

**毕业设计（论文）题目** 私人定制化电脑组装平台

**学 院** 软件学院

**专 业** 软件工程

**学 号** 201320180715

**学 生 姓 名**  陆学平

**指 导 教 师** 王强 **职称**副教授

**自** **月** **日至** **月** **日共** **周**

**信息工程学院 软件学院**

**二○ 年 月**

# 摘 要

本文研究了私人定制化电脑组装平台系统出现的背景、研究现状、发展趋势、开发私人定制化电脑组装平台系统的意义以及研究过程中所使用到的相关技术。

组装电脑又叫DIY电脑，是指自己根据自己的需求购买相关的硬件进行搭配组合安装。它的优势是可以根据自己的需求对相关硬件进行搭配，使其成为最实惠，最适合自己的电脑。随着电子商务的迅速发展，虽然各大商城网站很快的融入了人们的日常生活，但电脑的销售模式还比较单一，都是直接在商城售卖整机或者是零件。然而，整机的价格昂贵，性能虽稳定但却不够强，配置还不是很贴合自己需求，这无疑阻碍了电脑销售行业的发展。在这种形式下，私人定制化电脑组装平台系统的出现，为电脑销售行业带来了机遇和革新。

本系统实现了用户注册登录、一键组装电脑、自定义组装电脑、商品概览、参考组装电脑搭配、晒单以及修改方案等功能。本系统网站开发平台采用了目前流行的express框架搭建后台，使用mongodb作为数据库，同时使用react构建前端页面。

在当下越来越多的人享受DIY的过程的同时，私人定制化电脑组装平台系统的设计与实现，无疑证实了该平台系统给电脑销售行业带来的新的商业契机。

关键词：私人定制化电脑组装平台系统，电脑销售行业，组装电脑

**ABSTRACT**

This paper studies the background, research status, development trend, the significance of developing personal customization computer assembly platform system and the related technology used in the research process.

Assembly computer, also known as DIY computer, is their own needs to buy the relevant hardware with a combination of installation. Its advantage is that according to their own needs to match the relevant hardware, making it the most affordable, the most suitable for their own computer. With the rapid development of e-commerce, although the major shopping malls website quickly into people's daily life, but the computer's sales model is still relatively simple, are sold directly in the mall or parts. However, the machine is expensive, although the performance is stable but not strong enough, the configuration is not very fit their own needs, which undoubtedly hindered the development of computer sales industry. In this form, the emergence of private custom computer assembly platform system for the computer sales industry has brought opportunities and innovation.

The system to achieve the user registration login, a key assembly computer, custom assembly computer, product overview, reference assembly computer with, drying a single and modify the program and other functions. The system website development platform uses the current express framework to build the background, the use of mongodb as a database, while the use of react to build front-end pages.

In the moment more and more people enjoy the DIY process at the same time, private custom computer assembly platform system design and implementation, no doubt confirmed the platform system to the computer sales industry to bring new business opportunities.

**Key words：**Customized computerized assembly platform system， Computer sales industry， assembling computer

目 录

摘要...............................................................................................................................................Ⅰ

ABSTRACT......................................................................................................................................Ⅱ

[绪 论 1](#_Toc14274)

[第1章 系统相关技术理论基础 3](#_Toc210)

[1.1 Express框架 3](#_Toc2330)

[1.2 MongoDB数据库 3](#_Toc23750)

[1.3 React框架 4](#_Toc5988)

[1.4 Javascript语言 5](#_Toc5401)

[1.5 CSS语言 6](#_Toc26624)

[1.6 Webpack 6](#_Toc19748)

[第2章 系统需求分析 8](#_Toc32351)

[2.1 系统需求分析 8](#_Toc6700)

[2.2 业务需求及处理流程 9](#_Toc1017)

[2.2.1 登录流程 9](#_Toc27625)

[2.2.2 组装模块设计 9](#_Toc27967)

[2.3 功能需求分析 10](#_Toc569)

[2.3.1 组装逻辑设计 11](#_Toc27397)

[第3章 数据库设计与实现 1](#_Toc8550)2

[3.1 数据库设计 12](#_Toc13351)

[3.2 数据库代码实现 13](#_Toc425)

[3.2.1定义user的结构与类型 13](#_Toc20863)

[3.2.2 定义配置单结构与类型 14](#_Toc7130)

[第4章 系统详细设计与实现 1](#_Toc9830)6

[4.1登录代码设计 16](#_Toc17502)

[4.2 注册代码设计： 19](#_Toc592)

[4.3首页： 2](#_Toc20187)3

[4.4 项目架构设计 25](#_Toc11338)

[结 论 3](#_Toc2105)0

[致 谢 3](#_Toc8034)1

[参考文献 3](#_Toc503)2

# 绪 论

1. 本系统的研究背景和开发现状：

目前，人类社会已经迈入了网络时代，计算机和互联网已经与人们的日常生活、工作、学习息息相关。在科技发达的现代社会，电脑的用途广泛，方便快捷，深受欢迎。电脑作为很多人日常生活中不可或缺的一部分，不考虑便携性，台式机就是首选。但是很多人选购台式机都是直接买已经组装好的整机。整机的价格昂贵，性能虽稳定但却不够强，配置还不是很贴合自己需求。想要挑选一个符合自己审美以及实用的电脑更是困难，所以自己组装电脑无疑成为一个绝佳选择。

随着互联网电子商务的迅速发展，各大商城网站很快的融入了人们的日常生活。但是，在层出不穷、林林总总的商城网站中，大部分都是以衣物、化妆品、食品、家电等为主体。相比而言，电脑的销售模式还比较单一，都是直接在商城售卖整机或者是零件。然而，整机的价格昂贵，性能虽稳定但却不够强，配置还不是很贴合自己需求，这无疑阻碍了电脑销售行业的发展。而私人定制化电脑组装平台类的系统网站很是稀缺，开发现状令人堪忧。

1. 本系统的研究目的和意义

由于私人定制化电脑组装平台类的系统网站很是稀缺，本系统设计开发的目的就是促进电脑销售行业的发展，为人们的工作和生活提供更加便捷和优质的服务。在本系统上，使用者可以根据视频之类的操作步骤很简单的配置出一台完全符合个人需求的电脑。采用傻瓜式的操作，用户只需要根据自己的预算选择适合自己的配件。游戏党可以将预算更多的分配在显卡，内存上面，设计党可以将预算更多的分配在处理器和硬盘上。在这里，你还可以看别人高性价比的组装方案。同时还有丰富的视频及文字教程，让电脑小白也可以自己动手组装一台属于自己定制的电脑。

私人定制化电脑组装平台系统在适应了当下越来越多的人追逐DIY，享受DIY过程的趋势的同时还促进了电脑销售行业的发展，对电脑的售卖方式进行了一定程度的创新。

3.本文工作概述

私人定制化电脑组装平台系统网站是在对目前互联网前端流行的技术进行了深入的研究和学习的基础上，使用目前流行的express框架搭建后台，使用

mongodb作为数据库，同时使用react构建前端页面。致力于搭建一个完整的功能强大的人定制化电脑组装平台系统网站。

在开发本网站的过程中，首先要搭建相关开发环境，同时将项目利用Git管理起来；其次，需要进行需求分析，将所有需求整合考虑，确定开发步骤；然后，进行数据库分析、模块划分等；最主要的为代码的编写，此过程中要注意的是代码的美观和质量；最后，对系统进行详细的测试。致力于搭建一个完整的功能强大的人定制化电脑组装平台系统网站。

# 第1章 系统相关技术理论基础

## 1.1 Express框架

Express 是一个基于 Node.js 平台的极简、灵活的 web 应用开发框架，它提供一系列强大的特性，可以用来帮助你创建各种 Web 和移动设备应用。而且Express提供各种模块，可以快速地搭建一个具有完整功能的网站。

Express有丰富的 HTTP 快捷方法和任意排列组合的 Connect 中间件，让你创建健壮、友好的 API 变得既快速又简单。但是Express 不对 Node.js 已有的特性进行二次抽象，我们只是在它之上扩展了 Web 应用所需的基本功能。

Express的主要特点：

（1）可以设置中间件来响应http请求；

（2）定义了路由用于执行不同的http请求动作；

（3）可以通过模板传递参数来动态渲染html页面；

## 1.2 MongoDB数据库

MongoDB 是一个基于分布式文件存储的数据库。由C++语言编写，旨在为WEB应用提供可扩展的高性能数据存储解决方案。

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他

支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实

现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。

Mongo主要解决的是海量数据的访问效率问题。因为Mongo主要是支持海量数据存储的，所以Mongo还自带了一个出色的分布式文件系统GridFS，可以支持海量的数据存储。由于Mongo可以支持复杂的数据结构，而且带有强大的数据查询功能，因此非常受到欢迎。

适用场景：

（1）网站数据：适合实时的插入，更新与查询，并具备网站实时数据存储所需的复制及高度伸缩性。

（2）缓存：由于性能很高，也适合作为信息基础设施的缓存层。在系统重启之后，搭建的持久化缓存可以避免下层的数据源过载

（3）大尺寸、低价值的数据：使用传统的关系数据库存储一些数据时可能会比较贵，在此之前，很多程序员往往会选择传统的文件进行存储。

（4）高伸缩性的场景：非常适合由数十或者数百台服务器组成的数据库。

（5）用于对象及JSON数据的存储：MongoDB的BSON数据格式非常适合文档格式化的存储及查询。

应用案例：

（1）京东,中国著名电商,使用MongoDB存储商品信息,支持比价和关注功能。

（2）赶集网,中国著名分类信息网站,使用MongoDB记录pv浏览计数。

（3）奇虎360,著名病毒软件防护和移动应用平台,使用MongoBD支撑的HULK平台每天接受200亿次的查询。

（4）百度云,使用MongoDB管理百度云盘中500亿条关于文件源信息的记录。

（5）CERN，著名的粒子物理研究所，欧洲核子研究中心大型强子对撞机的数据使用MongoDB。

（6）纽约时报，领先的在线新闻门户网站之一，使用MongoDB。

（7）sourceforge.net，资源网站查找，创建和发布开源软件免费，使用MongoDB的后端存储。

## 1.3 React框架

React是一个JavaScript框架，用于构建“可预期的”和“声明式的”Web用户界面，它已经使Facebook更快地开发Web应用。

React有两个主要的特点：

（1）简单

简单的表述任意时间点你的应用应该是什么样子的，React将会自动的管理UI界面更新当数据发生变化的时候。

（2）声明式

在数据发生变化的时候，React从概念上讲与点击了F5一样，实际上它仅仅是更新了变化的一部分而已。React是关于构造可重用组件的，实际上，使用React你做的仅仅是构建组建。通过封装，使得组件代码复用、测试以及关注点分离更加容易。

使用React应注意的是：

（1）React不是一个完整的MVC框架，最多可以认为是MVC中的V（View），甚至React并不非常认可MVC开发模式；

（2）React的服务器端Render能力只能算是一个锦上添花的功能，并不是其核心出发点，事实上React官方站点几乎没有提及其在服务器端的应用；

（3）有人拿React和Web Component相提并论，但两者并不是完全的竞争关系，你完全可以用React去开发一个真正的Web Component；

（4）React不是一个新的模板语言，JSX只是一个表象，没有JSX的React也能工作。

## 1.4 Javascript语言

JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。

Javascript是由 Netscape公司开发的一种脚本语言（scripting language），或者称为描述语言。在HTML基础上，使用Javascript可以开发交互式Web网页。Javascript的出现使得网页和用户之间实现了一种实时性的、动态的、交互性的关系，使网页包含更多活跃的元素和更加精彩的内容。运行用Javascript编写的程序需要能支持Javascript语言的浏览器。

JavaScript 使网页增加互动性。JavaScript 使有规律地重复的HTML文段简化，减少下载时间。JavaScript 能及时响应用户的操作，对提交表单做即时的检查，无需浪费时间交由 CGI 验证。

Javascript脚本语言同其他语言一样，有它自身的基本数据类型，表达式和算术运算符及程序的基本程序框架。Javascript提供了四种基本的数据类型和两种特殊数据类型用来处理数据和文字。而变量提供存放信息的地方，表达式则可以完成较复杂的信息处理。Javascript可以直接嵌入HTML页面，但写成单独的js文件有利于结构和行为的分离。Javascript的跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如Windows、Linux、Mac、Android、iOS等）。

## 1.5 CSS语言

CSS层叠样式表(英文全称：Cascading Style Sheets)是一种用来表现HTML（标准通用标记语言的一个应用）或XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。

CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。CSS为HTML标记语言提供了一种样式描述，定义了其中元素的显示方式。CSS在Web设计领域是一个突破，利用它可以实现修改一个小的样式更新与之相关的所有页面元素。

CSS提供了丰富的文档样式外观，以及设置文本和背景属性的能力；允许为任何元素创建边框，以及元素边框与其他元素间的距离，以及元素边框与元素内容间的距离；允许改变文本的大小写方式、修饰方式以及其他页面效果。

CSS可以将样式定义在HTML元素的style属性中，也可以将其定义在HTML文档的header部分，也可以将样式声明在一个专门的CSS文件中，以供HTML页面引用。总之，CSS样式表可以将所有的样式声明统一存放，进行统一管理。

另外，可以将相同样式的元素进行归类，使用同一个样式进行定义，也可以将某个样式应用到所有同名的HTML标签中，也可以将一个CSS样式指定到某个页面元素中。如果要修改样式，我们只需要在样式列表中找到相应的样式声明进行修改。

## 1.6 Webpack

什么是Webpack，为什么要使用它？

现今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法。比如模块化，让我们可以把复杂的程序细化为小的文件;

类似于TypeScript这种在JavaScript基础上拓展的开发语言。使我们能够实现目前版本的JavaScript不能直接使用的特性，并且之后还能转换为JavaScript文件使浏览器可以识别，以及Scss，less等CSS预处理器。当然，这些改进确

实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常繁琐的，这就为WebPack类的工具的出现提供了需求。

Webpack是一个模块打包器。webpack把模块(s)连同它的依赖一起打包生成包含这些模块的静态资源。它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个浏览器可识别的JavaScript文件。 Webpack的特点：

1. webpack依赖树中有两个依赖类型：同步和异步。异步模块切分成一个新的的块。在块树（chunk tree）优化之后，文件会为每个chunk发文件。
2. webpack有一个智能解析器，它能处理几乎所有的第三方库。它甚至允许你在依赖中你像这样加表达式 require("./templates/" + name + ".jade") 。它可以处理最常见的模块化标准风格：CommonJS和AMD。
3. webpack可以处理javascript本身，但loader用来将其它资源转换为javascript。这样以来，所有资源都被格式化成模块了。

# 第2章 系统需求分析

## 2.1 系统需求分析

笔记本销售网站利用Internet 的优势，使用户足不出户就可以买到自己需要的笔记本电脑，从而打破了传统的电脑城经营模式。笔记本销售网站的宗旨在于方便用户购买自己喜爱的电脑，因此笔记本销售网站应该实现所有方便用户购买电脑所必需的功能和服务。同时，任何一个销售网站都需要进行管理，笔记本销售网站也不例外。 现在，网络购物系统已经比较常见，他的一些功能是非常基础的,为了更有特色一点,也为了更注重细节,所以我在这里的一些基本功能上并没有下太多的工夫,而把另外一部分精力放在了商品的分类管理和客户方面的需求。

## 

## 2.2 业务需求及处理流程

### 2.2.1 登录流程

**![IERZ8K7$TV(L](E]10VWFWO](data:image/png;base64,)**

图2-1 登录流程图

### 2.2.2 组装模块设计

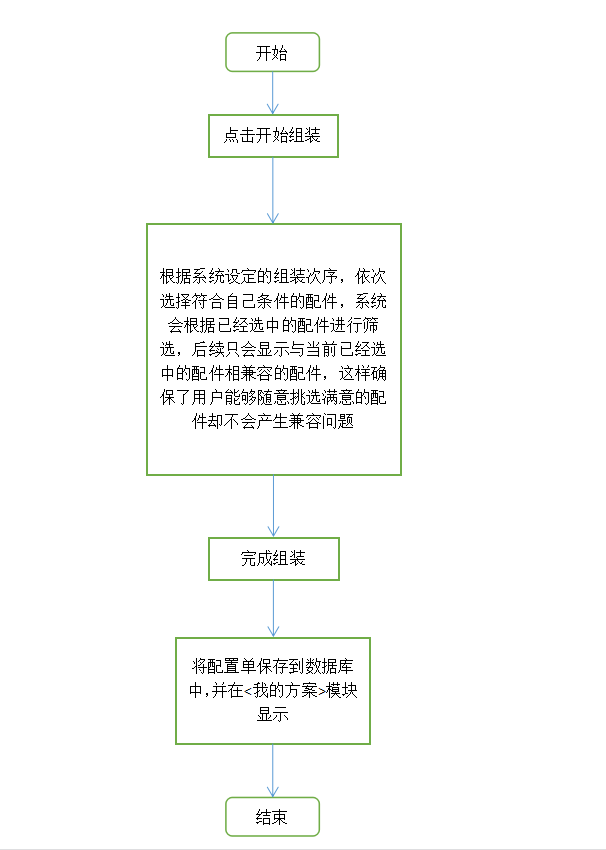


图2-2 组装模块设计

## 2.3 功能需求分析

### 2.3.1 组装逻辑设计

第一步：选择主板，主板主要按尺寸分为ATX主板和MATX主板，不同主板的大小会影响机箱的选择。主板的所有主要接口都是一致的，包括cpu接口，内存条接口，显卡接口以及硬盘等常用接口。

第二步：选择cpu，cpu主要分为N卡和A卡，分别为Nvidia，以及AMD。选择主板后，根据主板的cpu接口，显示符合该接口的所有cpu。

第三步： 选择内存条，内存条的选择主要考虑到主板的频率兼容性以及容量大小，一般支持主板支持双通道的话最好选择两条同等规格的内存条，能够一定程度上提高性能。

第四步： 选择显卡，拥有一个独立显卡能够显著提高机器的性能，显卡的选择主要考虑到主板和cpu的性能，当主板与cpu性能比较低时，不宜选择高性能的显卡。

第五步： 选择硬盘， 硬盘的选择主要考虑到个人需求，能够加上一个固态硬盘是个非常好的选择。一般建议选择固态+机械组合方式。

第六步： 选择机箱， 机箱的选择除去个人对外观的要求，主要考虑是否与主板大小匹配。

第七步： 选择电源， 一个好的电源能够提供充沛的电力，更好的安全性。

第八步： 选择其余配件， 根据个人的需求，可以适当选择一些配件，虽然这不是组装机器必需的，但是拥有这些配件能够给你提供更好的体验。比如：无线网卡，让你不用连接网线也能随心联网；极光风扇，让你的机箱看起来更加炫酷；水冷散热装置，能够提供更好的散热性同时能够减少机器噪音，等等。

# 第3章 数据库设计与实现

## 3.1 数据库设计

该系统中数据库采用mongodb，关于mongodb前面已经有相关介绍了。下面是关于用户信息以及配置方案的数据存储结构：

由于mongodb是非关系型数据库，本身没有太复杂的逻辑结构，这里主要以JSON的格式来存储数据。

用户信息表设计：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 说明 | type | required | ObjectID |
| username | 用户名 | string | true | string |
| password | 密码 | string | true | string |

用户信息：

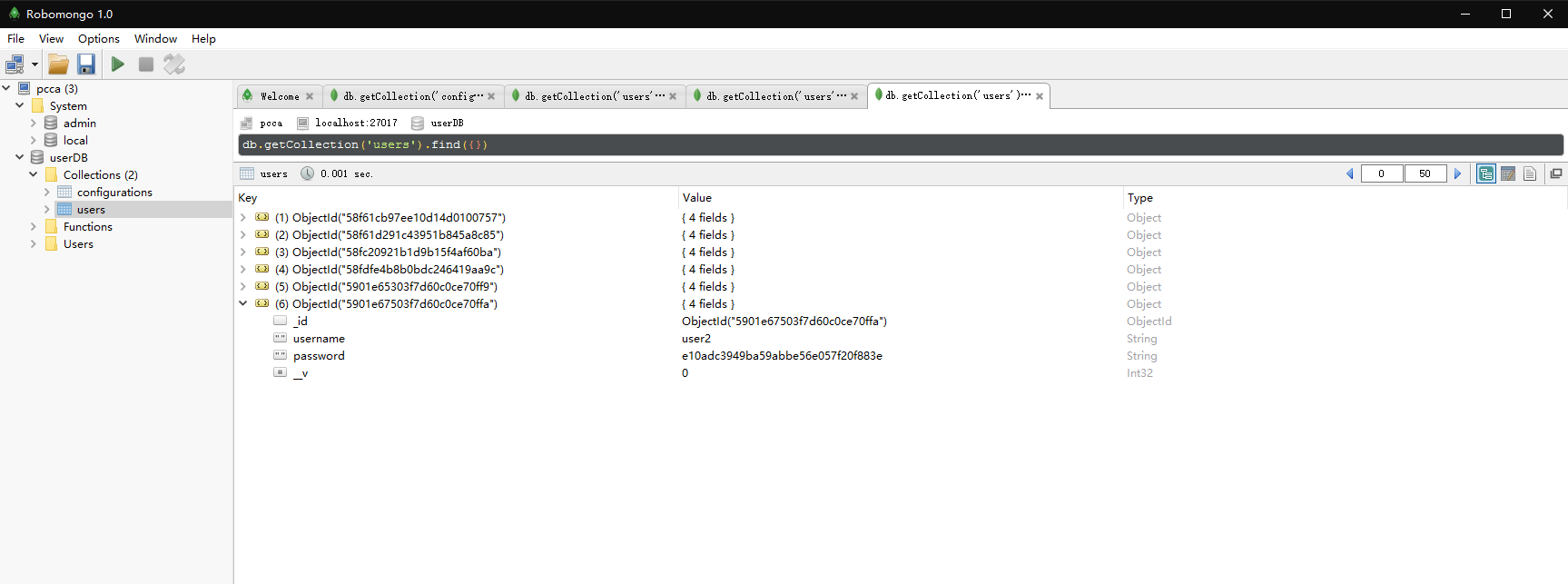


图3-1 用户信息

配置单表设计：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 说明 | type | required | ObjectID |
| id | 唯一标识ID | string | true | string |
| price | 商品价格 | string | true | string |
| name | 商品名称 | string | true | string |
| date | 配置单生成日期 | string | true | string |
| count | 所有商品总价 | string | false | string |

配置单：

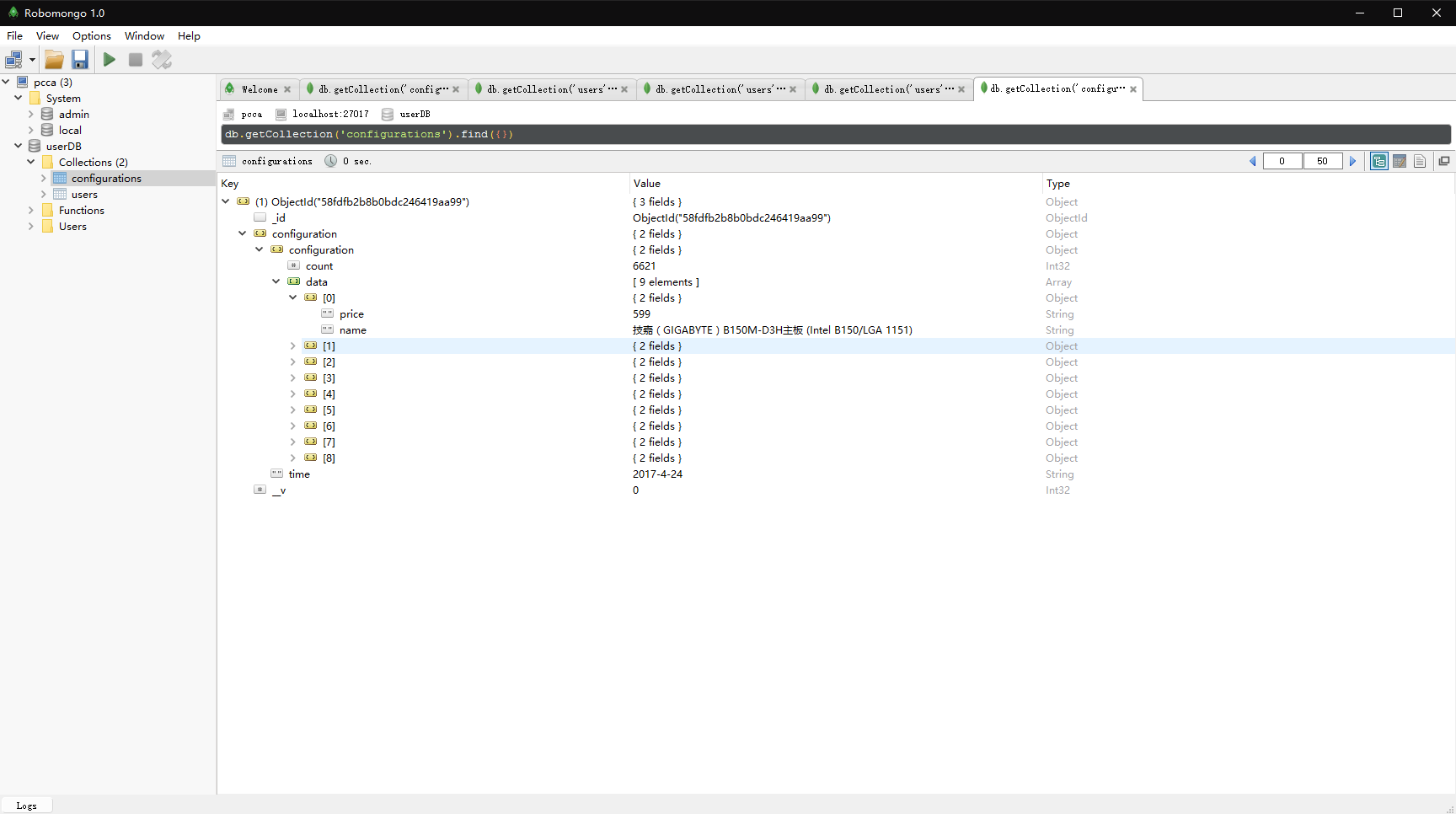


图3-2 配置单

由于原生mongodb的api操作比较繁琐不够便利，这里我使用mongoose来操作数据库，mongoose拥有更为便捷和强大的api，能够很方便的实现增删改查的操作。

## 3.2 数据库代码实现

### 3.2.1定义user的结构与类型

var mongoose = require('mongoose');

// 定义Schema

UserSchema = new mongoose.Schema({

username: {// 真实姓名

type: String,

required: true

},

password: { // 密码

type: String,

required: true

}

});

// 定义Model

var UserModel = mongoose.model('User', UserSchema);

// 暴露接口

module.exports = UserModel;

### 3.2.2 定义配置单结构与类型

var mongoose = require('mongoose');

// 定义Schema

ConfigurationSchema = new mongoose.Schema({

configuration: {

type: {}

}

});

// 定义Model

var ConfigurationModel = mongoose.model('Configuration', ConfigurationSchema);

// 暴露接口

module.exports = ConfigurationModel;

# 第4章 系统详细设计与实现

## 4.1登录代码设计

实现思路：点击登录按钮时，获取用户名输入框、密码框的值，通过ajax的post请求，将数据发送给/login路由，路由接收到该请求后，获取发送过来的数据，对数据进行验证，验证通过则发送一个登录成功的信息给前端，通过页面跳转到首页。

登录页面：

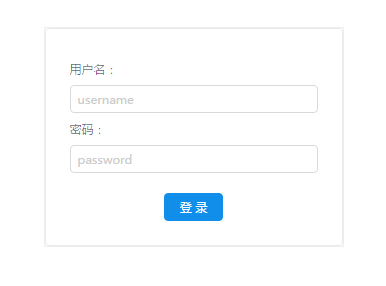


图4-1 登录页面

登录模板设计：

render(){

return (

<div>

<div id='login'>

<Card>

<h3>用户名：</h3>

<Input placeholder='username' ref='username' id='username'></Input>

<h3>密码：</h3>

<Input placeholder='password' type='password' ref='password' id='password'></Input>

<span className='errorMsg'>{this.state.errorMsg}</span>

<Button type='primary' onClick={this.handlerLogin.bind(this)}>登录</Button>

</Card>

</div>

</div>

)

}

登录逻辑设计：

handlerLogin(){

var \_this = this;

var username = document.querySelector('#username').value;

var password = document.querySelector('#password').value;

var data = {

"username": username,

"password": password

};

$.ajax({

url: '/login',

type: 'post',

data: data,

success: function(json){

console.log(json);

if(json.redirect){

\_this.setState({

errorMsg: ''

})

window.dispatchEvent(new Event('pcca-hasLogin'));

window.location = json.redirect;

}else{

\_this.setState({

errorMsg: json

})

}

},

error: function(){

console.log('error');

}

})

}

后端处理逻辑：

//处理登录请求，验证用户名密码

router.post('/login', function(req, res){

var username = req.body.username;

var md5 = crypto.createHash('md5');

var password = md5.update(req.body.password).digest('hex');

req.session.username = username;

req.session.password = password;

//定义用户实体

var user = new User({

username: username,

password: password

});

//在数据库中查询该用户

User.findOne({username: username}, function(err, doc){

if(err){

console.log(err);

console.log('查询数据库时出错啦');

}else if(doc == null){

res.send('查不到该用户!');

console.log('数据库中查无此人');

}else{

if(doc.username == username && doc.password == password){

res.send({redirect: '/#/index'});

}else{

res.send('密码错误');

}

console.log(doc);

}

})

});

## 4.2 注册代码设计：

输入用户和密码后，同登录一样，利用ajax请求，将数据发送给/register路由，路由接收到该请求后，首先验证该用户名是否已经注册，如果该用户名未注册，则将密码进行md5加密后，和用户名一起，存入数据库，否则给前端发送一个信息说明该用户名已经存在。

注册页面：

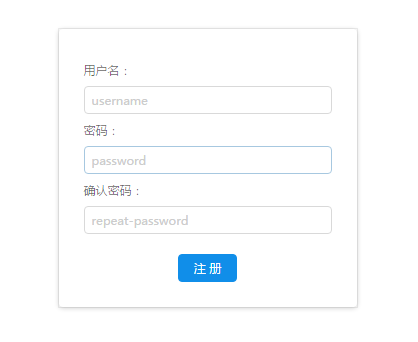


图4-2 注册页面

注册模块设计：

render(){

return (

<div>

<div id='login'>

<Card>

<h3>用户名：</h3>

<Input placeholder='username' id='username'></Input>

<h3>密码：</h3>

<Input placeholder='password' type='password' id='password'></Input>

<h3>确认密码：</h3>

<Input placeholder='repeat-password' type='password' id='repassword'></Input>

<span className='errorMsg'>{this.state.errorMsg}</span>

<Button type='primary' onClick={this.handlerRegister.bind(this)}>注册</Button>

</Card>

</div>

</div>

)

}

注册逻辑设计：

handlerRegister(){

var \_this = this;

var username = document.querySelector('#username').value;

var password = document.querySelector('#password').value;

var repassword = document.querySelector('#repassword').value;

if(password != repassword){

this.setState({

errorMsg: '两次密码不一致'

});

return;

}else{

this.setState({

errorMsg: ''

});

}

var data = {

"username": username,

"password": password

};

$.ajax({

url: '/register',

type: 'post',

data: data,

success: function(json){

console.log(json);

console.log('注册成功');

window.location = json.redirect;

},

error: function(){

console.log('注册失败');

}

})

}

后端逻辑设计：

//处理注册请求，当用户在数据库中未查询到时，将该用户信息存入数据库

router.post('/register', function(req, res){

var username = req.body.username;

var md5 = crypto.createHash('md5');

var password = md5.update(req.body.password).digest('hex');

//定义用户实体

var user = new User({

username: username,

password: password

});

//在数据库中查询该用户

User.findOne({username: username}, function(err, doc){

if(err){

console.log(err);

console.log('查询数据库时出错啦');

}else if(doc == null){

//如果用户名可以使用则将用户信息存入数据库

user.save(function(err, doc){

res.send({redirect: '/#/login'});

});

console.log('用户名可以使用');

}else{

res.send('用户名已经存在');

console.log(doc);

}

})

})

保存组装方案的后端设计：

//存储配置单

router.post('/saveConfiguration', function(req, res){

var username = req.session.username;

var configuration = JSON.parse(req.body.configuration);

//添加时间信息

var date = new Date();

var year = ''+date.getFullYear()+'-';

var month = ''+(date.getMonth()+1)+'-';

var day = ''+date.getDate();

var time = year + month + day;

console.log(time);

var data = {

"username": username,

"time": time,

"configuration": configuration

}

var ConfigurationEntity = new Configuration({

configuration: data

});

ConfigurationEntity.save(function(err,doc){

console.log('数据存储成功');

});

res.send('配置方案保存成功!,你可以在-我的方案-中查看已有的配置方案.');

})

## 4.3首页：

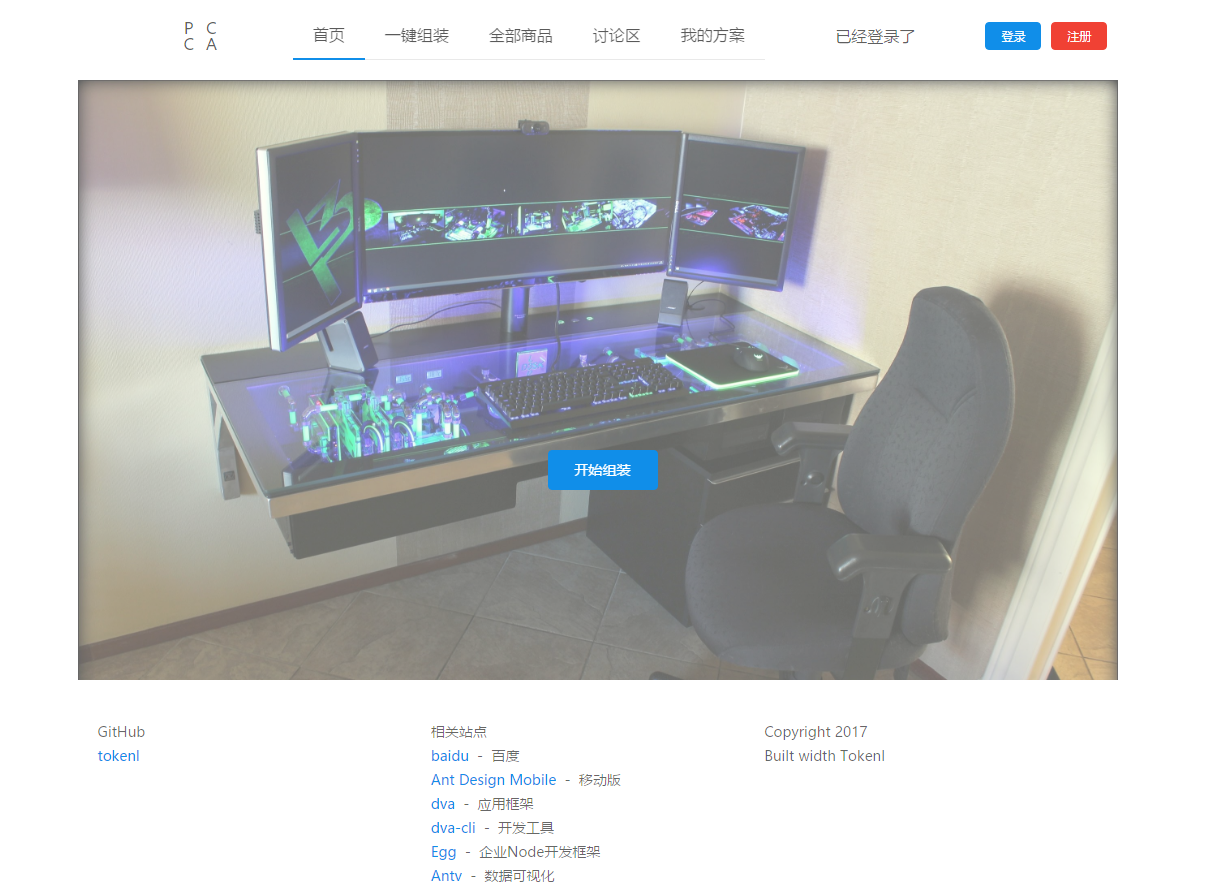


图4-3 首页

配置展示页：

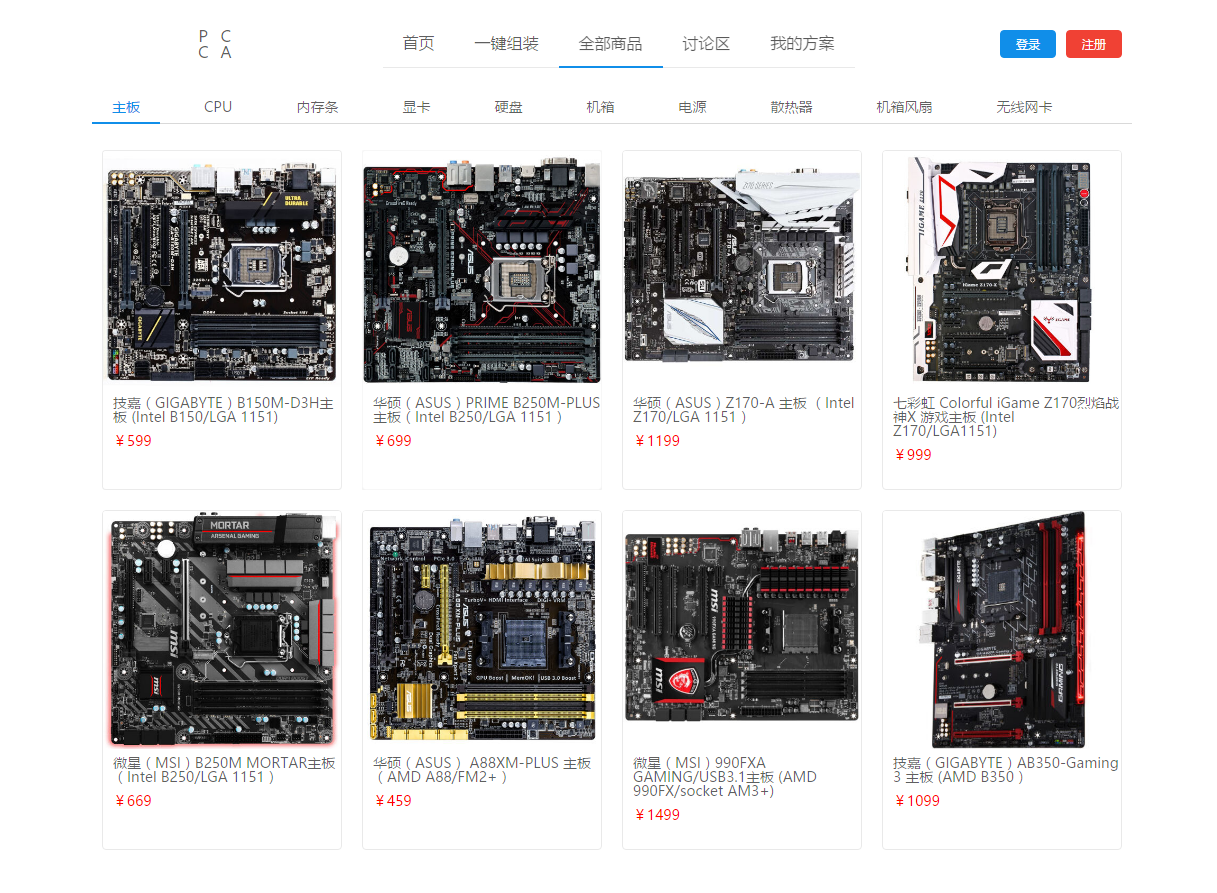


图4-4 配置展示页

组装页面：

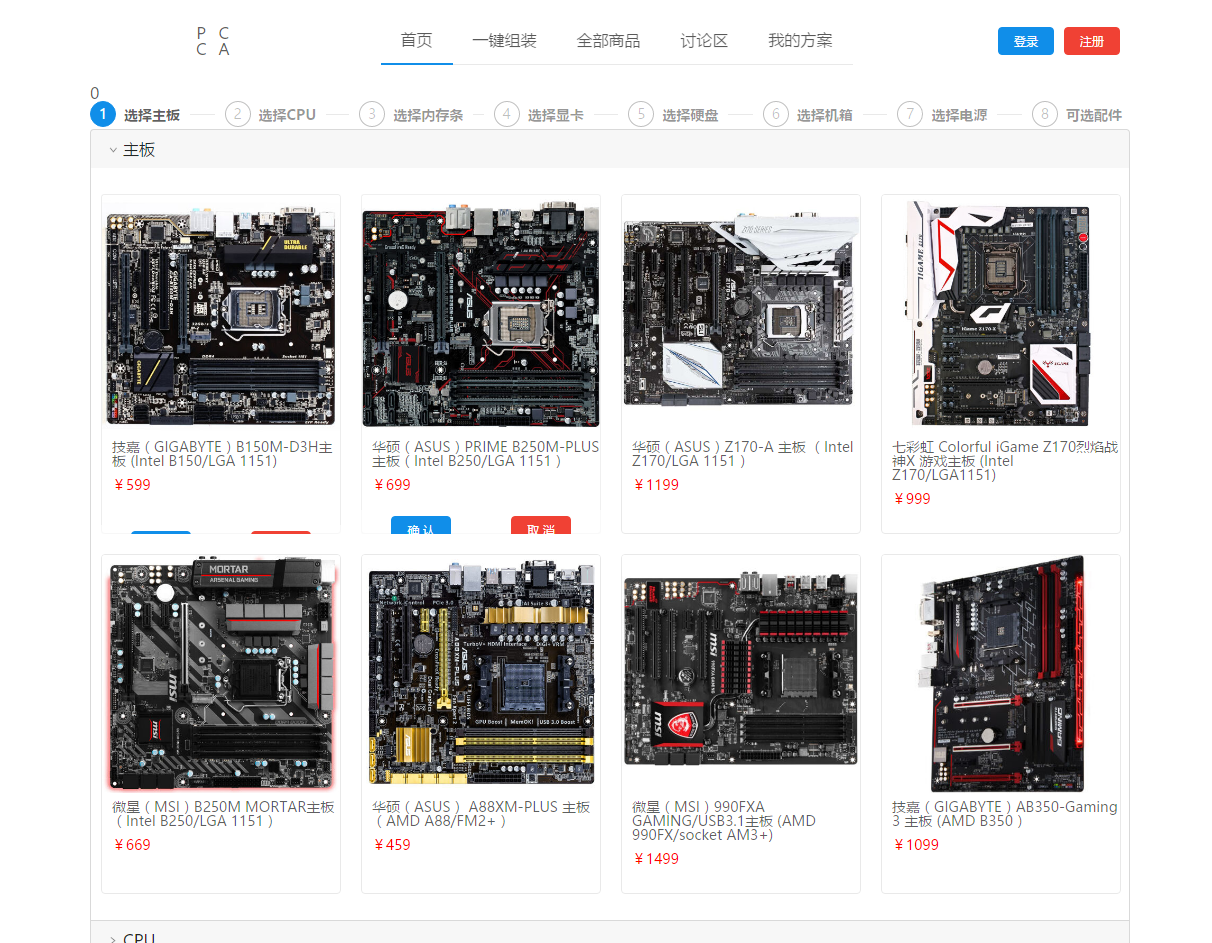


图4-5 组装页面

## 4.4 项目架构设计

首先构建前端页面

以webpack为骨架，React实现前端页面，同时配合ES6以及CSS3，实现了前端的交互页面以及交互逻辑。

webpack是一个流行的模块打包工具，利用它模块化特性，将整个项目的代码以模块的方式进行开发。每一个模块相互独立，通过组合，搭建起了前端的架构。

var path = require('path');

var webpack = require('webpack');

var NoErrorsPlugin = webpack['NoErrorsPlugin'];

var CommonsChunkPlugin = webpack.optimize['CommonsChunkPlugin'];

var ProvidePlugin = webpack['ProvidePlugin'];

var DefinePlugin = webpack['DefinePlugin'];

var dirs = {

src: path.join(\_\_dirname, 'src'),

dist: path.join(\_\_dirname, 'dist'),

content: path.join(\_\_dirname, 'content')

};

var files = {

entry: path.join(dirs.src, 'app.jsx')

};

var plugins = [];

plugins.push(new DefinePlugin({

'env': JSON.stringify('production')

}));

plugins.push(new NoErrorsPlugin());

plugins.push(new CommonsChunkPlugin({

name: 'vendors',

filename: 'vendors.bundle.js'

}));

plugins.push(new webpack.ProvidePlugin({

$: 'jquery',

jQuery: 'jquery',

"window.jQuery": 'jquery'

}));

module.exports = {

entry: {

app: files.entry,

vendors: ['react', 'react-dom', 'react-router', 'react-notify-toast', 'react-tap-event-plugin', 'immutable', 'jquery']

},

output: {

path: dirs.dist,

filename: 'app.bundle.js',

publicPath: 'dist'

},

module: {

loaders: [{

test: /\.(js|jsx)$/,

include: dirs.src,

loader: 'babel'

}, {

test: /\.css$/,

loader: 'style!css?sourceMap'

}, {

test: /\.less$/,

loader: 'style!css?sourceMap!less?sourceMap'

}, {

test: /\.(png|jpg)$/,

loader: 'url?limit=250000'

},{

test: /\.json$/,

loader: 'json'

}],

noParse: []

},

babel: {

plugins: ['transform-runtime', ["antd", { "style": "css" }]]

},

resolve: {

extensions: ['', '.js', '.jsx'],

alias: {}

},

plugins: plugins,

devServer: {

contentBase: dirs.content,

port: 9000

}

};

前端页面构建完成后，利用webpack特有的一个容器：webpack-dev-middleware

webpack-dev-middleware：

它就是一个用来组织包装 webpack 使其可以变成中介软体，或称中间件的容器。回想一下 express 你大概可以明白关于 middleware 的用途，就是在输入到输出的过程中 加工 的一种手段。单纯说 middleware 的话我们可以想成一系列任务, 动作(actions stack)，不只 express 有，在 Ruby 中的 rake 也具备这种机制。

从头说起的话就是 webpack 本身只负责打包编译的功能 bundle, webpack-dev-server 当然就是协助我们开发的伺服器，这个伺服器底层是靠 express 来实作的，接着思考一下我们要如何更新(live reload)呢? 当然是需要取得 webpack 编好的资料啊，于是就需要在从 request 到 response 的过程中透过 express 的 middleware 取得资料，而方法就是透过 webpack-dev-middleware 。

利用webpack-dev-middleware，构建了服务端的机制。搭建了express后端框架，完成了前后端的交互。

var webpackDevMiddleware = require("webpack-dev-middleware");

var express = require('express');

var session = require('express-session');

var fs = require('fs');

var webpack = require("webpack");

var webpackConfig = require("./webpack.config.js");

var app = express();

var compiler = webpack(webpackConfig);

var bodyParser = require('body-parser');

var cookieParser = require('cookie-parser');

var routers = require('./routers');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: false}));

app.use(webpackDevMiddleware(compiler, {

publicPath: '/dist'

}));

app.use(session({

secret: 'recommand 128 bytes random string',

resave: false,

saveUninitialized: true,

cookie: { maxAge: 3 \* 60 \* 60 \* 1000 }

}));

app.use(cookieParser());

app.use(bodyParser.urlencoded({ extended: false }));

app.get('/',function(req,res){

res.end(fs.readFileSync('./content/index.html','utf8'));

});

app.use('/', routers);

app.listen(9000, function () {

console.log('server started on port 9000');

});

最后，引入mongodb数据库，用于存放相关数据，在这里，为了便于操作，使用了mongoose库，mongoose拥有更灵活更强大的api，能够让我们更方便的操作数据库。

# 结 论

随着互联网的的迅速崛起，计算机和互联网已经与人们的日常生活、工作、学习息息相关。而随着互联网电子商务的迅速发展，各大商城网站很快的融入了人们的日常生活。但是，在层出不穷、林林总总的商城网站中，电脑的销售模式还比较单一，都是直接在商城售卖整机或者是零件。然而，整机的价格昂贵，性能虽稳定但却不够强，配置还不是很贴合自己需求，这无疑阻碍了电脑销售行业的发展。而私人定制化电脑组装平台类的系统网站很是稀缺，这就是本系统的一大亮点和创新点。

随着现代科学的发展，人们越来越注重生活的舒适、方便、快捷和参与感。在当下越来越多的人享受DIY的过程的同时，私人定制化电脑组装平台系统的设计与实现，为电脑销售行业带来了商业契机。鉴于对当前电脑销售行业现状的了解和分析，由此产生了我自己的一些想法，并想运用到毕业设计题目当中。在和老师进行一定的沟通，得到老师的肯定后，大胆行动起来。

私人定制化电脑组装平台系统实现了用户注册登录、一键组装电脑、自定义组装电脑、商品概览、参考组装电脑搭配、晒单以及修改方案等功能。本系统网站开发平台采用了目前流行的express框架搭建后台，使用mongodb作为数据库，同时使用react构建前端页面。而且整个网站系统具有界面简洁，反应快速，运行安全可靠的特征。

当然，尽管在系统设计中已经尽力最大努力，可是由于时间、经验等各方面的约束，对于私人定制化电脑组装平台系统还有很多不足的地方。如还没有进行供应商资料查询、管理员权限的选择，库存查询等相关功能的开发，系统还不够完善。但是，在以后的学习和工作中，会继续努力，会努力对系统进行一步步的修改和完善。总之，私人定制化电脑组装平台系统会越来越强大，越来越完善，为电脑销售行业的发展带来不可估量的作用。

# 致 谢

本论文是在导师王强副教授和实习单位的相关同事的悉心指导下完成的。在毕业论文的撰写过程中，导师渊博的专业知识，严谨的治学态度，精益求精的工作作风，诲人不倦的高尚师德，严以律己、宽以待人的崇高风范，对我产生了深远的影响。在毕业论文的撰写过程中，导师和同事的指导不仅使我树立了远大的学术目标、掌握了基本的研究方法，还使我明白了许多待人接物与为人处事的道理，让我受益匪浅。

本论文从选题到完成，每一步都是在导师的指导下完成的，倾注了导师大量的心血。在此，再次导师表示崇高的敬意和衷心的感谢!当然，在这三四年的学习期间，得到过很多同学的帮助和关爱。在此，我要感谢我亲爱的同学们，是他们与我共同学习，交流，与我分享学习的愉悦，分担困惑与烦忧。他们让我感受到了可贵的同窗之情，为我完成学业提供了有力的精神支撑。每当我遭遇困惑，抑郁徘徊之时，他们总能群策群力，提供建议，帮我想办法解决难题，在此表示深深的感谢。

# 参考文献

[1] Nicholas C. Zakas Professional JavaScript for Web Developers Wrox，2012

[2] [Jeremy Keith](https://book.douban.com/search/Jeremy Keith/) / [Jeffrey Sambells](https://book.douban.com/search/Jeffrey Sambells/)  Web Design with JavaScript and the Document Object Model

人民邮电出版社，2011

[3] Kristina Chodorow,Michael Dirolf MongoDB权威指南 人民邮电出版社，2011

[4] BYVoid Node.js开发指南 人民邮电出版社，2012

[5] 朴灵 Node.js深入浅出 人民邮电出版社，2013

[6] Kyle Banker MongoDB in Action Manning，2011

[7] 唐俊开 HTML5移动Web开发指南 电子工业出版社，2012

[8] Jeffrey Sambells,Aaron Gustafson JavaScript DOM高级程序设计 人民邮电出版社，2008

[9] 张轩/杨寒星 React全栈  电子工业出版社，2016

[10] Flanagan JavaScript权威指南 机械工业出版社，2007