iconfont weui-tabbar__icon'>&#xe601;</i>
          <p className='weui-tabbar__label'>收藏</p>
        </Link>
        <Link to='user' className={path.startsWith('/user')
          ?
          'weui-navbar__item weui-bar__item_on' : 'weui-navbar__item'}>
          <i className='iconfont weui-tabbar__icon'>&#xe600;</i>
          <p className='weui-tabbar__label'>我</p>
        </Link>
      </div>
    );
  }
}
```

图 6.2.4 Footer 组件实现部分代码

(5) SerchInput 组件的实现

SearchInput 组件实现了用户搜索图书实时请求，用户输入查询关键词时实时向后端请求查询对应关键词的图书信息，并将后端返回的结果实时响应，展示给用户。

```
class Search extends React.Component<ISearchProps, ISearchState> {
  constructor(props: ISearchProps, context: any) {
    super(props, context);
    this.state = {
      search: false,
      input: false
    };
  }
  handleBlur = () => {...};
  handleFocus = () => {...};
  handleInput = (e: any) => {...};
  render() {
    return (
      <div className='search'>
        <div className={this.state.search ?
          'weui-search-bar weui-search-bar_focusing' : 'weui-search-bar'}
          id='search_bar'>
          <xml/>
          <a href='javascript:'
            className='weui-search-bar__cancel-btn'
            id='search_cancel'>取消</a>
        </div>
        <div className={this.state.input ?
          'weui-cells weui-cells_access search_result_list search_show show':
          'weui-cells weui-cells_access search_result_list search_show hide'}
          id='search_show'>
          {
            this.props.search.get('list').map((item, i) => {
              if (i < 5) {
                return (
                  <div className='weui-cell' key={i}>
                    <div className='weui-cell__bd weui-cell_primary'>
                      <Link to={`home/book/${item.id}`}>{item.name}</Link>
                    </div>
                  </div>
                );
              }
            })
            return null;
          }
        </div>
      </div>
    );
  }
}
```

图 6.2.5 SerchInput 组件实现部分代码

（6）CategoryList 组件的实现

CategoryList 组件采用 map 的方式，将后端返回的图书类型一一展现在用户面前。该组件实现了按量加载，根据用户当前选择的分类和用户查看图书列表的位置，加载一部分图书列表，这样不会由于网络或者手机原因造成加载速度慢的问题，提高了用户体验。

```
class CategoryList extends React.Component {
  constructor(props: ICategoryListProps, context: any) {
    super(props, context);
    this.state = {
      id: '93',
      showAll: false,
    };
  }
  setId = (id: any) => {...};
  allcategoryShow = () => {...};
  renderCategoryList() {
    const { allcategory } = this.props;
    const number = this.state.showAll ? allcategory.size : 7;
    return (
      <ul>
        { allcategory.slice(0, number).map((item) => {
          return (
            <li key={item.id} >
              <Link to={'/home/' + item.id}
                className={this.state.id === item.id ? 'tagActive' : ''}
                onClick={this.setId.bind(null, item.id)}>
                {item.category}
              </Link>
            </li>
          );
        })
        <li onClick={this.allcategoryShow}
          style={{display: this.state.showAll ? 'none' : 'block'}}>
          <i className='iconfont weui-tabbar__icon'>&#xe602;</i>
        </li>
        <li className='pack' onClick={this.allcategoryShow}
          style={{display: this.state.showAll ? 'block' : 'none'}}>
          <i className='iconfont weui-tabbar__icon'>&#xe604;</i>
        </li>
      </ul>
    );
  }
  render() {
    return (
      <div className='category'>
        {this.renderCategoryList()}
      </div>
    );
  }
}
```

图 6.2.6 CategoryList 组件实现部分代码

(7) RemindPopup 组件的实现

RemindPopup 主要实现获取用户的历史状态，及时提醒用户回到上一次阅读状态。

```
class RemindPopup extends React.Component<any, any> {
  render() {
    const { bookId, menuId } = this.props;
    return (
      <div className='remind-popup'>
        <p>是否回到上次阅读? </p>
        <div className='button-group'>
          <a onClick={this.props.handleClick}>暂时不了</a>
          <Link onClick={this.props.handleClick}
            to={`/home/book/${bookId}/content/${menuId}`}>
            继续阅读
          </Link>
        </div>
      </div>
    );
  }
}
```

图 6.2.7 RemindPopup 组件实现部分代码

(8) BookList 组件的实现

BookList 同样采用 map 的方式，将对应分类的图书列表展示给用户。

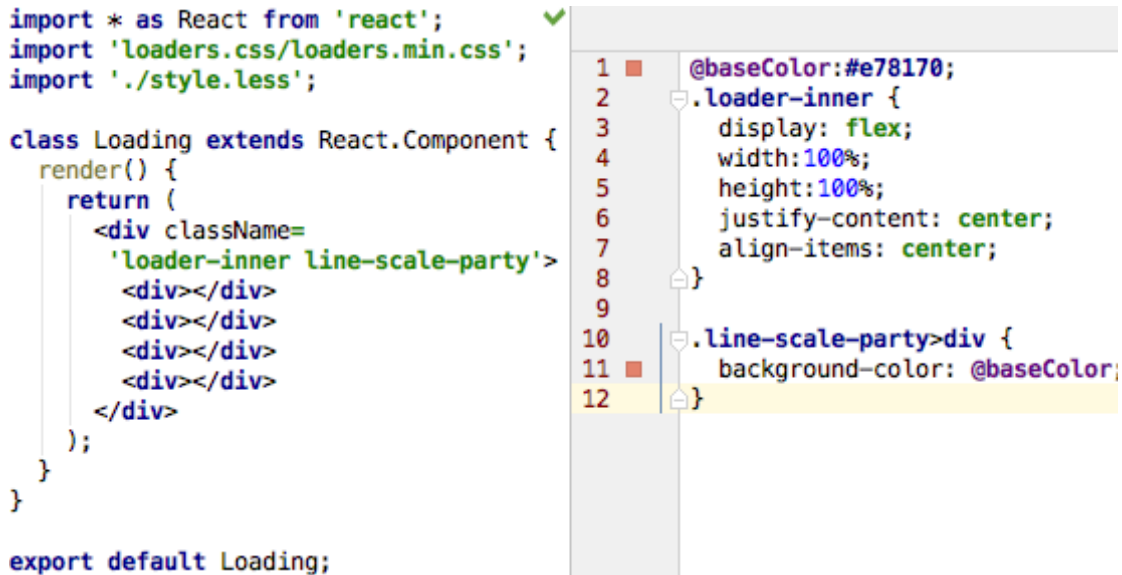
```
class BookList extends React.Component<IBookListProps, IBookListState> {
  renderBookList () {
    const { allbooklist } = this.props;
    return (
      <ul className='books'>
        {allbooklist.map((book) => {
          return(
            <Link key={book.id} to={'home/book/' + book.id} className='item'>
              <img src={book.picture} className='picture' />
              <div className='info'>
                <div className='title'>{book.name}</div>
                <div className='author'>{book.author}</div>
                <div className='introduce'>{book.description}</div>
              </div>
            </Link>
          );
        })}
      </ul>
    );
  }

  render() {
    return (
      <div className='book-list'>
        {this.renderBookList()}
      </div>
    );
  }
}
```

图 6.2.8 BookList 组件实现部分代码

（9）Loading 组件的实现

Loading 组件主要用户页面在请求 API，没有返回数据的时候显示，用户提醒用户需要稍等片刻，提高用户体验。



```
import * as React from 'react';
import 'loaders.css/loaders.min.css';
import './style.less';

class Loading extends React.Component {
  render() {
    return (
      <div className=
        'loader-inner line-scale-party'>
        <div></div>
        <div></div>
        <div></div>
        <div></div>
      </div>
    );
  }
}

export default Loading;
```

```
1 @baseColor: #e78170;
2 .loader-inner {
3   display: flex;
4   width: 100%;
5   height: 100%;
6   justify-content: center;
7   align-items: center;
8 }
9
10 .line-scale-party > div {
11   background-color: @baseColor;
12 }
```

图 6.2.9 Loading 组件实现部分代码

（10）Share 组件的实现

Share 实现了当用户点击分享按钮时，提供可分享的平台按钮，供用户分享。

```
class Share extends React.Component<IShareProps, IShareState> {
  render() {
    return (
      <Motion style={{x: spring(this.props.showShare ? 0 : 100)}}>
        {{{x}} => {
          return (
            <div className='share'
              style={{transform: `translateY(${x}px)`}}>
              {SharePlatform.map((item, i) => {
                return (
                  <a key={i} className='item' href={item.link}>
                    <img src={item.icon} alt={item.name} />
                    <span>{item.name}</span>
                  </a>);
                })
              }
            </div>
          );
        }}
      </Motion>
    );
  }
}
```

图 6.2.10 Share 组件实现部分代码

（11）BookContent 组件的实现

BookContent 组件为阅读页面的布局，由 header、footer、content 组成。其中 content 是将文章内容安装当前设备大小进行布局，将页面逼真的显示在用户面前。

```
class BookContent extends React.Component {
  constructor(props: IBookContentProps) {...}
  componentWillReceiveProps(nextProps: IBookContentProps) {...}
  prevPage = () => {...};
  nextPage = () => {...};
  getOffsetContentWidth = (i: number) => {...};
  setOffset = (offset: number) => {...};
  render() {
    const { everyWidth, page, worked,
      setWorked, bookInfo, menuId } = this.props;
    const { currentPage } = this.state;
    const showPage = page === 0 ?
      [1] : Array.apply(null, Array(page + 1));
    const rate = page ? currentPage / page : 0;
    return (
      <FlipAnimation
        setOffset={this.setOffset} nextPage={this.nextPage}
        prevPage={this.prevPage} worked={worked} setWorked={setWorked}>
        <xml/>
        {
          showPage.map((item, i) => {
            if (Math.abs(i - currentPage) <= 1) {
              const offsetContentWidth = this.getOffsetContentWidth(i);
              const offsetWrapWidth = i * (everyWidth - 30);
              return (
                <Motion key={i}
                  style={{ x: spring(offsetContentWidth),
                    y: offsetWrapWidth }}>
                {{ { x, y } } => {...}
                }
                </Motion>
              );
            }
          })
        }
        <div className='page-footer'>
          <xml/>
        </div>
      </FlipAnimation>
    );
  }
}
```

图 6.2.11 BookContent 组件实现部分代码

（12）BookContentHeader 组件的实现

BookContentHeader 组件为当用户点击阅读页面时，缓缓展示给用户的 header 部分。实现了用户退出阅读模式，收藏和分享功能。

```
class BookContentHeader extends React.Component {
  constructor(props: IBookContentHeaderProps) {
    super(props);
    this.state = {
      collect: false,
    };
  }
  collect = () => {...};
  componentDidMount() {...}
  componentWillReceiveProps(nextProps: IBookContentHeaderProps) {...}
  isCollected = (props: IBookContentHeaderProps) => {...};
  render() {
    const isCollected = this.isCollected(this.props) || this.state.collect;
    return (
      <Motion style={{ x: spring(this.props.show ? 0 : -60) }}>
        {({ x }) => {
          return (
            <div className='book-content-header'
              style={{ transform: `translateY(${x}px)` }}>
              <Link to={` /home/book/${this.props.bookId}`}
                className='left'>
                <xml/>
              </Link>
              <div className='right'>
                <div className='icon'
                  onClick={this.collect}>{isCollected ?
                    <xml/>
                  :
                    <xml/>}
                </div>
                <div className='icon'
                  onClick={this.props.setShareShow}>
                  <xml/>
                </div>
              </div>
            </div>
          );
        }}
      </Motion>
    );
  }
}
```

图 6.2.12 BookContentHeader 组件实现部分代码

（13）BookContentFooter 组件的实现

BookContentFooter 组件为用户阅读设置组件，监听拖动屏幕亮度设置的滚动条，实时将对应屏幕亮度的比例传到数据层，并实现对应的改变屏幕亮度的功能；监听点击白天夜间模式按钮，对应的将页面的亮度调至最高和最低；监听设置字体大小的按钮，在当前字体大小的基础上对新的字体大小的设置进行计算，并对应将页面字体大小改变。

```
class BookContentFooter extends React.Component {
  render() {
    const { actions } = this.props;
    return (
      <Motion style={{x: spring(this.props.show ? 0 : 100)}}>
        {{{x}}} => {
          return(
            <div className='book-content-footer'
              style={{transform: `translateY(${x}px)`}}>
              <div>
                <Brightness />
              </div>
              <div className='setFont'>
                <button
                  onClick={actions.setFontSize.bind(null, '+')}>
                  A+
                </button>
                <span>{this.props.fontSize}</span>
                <button
                  onClick={actions.setFontSize.bind(null, '-')}>
                  A-
                </button>
              </div>
            </div>
          );
        }
      </Motion>
    );
  }
}
```

图 6.2.13 BookContentFooter 组件实现部分代码

（14）BookItem 组件的实现

BookItem 组件实现了获取当前设备的尺寸推算当前文章的页数大小等功能，将数据传给需要该数据的组件用户界面的显示。且实现了设置屏幕亮度时对应页面的亮度变化。

```
class BookItem extends React.Component<IBookItemProps, IBookItemState> {
  componentDidUpdate(prevProps: IBookItemProps) {
    if (prevProps.content.size === 0 &&
        this.props.content.size > 0 ||
        prevProps.fontSize !== this.props.fontSize) {
      const allWidth = this.refs.wrap.scrollWidth;
      const everyWidth = this.refs.content.offsetWidth + 20;
      const page = Math.round(allWidth / everyWidth);
      this.props.actions.getPageWidth({everyWidth: everyWidth, page: page});
    }
  }
  render() {
    const { i, x, y, bright, night, fontSize } = this.props;
    /**
     * 最小亮度为30%，最大亮度为100%，left 最小值为6，最大值为206
     * (left - 6) / 206 - 6 === (brightness - 0.3) / 0.7
     * 初始亮度为95，所以计算left为186
     */
    const brightness = (((bright - 6) * 0.7 / 200 + 0.3) * 100).toFixed(0);
    const style = night ?
    {transform: `translateX(${x}px)`, zIndex: i}
    :
    {transform: `translateX(${x}px)`, zIndex: i,
      background: `hsl(0, 0%, ${brightness}%)`};
    return (
      <div className={night?'book-content book-content-night':'book-content'}
        ref='content' style={style}>
        <div className='wrap' ref='wrap'
          style={{transform: `translateX(-${y}px)`,
            fontSize: `${fontSize}px`}}>
          { this.props.content.map((item, i) => {
            return (
              <p key={i}>{item}</p>
            );
          }) }
        </div>
      </div>
    );
  }
}
```

图 6.2.14 BookItem 组件实现部分代码

（15）FlipAnimation 组件的实现

FlipAnimation 组件完成了手势跟随的翻页动画的实现。通过 touchStart, touchMove 和 touchEnd 的方法记录用户手势在屏幕上移动的位置，并根据这些数据计算移动点之间的距离和角，再结合现实读书翻阅时的情况，将图书翻页动画实现的淋漓尽致，给用户以逼真的读书体验。

```
class FlipAnimation extends React.Component<any, any> {
  pointStart= { x: 0, y: 0 };
  pointCurrent = { x: 0, y: 0 };
  pointLast= { x: 0, y: 0 };
  pointEnd = { x: 0, y: 0 };
  pointMiddle = { x: 0, y: 0 };
  currentDir = '';
  changeDir = true;
  // 获取touch的点
  getTouchPos = (e: any): IPoint => {...};
  // 计算两点之间的水平距离
  getDistX = (p1: IPoint, p2: IPoint): number => {...};
  // 计算两点之间的垂直距离
  getDistY = (p1: IPoint, p2: IPoint): number => {...};
  // 计算两点之间的距离
  getDist = (p1: IPoint, p2: IPoint): number => {...};
  // 计算两点之间所成角度
  getAngle = (p1: IPoint, p2: IPoint): number => {...};
  // 获取移动的方向
  getSwipDir = (p1: IPoint, p2: IPoint): string => {...};
  startEvHandle = (e: any) => {...};
  moveEvtHandler = (e: any) => {...};
  endEvtHandler = () => {...};
  reset = () => {...};
  render() {
    return (
      <div className='book-content-in'
        onTouchStart={this.startEvHandle}
        onTouchMove={this.moveEvtHandler}
        onTouchEnd={this.endEvtHandler}>
        {this.props.children}
      </div>
    );
  }
}
```

图 6.2.15 FlipAnimation 组件实现部分代码

(17) Battery 组件的实现

Battery 组件通过读取用户当前设备的电量及是否充电的状态，将电量状态实时的显示在页面，让用户在阅读页面也能对当前设备的电量了如指掌。

```
class Battery extends React.Component<IBatteryProps, IBatteryState> {
  constructor(props: IBatteryProps) {
    super(props);
    this.state = {
      battery: 1,
      charging: false
    };
    this.getBattery = this.getBattery.bind(this);
  }
  componentDidMount() {
    this.getBattery();
  }
  getBattery() {
    (navigator as any).getBattery().then(res => {
      this.setState({ battery: res.level, charging: res.charging });
    });
  }
  render() {
    const level = (this.state.battery * 100).toFixed(0);
    const date = new Date();
    const hours = date.getHours();
    const minutes = date.getMinutes();
    return (
      <div id='battery'>
        { hours } : { minutes }
        {
          this.state.charging ?
            <div className='battery'>
              <i className='iconfont weui-tabbar__icon'>&#xe627;</i>
            </div>
          :
            <div className='battery'>
              <div className='left' style={{flex: level}}></div>
              <div className='right'
                style={{flex: 100 - +level}}>
              </div>
            </div>
        }
      </div>
    );
  }
}
```

图 6.2.16 Battery 组件实现部分代码

（18）Article 组件的实现

Article 组件为图书介绍页详细信息的布局组件。该组件将图书的详细信息逐一详细的显示在页面中，并将菜单按钮做了部分影藏功能，当用户点击更多按钮时，才将全部的菜单信息渲染在页面，供用户查看。

```
renderMenu() {
  const { bookinfo, id } = this.props;
  const menu = bookinfo.get('catalog');
  const number = this.state.showAll ? menu.size : 15;
  return(
    <ul className = 'menu-list'>
      {
        menu ? menu.slice(0, number).map((item) => {
          return <li key={item.id}>
            <Link to={'/home/book/' + id + '/content/' + item.id}>
              { item.chapter }
            </Link>
          </li>;
        }) : null
      }
      <li onClick={this.allcategoryShow}
        style={{display:this.state.showAll ? 'none' : 'block'}}>
        <a className='more'>
          <i className='iconfont weui-tabbar__icon'>&#xe608;</i>
        </a>
      </li>
      <li onClick={this.allcategoryShow}
        style={{display:this.state.showAll ? 'block' : 'none'}}>
        <a className='less'>
          <i className='iconfont weui-tabbar__icon'>&#xe607;</i>
        </a>
      </li>
    </ul>
  );
}
```

图 6.2.17 Article 组件实现部分代码

(19) LoginForm 组件的实现

LoginForm 组件实现了登录的表单。采用 Redux-Form 的方式实现了实时提示用户输入信息。

```
class LoginForm extends React.Component<any, any> {
  constructor(props: any) {
    super(props);
    this.state = {
      success: false
    };
  }
  getErrorMessage = (error: string) =>{...};
  submit = (values: any) => {
    this.props.AuthActions.loginRequest(values.toJS());
  };
  componentWillReceiveProps(nextProps: any) {
    if (nextProps.auth.get('login').get('data') !== null
      && this.props.auth.get('login').get('data') === null) {
      this.setState({ success: true });
      (document.getElementsByClassName('back')[0] as any).click();
    }
  }
  render() {
    const { handleSubmit, submitting, auth } = this.props;
    const error = auth.get('login').get('error') ?
      auth.get('login').get('error').detail : '';
    return (
      <form className='login-form' onSubmit={handleSubmit(this.submit)}>
        <Field name='account' type='text'
          component={renderField} label='用户名' />
        <Field name='password' type='password'
          component={renderField} label='密码' />
        <div className='form-checkbox'>
          <div>
            <Field name='remember' id='remember'
              component='input' type='checkbox' />
            <span>记住我</span>
          </div>
          <Link to='forgetPassword'>忘记密码? </Link>
        </div>
        <button type='submit' disabled={submitting}>登录</button>
        {error && <div className='error'>{this.getErrorMessage(error)}</div>}
        {this.state.success && <div className='success'>登录成功! </div>}
      </form>
    );
  }
}
```

图 6.2.18 LoginForm 组件实现部分代码

（20）RegisterForm 组件的实现

RegisterForm 实现了注册表单，同样使用了 Redux-Form 的方式来提高用户体验。

```
class RegisterForm extends React.Component<any, any> {
  constructor(props: any) {
    super(props);
    this.state = {
      leftTime: 0,
      success: false,
    };
  }
  submit = (values: any) => {
    this.props.AuthActions.registerRequest(values.toJS());
  }
  countDown = () => {...}
  getVcode = () => {...}
  getErrorMessage = (error: string) => {...}

  componentWillReceiveProps(nextProps: any) {...}
  render() {
    const { handleSubmit, submitting, auth } = this.props;
    const error = auth.get('register').get('error') ?
      auth.get('register').get('error').detail : '';
    return (
      <form className='register-form' onSubmit={handleSubmit(this.submit)}>
        <Field name='nickname' type='text'
          component={renderField} label='昵称' />
        <Field name='account' type='text'
          component={renderField} label='邮箱或者手机号' />
        <Field name='vcode' type='text'
          leftTime={this.state.leftTime} getVcode={this.getVcode}
          component={renderVcodeField} label='验证码' />
        <Field name='password' type='password'
          component={renderField} label='密码' />
        <Field name='passwordRepeat' type='password'
          component={renderField} label='重复密码' />
        <div className='form-checkbox'>
          <div></div>
          <Link onClick={this.props.handleClick} to='/login'>已有账号? </Link>
        </div>
        <button type='submit' disabled={submitting}>注册</button>
        {error && <div className='error'>{this.getErrorMessage(error)}</div>}
        {this.state.success && <div className='success'>注册成功! </div>}
      </form>
    );
  }
}
```

图 6.2.19 RegisterForm 组件实现部分代码

（21）Collection 组件的实现

Collection 组件为收藏图书列表和推荐图书列表的布局实现已经用户没有登录状态下的提示信息的显示。

```
const MyCollection = ({ collect, recommend }) => {
  return (
    <div className='collection'>
      <section className='my_collection'>
        <ul>
          {
            collect.get('list').map((item, i) => {
              return (
                <li key={i}>
                  <Link to={` /home/book/${item.id}`}>
                    <img src={item.picture} />
                  </Link>
                </li>
              );
            })
          }
        </ul>
      </section>
      <section className='my_collection'>
        <h3>根据您的兴趣推荐</h3>
        <ul>
          {
            recommend.get('list').map((item, i) => {
              return (
                <li key={i}>
                  <Link to={` /home/book/${item.id}`}>
                    <img src={item.picture} />
                  </Link>
                </li>
              );
            })
          }
        </ul>
      </section>
    </div>
  );
};
```

图 6.2.20 Collection 组件实现部分代码

（22）User 组件的实现

User 组件用户展示用户基本信息，并实时从后端获取用户的阅读历史数据和用户喜欢的阅读类型和作者的分析数据，展示给用户。

```
class User extends React.Component<IUserProps, IUserState> {
  render() {
    const account = this.props.account.get('data');
    return (
      <div className='app'>
        <Header />
        <div className='user'>
          <header>
            <div className='nickName'>
              <div className='name'>
                <i className='iconfont weui-tabbar__icon picture'>&#xe74c;</i>
                {account ?
                  <span>{account.nickname}</span>
                  :
                  <Link to='/login'>去登录</Link>
                }
              </div>
              <i className='iconfont weui-tabbar__icon level'>&#xe62d;</i>
            </div>
            <xml/>
          </header>
          {account &&
            <section>
              <div className='title'>喜欢的阅读类型</div>
              <xml/>
            </section>
          }
          {account &&
            <section>
              <div className='title'>喜欢的作者</div>
              <xml/>
            </section>
          }
        </div>
        <Footer />
      </div>
    );
  }
}
```

图 6.2.21 User 组件实现部分代码

第七章 总结

本项目结合平时学习生活中手机阅读中所需要的一些必要功能，结合市场上阅读器的众多优点，也针对现有软件的一些不足，实现的一个基于 WEB 应用的多平台的手机阅读器。

项目开发过程中，主要使用 Visual Studio Code 作为主要开发工具，采用 Mongodb 这个非关系数据库，使用 Go 语言实现访问数据库和处理后端逻辑，前端采用 React+Redux+TypeScript 开发框架，采用 Webpack+Gulp 作为前端打包工具。采用 react-redux-router 实现了页面跳转无刷新，Immutable 完美实现了不可变数据。本地数据存储采用 localStorage，用户信息存储使用 cookie，降低了服务器的压力。使用 React 封装的 touch 事件和 CSS3 动画完美的实现了手势跟随翻页的复杂动画，并使用第三方登录、分享接口，实现了账号统一，基本社交功能的需求。

在项目的设计开发过程中，我充分的了解了软件设计开发过程中的设计思想和流程，分析了在学习生活中现有手机阅读器的众多优势和不足，并在学生群体中做了充分的调查。该项目的开发不仅实现了所需的必要功能，还优化了很多现有软件的不足。下面我将对我的项目特色做详细的介绍：

（1）精心挑选的颜色搭配及个性化的用户设置。

项目主界面采用浅橙色这种既不失重要提示作用，又给用户以舒服感觉的颜色，让用户在使用的过程中，保持愉悦的心情，迅速找到自己所需的书籍。阅读界面分白天、夜间两种模式，实现了在不同灯光下对眼睛的最大保护作用。

（2）简洁舒服的界面设计。

项目的设计十分简洁，将重要功能，比如搜索、分类、菜单放置在页面最显眼的位置。阅读界面尽可能的实现仅包含阅读功能，这样能更好的让读者专心阅读，身临其境。

（3）流畅的翻书动画。

项目尽可能模拟现实物理现象实现动画，并针对手势做了多方面的细节处理，给用户以真实的阅读体验。

（4）第三方登录、分享的实现。

项目除了自身的登录功能之外，实现了第三方登录接口，用户可以使用自己常用的社交账号进行登录，减少登录注册的麻烦，方便用户更快、更好的使用。

（5） 实时自动记录用户阅读历史。

用户在找书，阅读的过程中，应用会自动记录用户的查找、阅读历史，并在用户再次进入应用时智能提醒用户是否回到上次阅读的状态。

（6） 按需加载，节省流量。

流量的使用情况关系到用户是否放心使用该应用，所以这方面的优化也是必不可少的。在用户使用应用时，会获取用户当前的网络连接方式，如果是在数据流量的情况下，尽可能的减少图片加载，减少无关信息的加载等，在 Wifi 情况下，对用户最新访问的图书进行附近章节的缓存，以保证用户在无网络情况下仍可正常使用。

（7） 智能推荐，为用户推荐可能想看的书籍。

项目会根据用户的阅读历史，阅读行为进行智能分析，并在数据库中匹配相应的书籍推荐给用户。

由于开发时间短，项目在细节处理方面还有很多不足，需要在接下来的学习生活中继续实践和完善。

参考文献

- [1] Lea, Verou. CSS Secrets: Better Solutions to Everyday Web Design Problems[M]. O'Reilly Media:Lea Verou, 2015.
- [2] Biography. Learning React Native: Building Native Mobile Apps with JavaScript[M]. O'Reilly Media:Biography, 2016.
- [3] Juho, Vepsäläinen. SurviveJS - Webpack and React: From apprentice to master[M]. CreateSpace Independent Publishing Platform:Juho Vepsäläinen, 2016.
- [4] Remo, H, Jansen. TypeScript 2.0 Cookbook[M]. UK:Packt Publishing, 2017.
- [5] Cassio, de, Sousa, Antonio. Pro React[M]. Apress; 1st ed. 2015 edition:Cassio de Sousa Antonio, 2015.
- [6] David, Flanaga. JavaScript: The Definitive Guide: Activate Your Web Pages [M]. O'Reilly Media:David Flanaga, 2015.
- [7] Douglas, Crockford. JavaScript: The Good Parts[M]. O'Reilly Media:Douglas Crockford, 2008.
- [8] Stoyan, Stefanov. JavaScript Patterns[M]. O'Reilly Media:Stoyan Stefanov, 2010.
- [9] Ethan, Brown. Learning JavaScript: JavaScript Essentials for Modern Application Development[M]. O'Reilly Media:Ethan Brown, 2016.
- [10] Eric, T, Freeman. Head First JavaScript Programming[M]. O'Reilly Media:Eric T. Freeman, 2014.
- [11] Ravi, Kumar, Gupta. Test-Driven JavaScript Development[M]. Packt Publishing:Ravi Kumar Gupta, 2015.
- [12] Clement, Nedelcu. Nginx HTTP Server - Third Edition[M]. Packt Publishing:Clement Nedelcu, 2015.
- [13] Steve, Corona. nginx: A Practical Guide to High Performance[M]. O'Reilly Media:Steve Corona, 2016.
- [14] 谢孟军. Go Web 编程[M]. 北京:电子工业出版社, 2013.
- [15] 刘一奇. React 与 Redux 开发实例精解[R]. 北京:电子工业出版社, 2016.

致谢

经过三个多月的不懈努力，毕业设计即将告一段落了。总体来说，这次毕业设计还算是比较成功的。毕业设计的选题源自我的学习生活，初步设计的功能也都基本实现，包括细节和用户体验也做了多方面的处理。当然毕竟时间有限，加上自己的知识面也没有达到一定的宽度，项目的很多地方还是可以完善的，在今后的学习生活中，我会将该项目作为日常应用使用并不断的优化，争取做到可以上线的版本。

在这次毕业设计的过程中，我对大学四年所学的知识有了一个系统的认识，学习到了一个简单项目完成的不易。从产品设计到需求整理，再到真正的前后端分离的实现，还有一系列细节问题的处理，让我深感自己知识的欠缺，也激起了我继续学习的热情。

在这次毕业设计的过程中，我要特别感谢我的指导老师符琦老师，符琦老师是一位对学生，对工作认真负责的好老师。从大一开始，他就教给我们很多学习计算机的必备技能，从他的学习工作经历出发，给我们提供了更大的平台去学习技术。符琦老师对于新时代的新技术也一直充满激情，也一直鼓励我们除了书本知识，要去培养自己对于新技术的学习能力。在完成本次毕业设计的过程中，不管在选题方面，还是细节方面，都给了我很多建设性的建议，使我顺利完成了此次毕业设计。

我还要感谢李志刚老师，在我刚刚进入大学还很迷茫的时候，给了我很多方向上的建议，让我更加坚定的走技术这条路。

同时我还要感谢从大一一开始就带我打比赛，一起讨论技术问题的王同学，在他的帮助下，我少走了很多技术弯路，让我能够在很短的时间内，学到更多的知识和技能。