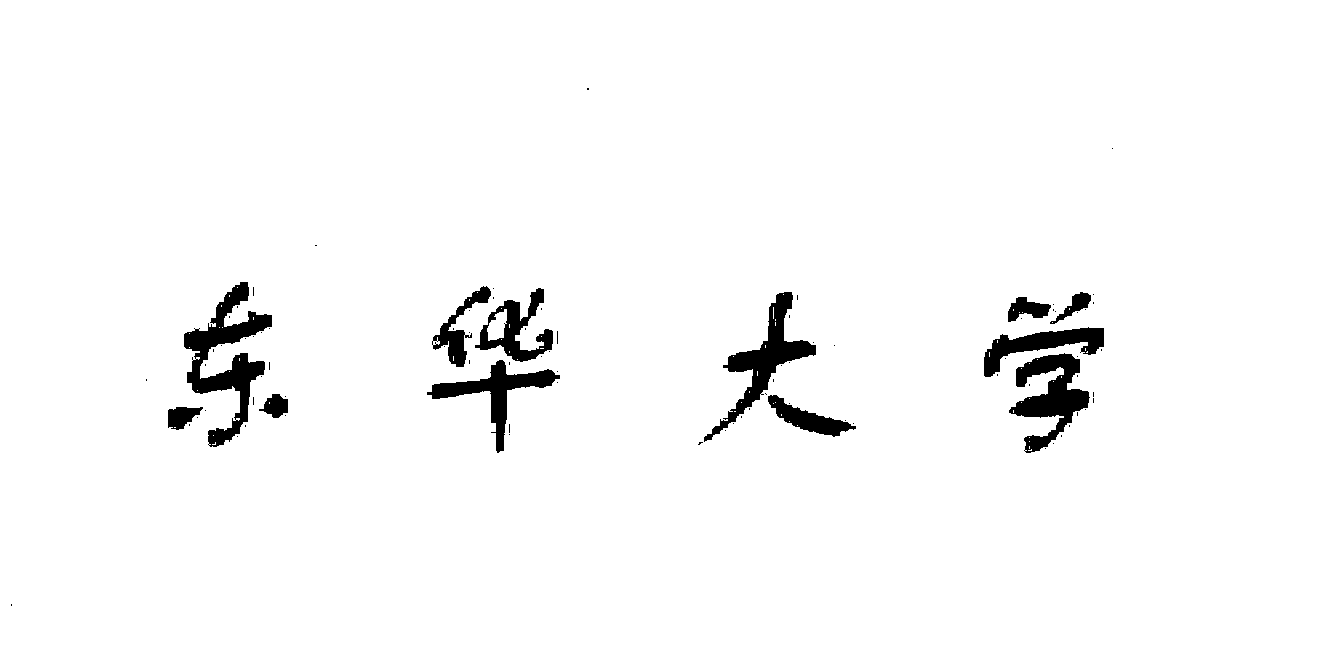
**毕业设计开题报告**



课 题 名 称 ： 基于React技术栈的个人博客系统

　　　 的设计与实现

学 院 ：　　 计算机科学与技术学院

专 业 ：　　 卓越软件1402

姓 名 ： 毛 泉

学 号 ：　　　　　 141320222

指 导 教 师 ：　　　　　　孙 莉

**2018 年 3 月 3 日**

**基于React技术栈的个人博客系统的设计与实现**

**1、背景介绍**

在这个信息爆炸的时代，随着一些社交平台的兴起，大多数人都想要撰写一些内容共享到互联网上，可能是技术的共享，文采的展现，亦或是个人生活轶事等等。但是一些简单的社交平台对于一些较长篇文章的展现和发布有着一些不足。这些不足，也让一部分人开始使用博客，这个诞生于上个世纪的互联网产物，博客平台可以完美支持这些内容，但是对于这个平台都需要一个稳定的、易用的、展现良好的博客系统。

在当前的互联网上有着许多可以搭建博客的网络服务，例如新浪博客，简书等，但是这些平台大多数都起源于上个世纪末本世纪初，有些功能、界面和操作方法可能已经并不适应当前网民的想法和时代潮流，在操作体验上有着或多或少的缺失。但不可否认，这些平台上手有着无技术门槛以及简单便捷的优点，但也有着一些缺点，例如在浏览的时候，会掺入广告，导致浏览者的文章阅读有很多的不愉快和不方便，个人的自定义上也并没有十足的空间进行施展。

所以本次的毕业设计将会实现一个个人的博客系统，通过这个系统平台来实现多用户可以利用的博客发布系统。以此来用自己的想法达成使用方便，去除冗余功能，且无广告能够提供完整交互和完美体验，融合进一些社交平台的优点，以符合当今设计潮流目标。

**2、研究现状**

通过一定的研究，发现在互联网上也有着一些可以提供简单搭建个人网站的服务和程序，例如wordpress或者github的静态网站服务等，这些现有的程序只要在服务器上安装，并稍作配置即可搭建完成一个简单的博客系统。

这一类型的服务主要可以发布文章，并且支持简单的评论功能，但也有着一些不便之处，主要体现在：

1） 使用的技术框架有一定的门槛，在自定义配置的时候需要一定的技术知识。对于一些需要简单操作的人们有着一些不方便的地方，操作并没有达到简洁的标准。

2）这些平台发布文章的时候没有便捷的把本地文章简单上传的方法，只能通过将文章复制黏贴到编辑框中实现，使用并不方便，并且不支持现在主流的markdown语法。

3）这些平台使用的技术因为需要高要求的稳定性，重量级较高，对于一些只需要简单文章发布和浏览等功能的人来说有一定程度大材小用。

**3、毕业论文内容**

本次毕业论文内容主要有着一些几个方面：

1) 研究学习React技术框架以及Node.js的后端构建技术等

这一部分内容由于课堂上并没有过多设计，通过课外资料的学习进行补充自学，基本具备完善和搭建具体应用的能力。

2） 对整个系统的功能模块进行划分，进行任务分析、需求分析和API的约定，对于后续的进展提供一定的指导作用。

3) 确定本博客系统需要完成的基本和进阶功能点

通过与现有程序的对比比较，充分汲取现有系统的优点并且对于自己使用经验，对希望可以优化使用体验的点在本系统中进行修改和完善。

4) 利用学习的技术和确定的功能点具体搭建整个博客系统

利用React技术栈、Node.js与Express后端技术、MongoDB的NoSQL数据库技术，并将前端部分利用Webpack进行工程化的构建，以建设本课题的主体部分。

5) 对搭建起的系统进行测试，获得测试结果

利用一系列的测试用例对已经搭建起的系统进行测试，及时完善系统并且解决在测试中发现的问题

**4、技术路线**

基于React技术栈的个人博客系统主要构建在JavaScript这个语言之上，前端利用React框架及其附属的一些工具,如Redux等，后端使用Node.js来进行实现API访问的处理，从而实现前后端分离，而数据库则是MongoDB。

1) React

在Web技术的发展之后，网页已经不仅仅是展现的工具，更可作为交互的工具。为了满足如此的需求，便有了React等一系列的前端交互框架。

React是一个声明式、组件化的前端框架，其主要负责的是类MVC软件结构中的V部分，也就是UI部分。通过编写拥有各自状态的组件，再由组件构成更加复杂的界面。无需再用模版代码，通过使用JavaScript编写的组件你可以更好地传递数据，将应用状态和DOM拆分开来。更好的完成前后端开发的分离，对个人甚或是企业可以进行行业工作的细分和术业专攻。并且无论现有项目正在使用什么技术栈，你都可以随时引入 React 开发新特性。

React的主要特征就是有状态的组件，React 组件使用一个名为 render() 的方法， 接收数据作为输入，输出页面中对应展示的内容。状态主要体现在了除了使用外部传入的数据以外 (通过 this.props 访问传入数据), 组件还可以拥有其内部的状态数据 (通过 this.state 访问状态数据)。 当组件的状态数据改变时， 组件会调用 render() 方法重新渲染。

React作为一个基础的框架，在其基础上也发展了很多基于其的功能扩展，本课题中也将进行使用。

2） Redux

随着基于React框架的开发日趋复杂，React框架需要管理很多的 state （状态）。 这些 state 可能包括服务器响应、缓存数据、本地生成尚未持久化到服务器的数据，也包括 UI 状态，如激活的路由，被选中的标签，是否显示加载动效或者分页器等等。

管理不断变化的 state 也变得逐渐困难。如果一个 model 的变化会引起另一个 model 变化，那么当 view 变化时，就可能引起对应 model 以及另一个 model 的变化，依次地，可能会引起另一个 view 的变化。直至你搞不清楚到底发生了什么。state 在什么时候，由于什么原因，如何变化已然不受控制。 当系统变得错综复杂的时候，想重现问题或者添加新功能就会变得举步维艰。

为了解决这样的问题，Redux这个React 状态容器就出现了，它提供可预测化的状态管理。

Redux 是一个混合产物。它和一些设计模式及技术相似，但也有不同之处。它利用React提供的Flux思路进行状态管理，并且融合了一些当前比较新潮的函数式编程思想搭建了这个单向数据流管理系统。

3） Node.js与Express

Node.js是JavaScript语言从前端向后端发展的产物。

简单的说 Node.js 就是运行在服务端的 JavaScript。Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。而Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助创建各种 Web 应用，和丰富的 HTTP 工具。使用 Express 可以快速地搭建一个完整功能的网站的后端部分。

通过Node.js和Express可以将前后端均利用JavaScript进行处理，大大减少了不同语言之间进行转换可能导致的错误和不便。

4） MongoDB

MongoDB 是由C++语言编写，是一个基于分布式文件存储的开源NoSQL数据库系统。在高负载的情况下，添加更多的节点，可以保证服务器性能。MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。

正因为类似于JSON对象这个特性，其可以与JavaScript进行简单的交互和获取数据，不用进行多一步的转换，大大减低了过去SQL语言或者SSH框架的复杂性。

5） RESTful架构

RESTful架构，就是目前最流行的一种互联网软件架构。它结构清晰、符合标准、易于理解、扩展方便，所以正得到越来越多网站的采用。REST这个词，是Roy Thomas Fielding在他2000年的博士论文中提出的。

REST，即Representational State Transfer的缩写。主要有如下几个表现特征：每一个URI表示一个资源，客户端和服务端之间传递这种资源，通过HTTP动词对服务器资源进行操作。

利用以上技术，基本可以通过ES6的JavaScript完成整个系统，提供了完善和一致的开发体验。

**5、要解决的技术问题**

1） 前端后端的协调并进，在一个服务器上运行前端和后端两套服务。

2） 决定前后端交互的API，对于RESTful的深度理解。

3） 数据在React框架中的转化和表现。

4） 对于Redux构建系统产生的数据流动的理解。

5） 界面布局的具体风格以及界面框架的选择。

6） 利用Webpack进行前端工程化的实践，前端和后端的热更新简化调试。

7） 浏览数的具体统计以及前端本地化的存储方式、Cookie的合理使用等。

**7、日程安排**

|  |  |  |
| --- | --- | --- |
| **序号** | **各阶段名称** | **起止日期** |
| 1 | **任务熟悉、技术准备** | 2017/11/20–2018/1/18 |
| 2 | **搜集相关资料和参考文献，知识完善** | 2018/1/19-2018/2/28 |
| 3 | **完成开题报告** | 2018/3/1-2018/3/8 |
| 4 | **需求设计，风格设计，模块功能划分** | 2018/3/9-2018/3/25 |
| 5 | **模块代码实现** | 2018/3/26-2018/5/11 |
| 6 | **模块连接，功能调试** | 2018/5/12-2018/5/20 |
| 7 | **论文的撰写和完善** | 2018/5/21-2018/5/29 |

**8、参考文献**

[1] 朴灵 著. 深入浅出Node.js: 人民邮电出版社. 2013年12月

[2] [美] Nicholas C. Zakas著．JavaScript高级程序设计（第3版）：人民邮电出版社．2012年3月

[3] 陈屹著．深入react 技术栈：人民邮电出版社．2016年

[4] David Gourley;Brian Totty;．HTTP权威指南：人民邮电出版社。2012年9月

[5] Kristina Chodorow著 ．MongoDB权威指南（第2版）：人民邮电出版社．2014年1月

[6] [美] Raymond Camden．客户端存储技术：人民邮电出版社．2017年3月

[7] [美] Pete Hunt, Dave Smith, Terry Coatta React: Facebook’s Functional Turn on Writing JavaScript ： http://www.ccs.neu.edu/home/rmacnz/cacm-react.pdf

[8] [日]水野贵明著.Web API的设计与开发:人民邮电出版社.2017年6月

[9] Nicholas C. Zakas著 JavaScript面向对象精要: 人民邮电出版社. 2015年4月 [10] [美] Eric A.Meyer著 CSS权威指南(第三版): 中国电力出版社. 2007年10 月

[11] [美] Jon Loeliger; Matthew McCullough著 Git版本控制管理(第2版):人民邮 电出版社 2015年3月

[12] [德] Matthias Kevin Caspers著 React and Redux: www.uni-oldenburg.de/fileadmin/user\_upload/informatik/ag/svs/download/reader/reader-seminar-ws2016.pdf#page=14