# 摘 要

个人网站的建设既能巩固自己的相关知识，还能在建设网站的过程中学到实战的网络知识体系，这样既积累了知识以及实战经验，对于从事网站设计工作的学生来说是个很好的锻炼机会。

论文叙述了从零搭建个人网站的整个过程，并对在搭建网站中所运用的知识以及工具都有详细的介绍和讲解。比如：用于存储数据的关系型数据库mysql；用于和数据库交互的基于python的后端django框架；用于编写前端显示页面的vue框架；用于代码管理工具以及版本控制系统的git技术；用于编写后端python代码的pycharm ide软件;用于编写前端html,css,javascript的Visual Studio Code编辑软件.

**关键词**：网站搭建， mysql, django, vue, git

# ABSTRACT

The construction of personal website can not only consolidate their own relevant knowledge, but also in the process of building a website to learn the actual network knowledge system, so as to accumulate knowledge and practical experience, for students engaged in website design work is a good opportunity to exercise.

This paper describes the whole process of building a personal website from scratch, and the knowledge and tools used in building a website are introduced and explained in detail. For example: mysql, a relational database for storing data; A python-based back-end django framework for interacting with databases; Vue framework for writing front-end display pages; Git technology for code management tools and version control systems; Pycharm ide for writing back-end python code; Used to write front-end HTML, CSS,javascript Visual Studio Code editing software

**The keywords：**Website construction, mysql, django, vue, git

# 目 录

摘 要 1

ABSTRACT 2

目 录 3

1绪论 5

1.1 数据挖掘代码生成器的背景和意义 5

1.2 自动代码生成技术的发展状况 5

1. 3 数据挖掘代码生成器的研究内容 7

1. 4 系统功能简述 8

2 数据挖掘代码生成系统分析 8

2. 1 整体架构分析 8

2.1.1 系统架构图 8

2.1.2 采用前后端分离 9

2.2 网站需求分析 11

2.2.1 需求分析概述 11

2.2.2 网站系统流程图 12

2.3 本章小结 14

3 数据库分析与设计-MongoDB 15

3.1 后台数据库以及MongoDB概述 15

3.1.1 Mongodb数据库介绍 15

3.1.2 为什么选择MongoDB数据库 15

3.2 数据集合的具体设计 15

3.2.1 数据库设计三范式 16

3.2.2 数据库集合设计展示 16

3.2.3正确设计一张表的结构 20

3.3 本章小结 20

4 开发前的环境准备 20

4.1 前端程序的开发时环境nodejs 21

4.1.1 node.js简介 21

4.1.2 node.js的下载 21

4.2 后端程序运行时环境python 21

4.2.1 python简介 21

4.2.2 Ubuntu下安装python 21

4.3 版本控制工具git 21

4.3.1 版本控制系统的作用 21

4.3.2 选择git作为版本控制工具 22

4.3.3 git的安装及相关命令 22

4.4 本章小结 23

5 后端api的实现Django 23

5.1 Django简介 24

5.1.1 初识Django 24

5.1.2 前后端分离式的Django运行流程 24

5.2 后端程序的全局配置 25

5.3 路由系统及视图函数绑定 26

5.3.1 根级路由和应用子路由 26

5.3.2 view视图函数接收请求和作出响应 27

5.4 和数据库的交互 28

5.4.1 模型（Models）基本概念 28

5.4.2 Django自带的ORM 28

5.4.3 以模型的设计来实现数据表的创建 29

5.4.4 通过ORM来完成数据库CRUD 30

5.5 本章小结 31

6 前端页面实现Vue 31

6.1 页面技术点分析 31

6.1.1 页面开发三要素 31

6.1.2 简化样式的编写bootstrap 32

6.1.3 使用vue来构建用户界面 32

6.1.4 spa技术 33

6.1.5 实现ajax请求-axios 33

6.2 页面的具体实现 34

6.2.1 前端项目目录结构 34

6.2.2 各个页面组件的实现效果 34

6.3 前端页面完成后的测试 39

6.4 本章小结 39

7 网站的上线部署 39

7.1 上线前的准备 39

7.2 上线后的网站流程分析 40

7.3 正式部署 40

7.3.1 前端vue的部署 40

7.3.2 后端部署 41

7.3.3 配置静态文件路径 43

7.4 总结 43

参考文献 45

致 谢 46

# 1绪论

## 1.1 数据挖掘代码生成器的背景和意义

近年来，数据挖掘引起了信息产业界的极大关注，其主要原因是存在大量数据，可以广泛使用，并且迫切需要将这些数据转换成有用的信息和知识。获取的信息和知识可以广泛用于各种应用，包括商务管理、生产控制、市场分析、工程设计和科学探索等。数据挖掘利用了来自如下一些领域的思想：来自统计学的抽样、估计和假设检验；人工智能、模式识别和机器学习的[建模技术](https://baike.baidu.com/item/%E5%BB%BA%E6%A8%A1%E6%8A%80%E6%9C%AF/19041163)和学习理论。数据挖掘也迅速地接纳了来自其他领域的思想，这些领域包括最优化、进化计算、信息论、信号处理、可视化和信息检索。一些其他领域也起到重要的支撑作用。特别地，需要数据库系统提供有效的存储、索引和查询处理支持。源于高性能（并行）计算的技术在处理海量数据集方面常常是重要的。分布式技术也能帮助处理海量数据，并且当数据不能集中到一起处理时更是至关重要。

随着互联网各种的技术的成熟以及应用的落地，数据量的与日俱增，不同行业对数据分析的需求也因此产生了巨大的增长。此外，在目前的各种互联网产品中，后台存储的大量数据拥有着巨大的挖掘价值，随着机器学习技术在各领域的广泛应用，使得对大批量数据的建模分析变得容易。

然而，在企业实际运作中，由于不同数据源产生的数据往往有着不同字段，以及不同的预处理逻辑，使得面对不同的数据集时需要针对特定场景进行代码的重新编写。鉴于数据挖掘流程有着明确的处理流程，高可复用的逻辑，即数据探索、数据清洗、数据建模，模型评估等步骤。本文着眼于将数据挖掘流程模板化，通过用户在界面选择对应方法，个性化定制，生成以用户上传数据集为分析目标的python数据挖掘代码，提高数据分析流程的效率。

## 1.2 自动代码生成技术的发展状况

自动代码生成技术从20世纪末的萌芽阶段发展之今,已经趋于平稳和成熟,“自动代码生成”从1991年开始出现相关研究,2015年达到最热｡如图1所示(统计图中的纵轴代表研究成果,文中的数据统计于百度学术)

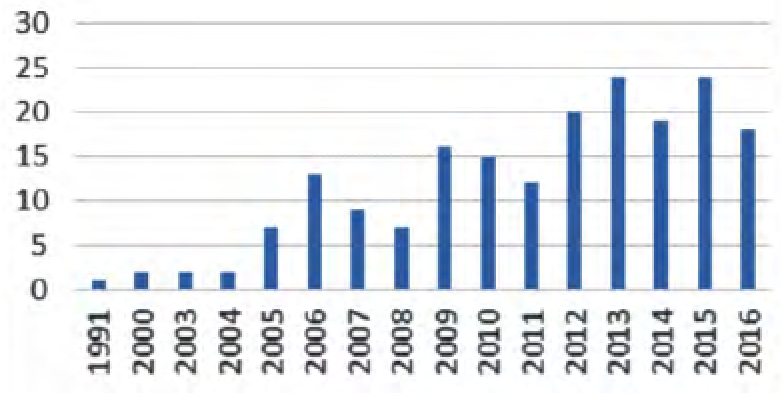


图1自动代码生成技术趋势图

自动代码生成是一个多领域多学科交叉的学科,涉及的面比较广｡随着研究的不断深入,出现了越来越多与“代码生成”相关的研究点,形成了庞大的研究网络,该网络囊括:软件开发､编译器､模型驱动､嵌入式系统､设计模式和编译程序等领域的知识｡国内的软件企业在自动代码生成和模型驱动构架的研究和开发中也有不少成果,比如金蝶软件公司是首家实现支持模型驱动构架软件开发的商业工程工具金蝶EAS4.0,是一个基于模型驱动架构､采用“业务模型驱动的全自动化软件工厂”开发的企业应用平台｡楚凡科技是专业复杂软件及系统开发解决方案的供应商｡为软件开发的整个生命周期提供集成的产品和专业化服务,楚凡科技工具KantStudio集合了UML建模和数据库建模;实现数据库模型和UML模型的双向转换;实现代码和模型之间的迭代转换｡

一些中小企业和开源产品在上自动代码生成方面也有较好表现,开发了小型的代码生成器支持从模型或者模板到不同程序设计语言的转化｡如动软.Net代码生成器是基于关系数据库元数据库的C#自动代码生成器,Codematic生成的代码,基于面向对象思想和三层架构设计｡BBOSS自动代码生成工具是一款为专有框架和平台生成前端､后端代码､WEB服务代码､SQL配置､IOC配置的可视化配置管理工具｡CodeGenerator生成工具是基于Freemarker和Velocity模板生成代码的引擎,该生成器是根据关系实体的元数据来生成源代码｡

国内的众多高校和研究机构也在代码生成､尤其是基于模型驱动骨架的代码生成软件工程也进行了长期的研究,取得了众多的成果｡下面给出国内的在自动代码生成方面研究的典型问题,文献｡在基于模型转换和代码生成､对象关系映射方面的自动代码生成,基于模式的用户界面自动代码生成技术,在逆向工程和程序流程图到自动代码生成提出一种基于图的生成算法,在平台无关模型到平台相关模型,web用户界面建模和自动生成,用户界面代码自动生成等等[1]

## 3 数据挖掘代码生成器的研究内容

本文主要研究通过python实现基于Web端的数据挖掘代码生成器。代码的生成主要采用基于基于模板的代码生成方法。基于模板的代码生成的基本原理是把软件的需求分成两部分,一部分是相对来说固定不变的部分,称为静态部分，即后台系统中的模板文件,另外一部分是根据外部输入变化的部分，即来自前端用户的输入（如选择哪些列做数据预处理，选择什么模型构建数据等等），如图2所示。



图2基于模板的自动代码生成

本系统基于数据挖掘的流程，通过预先设置好数据预处理、数据可视化、数据建模对应的模板文件。经过代码生成引擎调用，结合前端数据的输入，动态生成可运行的python脚本，同时运行脚本生成可视化报告，用于前端展示。

为了能够取得更直观的界面以及更佳的用户体验，本文采用Ｄｊａｎｇｏ＋ｍｏｎｇｏＤＢ＋Ｖｕｅ的架构实现代码生成器的系统功能。

用户通过上传数据集，系统读取数据存入mongodb，前端可展示该用户所有数据集，通过预览按钮解析上传文件并做对应前端展示。数据展示端提供一键生成数据分析报告功能，以交互式网页展示所选数据集每一列的类型、缺失值、异常值、强相关变量，并以高亮的形式展现，用以提示用户，作为数据预处理的参考

## 4 系统功能简述

本系统以数据驱动。通过用户上传数据集到用户数据管理，选择需要分析的数据集，进行数据相关操作。

①、数据预览。数据预览将数据集以表格的形式展示，并支持一键生成数据分析报告。数据分析报告针对数据进行整体的探索性分析，将数据集中存在缺失值、异常值、高相关度的关键信息予以高亮展示。同时针对数据集各列的维度进行详细分析以及分布特征的可视化图表。数据分析报告给出的高亮信息用以为数据清洗提供参考方向

②、数据清洗。用户在已上传数据列表选择需要分析的数据集，自定义选择不同的列，提供不同的数据清洗规则。支持的数据清洗规则主要包括缺失值填充、排序、模糊匹配、归一化等，允许用户自定义调换规则顺序，后台引擎根据用户输入的规则进行校验，并展示清洗后的数据表格。

③、数据建模。用户选择指定数据集，选择数据集中的若干列作为数据特征，选择一列作为目标列。提供分类、回归问题的若干种训练模型以及模型评估方法。参数选择完毕通过弹窗显示生成的代码

# 数据挖掘代码生成系统分析

## 1 整体架构分析

### 系统架构图

本系统基于Web框架运行，使用Python采用前后端分离的方式进行。

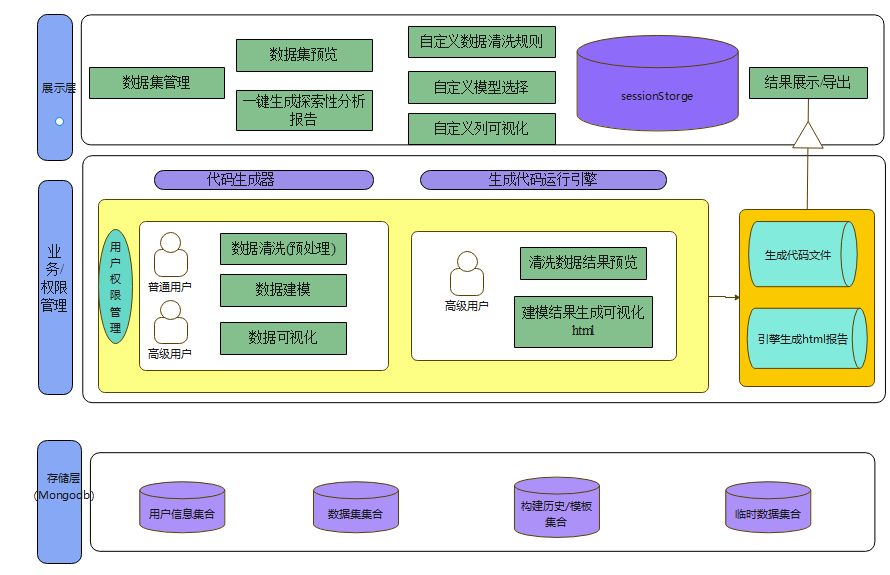


图2-1 数据挖掘代码生成系统架构图

## 2.2 网站需求分析

### 2.2.1 需求分析概述

1.普通用户：普通用户可上传数据集（最大支持上传5份数据集），用户可在上传的数据集中选择需要分析的数据集进行预览，根据预览界面的数据分析报告生成详细数据探索分析报告，该报告为用户进行数据清洗提供指导方向。若数据集无异常数据，则可通过数据建模界面选择指定列需要的模型。根据用户配置，前端弹出窗口展示生成的数据文件。生成的代码支持导出（不支持运行）。

2.高级用户：高级用户可上传数据集（最大支持上传10份数据集），用户可在上传的数据集中选择需要分析的数据集进行预览，根据预览界面的数据分析报告生成详细数据探索分析报告，该报告为用户进行数据清洗提供指导方向。若数据集无异常数据，则可通过数据建模界面选择指定列需要的模型。根据用户配置，前端弹出窗口展示生成的数据文件。生成的代码支持导出，同时支持。

2.用户：能够访问网站所有页面，且拥有网站所提供的所有的功能，具体如：一、行动日志模块：能够查看每天，每周，每月，每年的待办事项，能够对待办事项的状态和内容进行更改和删除；二、个人中心模块：能够对个人的信息进行更改，比如上传个人头像，更改昵称等，查看以及删除已收藏的总结文章，查看以及删除个人关注，并且可以查看有哪些用户关注了自己；三、总结管理模块：除了可以在论坛页查看所有人的总结外，还可以查看自己所有的总结，以及写总结，对他人总结评论、收藏、点赞、评论，对他人评论进行回复等功能。

### 网站系统流程图

1. 网站系统流程图：

图2-2

1. 各个模块功能介绍图

图2-3

图2-4

图2-5

## 2.3 本章小结

本章就网站的整体架构和功能需求进行了详细的介绍和图解。使用internet应用程序的B/S结构来实现网站，在制作形式上采用前后端分离的方式，详细的介绍了前后端分离的优点以及会出现的问题，通过token-令牌方式解决http的无状态，以及采用CORS的方式来解决跨域问题；后续又通

过流程图的形式来介绍了下整个网站系统的流程，以及各个模块所具有的功能。

# 3 数据库分析与设计-MongoDB

## 3.1 后台数据库以及MongoDB概述

### 3.1.1 Mongodb数据库介绍

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。

### 3.1.2 为什么选择MongoDB数据库

本系统以数据集为核心构造数据挖掘代码，由于不同的数据集拥有不同的列索引以及不同的数据格式，考虑到平台在代码生成/数据预处理过程中会对频繁访问数据集，同时对数据集的不同列进行不同的操作，如果将数据文件以数据库表的形式存储在关系型数据库，则无法遵循传统关系型数据库的ACID规则。

MongoDB 是一个面向文档存储的数据库。文档存储一般用类似json的格式存储，存储的内容是文档型的。这样也就有机会对某些字段建立索引，实现关系数据库的某些功能，同时拥有更好的可拓展性。

## 3.2 数据集合的具体设计

### 3.2.1 数据库设计三范式

什么是范式：简单来说就是，在关系型数据库中的一些规范，范式是关系型数据库的理论基础，也是我们在设计数据库结构中要遵循的规则，以下就是三范式的基本概述

1. 第一范式：数据表中的每一列（字段），必须是不可拆分的最小单元，即确保每一列原子性。
2. 第二范式：在满足第一范式的要求后，要求表中的除主键外的所有字段都必须依赖完整的主键（表中只存在一个主键即可满足第二范式）
3. 第三范式：在满足第二范式的要求后，要求表中的除主键外的所有字段必须直接的依赖主键

遵守数据库设计的三大范式是设计数据表最基本的原则，本网站所运用到的所有数据表都严格遵循了数据库设计的三范式

### 3.2.2 数据库集合设计展示

以下就是本网站系统中所用到的所有数据集合

1. user\_model-用户集合：存储用户信息以及用户上传的数据集列表

表3-1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| \_id | Int(11) | 主键（PRI） | NO | 用户id，主键，  auto\_increment |
| Username | Varchar(20) | 唯一索引（UNI） | NO | 用户名 |
| Nickname | Varchar(30) |  | YES | 昵称 |
| Password | Varchar(40) |  | NO | 密码 |
| Email | Varchar(254) |  | NO | 邮箱 |
| Phone | Varchar(11) |  | NO | 手机号码 |
| Description | Varchar(200) |  | YES | 个人描述 |
| Create\_time | Datetime(6) |  | NO | 创建时间 |
| Update\_time | Datetiem(6) |  | NO | 修改时间 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Login\_time | Varchar(12) |  | NO | 上次登陆时间 |
| Avatar | Varchar(100) |  | YES | 头像地址 |
| Avoid\_login | Tinyint(1) |  | NO | 是否免登录 |

1. User\_follow\_user 中间表：用于存储用户与用户之间多对多的关注关系

表3-2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键，自增长 |
| Follow\_user\_id | Int(11) | 外键（MUL） | NO | 关注用户 |
| Followed\_user\_id | Int(11) | 外键（MUL） | NO | 被关注用户 |

1. topic:总结文章表：存储所有用户的总结

表3-3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Type | Varchar(20) |  | NO | 总结标签 |
| Title | Varchar（30） |  | NO | 文章标题 |
| introduce | Varchar(50) |  | NO | 文章简介 |
| Content | Longtext |  | NO | 文章内容 |
| Created\_time | Datetime(6) |  | NO | 创建时间 |
| Updated\_time | Datetime(6) |  | NO | 修改时间 |
| Like\_count | Int(11) |  | NO | 点赞量 |
| Watch\_count | Int(11) |  | NO | 阅读量 |
| Collect\_count | Int(11) |  | NO | 收藏量 |
| Limit | Varchar(20) |  | NO | 权限 |
| Author\_id | Int(11) | 外键（MUL） | NO | 作者id |
| Message\_count | Int(11) |  | NO | 评论量 |

1. topic\_like\_user 中间表：用来存储文章和点赞用户之间的多对多关系

表3-4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Topic\_id | Int(11) | 外键（MUL） | NO | 文章id |
| User\_id | Int(11) | 外键（MUL） | NO | 点赞用户id |

1. topic\_collect\_user 中间表：用来存储文章和收藏用户间的多对多关系

表3-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Topic\_id | Int(11) | 外键（MUL） | NO | 文章id |
| User\_id | Int(11) | 外键（MUL） | NO | 收藏用户id |

1. topic\_images: 存储文章中所用图片的路径

表3-6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Topic\_image\_url | Varchar(100) |  | YES | 图片路径字符串 |
| User\_id | Int(11) | 外键（MUL） | NO | 用户id |

1. message 评论表：存储总结问章下的所有用户评论

表3-7

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Content | Varchar(200) |  | NO | 评论内容 |
| Created\_time | Datetime(6) |  | NO | 创建时间 |
| Like\_count | Int(11) |  | NO | 点赞数 |
| Reply\_count | Int(11) |  | NO | 回复数 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Topic\_id | Int(11) | 外键（MUL） | NO | 文章id |
| User\_id | Int(11) | 外键（MUL） | NO | 评论用户id |

1. message\_like\_user 中间表:用于存储评论和点赞评论用户的n2n关系

表3-8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Message\_id | Int(11) | 外键（MUL） | NO | 评论id |
| User\_id | Int(11) | 外键（MUL） | NO | 点赞用户id |

1. reply 回复表：存储用户在评论下的回复

表3-9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键 |
| Content | Varchar(50) |  | NO | 内容 |
| Created\_time | Datetime(6) |  | NO | 创建时间 |
| Like\_count | Tinyint(1) |  | NO | 点赞数量 |
| Message\_id | Int(11) | 外键（MUL） | NO | 评论id |
| Parent\_reply\_id | Int(11) | 外键（MUL） | YES | 父级回复 |
| User\_id | Int(11) | 外键（MUL） | NO | 用户 |
| Reply\_count | Int(11) |  | NO | 回复量 |

1. reply\_like 中间表：存储回复和点赞用户间的多对多关系

表3-10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Reply\_id | Int(11) | 外键（MUL） | NO | 回复id |
| User\_id | Int(11) | 外键（MUL） | NO | 用户id |

1. day\_todo 日待办事项表: 存储用户具体日的行动日志

表3-11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 类型 | 索引标识 | 是否为空 | 说明 |
| Id | Int(11) | 主键（PRI） | NO | 主键自增长 |
| Content | Varchar(50) |  | NO | 待办内容 |
| Created\_time | Datetime(6) |  | NO | 创建时间 |
| Level | Varchar(5) |  | NO | 待办等级 |
| State | Varchar(20) |  | NO | 状态 |
| User\_id | Int(11) | 外键（MUL） | NO | 用户id |
| Day | Int(11) |  | NO | 哪一天 |
| Year | Int(11) |  | NO | 哪一年 |

表3.11

12 week\_todo, month\_todo, year\_todo,其表结构和表3.11如出一辙,至于为什么要构建四张表而不是在表中加一个类型字段，主要考虑到一张表数据量可能过大而影响检索速度，不过这样会导致冗余量较大。

### 3.2.3正确设计一张表的结构

在此之前论文已经介绍了数据表设计的三范式，如何具体的设计一张数据表的结构：在遵循三范式的同时，首先应分析存储内容，明确自己要存的是什么；然后确定字段构成，即需要那些字段去存储这些内容；最后设计字段类型，在数据库所支持的所有数据类型中选择合适的数据类型

## 3.3 本章小结

本章从一开始对数据库的发展、基本概念以及类型进行了简单的概述，又对mysql数据库进行了简单的介绍，然后得出为什么本网站选择mysql作为后端存储的数据库，其原因主要有：1开源，2支持多平台，3提供了用于C、C++、Python、Java、PHP等众多语言的api，4存储结构优良，运行速度快，5功能全面丰富

之后，又陈列出本网站在存储数据时所用到的全部的数据表，并且简单介绍了下如何去设计一张关系型数据表：在遵循三范式的同时，1分析存储内容，2确定字段构成，3设计字段类型

# 4 开发前的环境准备

## 4.1 前端程序的开发时环境nodejs

### 4.1.1 node.js简介

Node.js是一个javascript运行时环境，而并非一个js文件，其实质是对ChromeV8引擎进行了封装。

### 4.1.2 node.js的下载

由于我的程序是运行在ubuntu 18.04下的，所以可以直接用一下命令进行安装：

Sudo apt-get install nodejs

同时需要安装nodejs的包管理工具npm, Ubuntu下安装命令如下：

Sudo apt-get install npm

## 4.2 后端程序运行时环境python

### 4.2.1 python简介

Python是一个高层次的结合了解释性、编译性、互动性以及面向对象的编程语言。Python的设计具有极强的可读性，这对于初学者来说相当友好，这也是我选择python作为后端服务语言的原因之一[13]

### 4.2.2 Ubuntu下安装python

在ubuntu下安装python是再简单不过了，我的python用的是3.6.8的版本所以安装如下：

Sudo apt-get install python3.6.8

## 4.3 版本控制工具git

### 4.3.1 版本控制系统的作用

所谓的版本控制系统，从狭义上来讲，它是项目开发过程中用于存储我们所写的代码所有修订版本的软件，但其实我们可以将任何对项目有帮助的文档交付版本控制系统进行管理。

版本控制系统的功能在于跟踪记录整个项目的开发过程，包括项目本身和相关文档，以便于对不同阶段的项目及相关文档进行表示以及进行差别分析，对项目代码进行可撤消的修改，以便于汇总不同开发人员所做的修改，辅助管理和协

调项目开发团队。

### 4.3.2 选择git作为版本控制工具

以git为代表的分布式版本控制工具让每个人在本地都可以有自己的版本库，而不是连接到服务器上的一个公共版本库中去。

### 4.3.3 git的安装及相关命令

1.Ubuntu下安装git

Sudo apt-get install git

2.git原理图

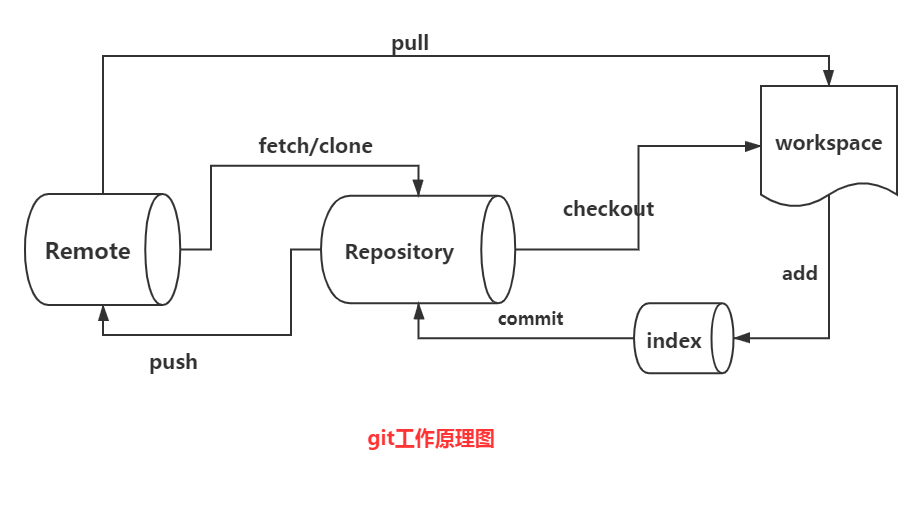


图4-1

3.在我的代码管理流程中用到的git命令

git clone <https://github.com/Zero97-zk/YS.git> 拉取项目代码

git checkout –b T001 创建并切换到T001分支

git add . 将本次工作区的修改添加到暂存区

git commit –m”xxx” 将暂存区的内容提交到本地仓库

git push origin T001 将分支T001的修改推送到远程仓库

登录个人github账户，将T001的修改进行合并

git checkout master 本地切到换分支

git pull 更新本地仓库的代码

git branch –D T001 删除T001分支

git chekcout –b T002 常见并切换到T002分支

以上便是我甚至很多正规企业在项目开发中git的基本流程

## 4.4 本章小结

这一章主要讲了在正式网站开发之前我需要做的准备，先是前端程序运行时环境nodejs的下载,然后是后端程序运行时环境python3.6.8的下载,因为在未部署上线之前之前，整个网站项目的运行流程是先是在前端程序中开启本地8000端口服务，即http://127.0.0.1:8000，然后在前端程序的js代码中发送ajax请求调用后端已经编写好的api接口向后端发送http请求，后端的python运行程序在接收到请求后调用数据库接口从数据库中拿到数据进行整合加工，接着将数据响应给前端程序，最后前端拿到数据动态渲染页面，本地运行简单的流程图如下：

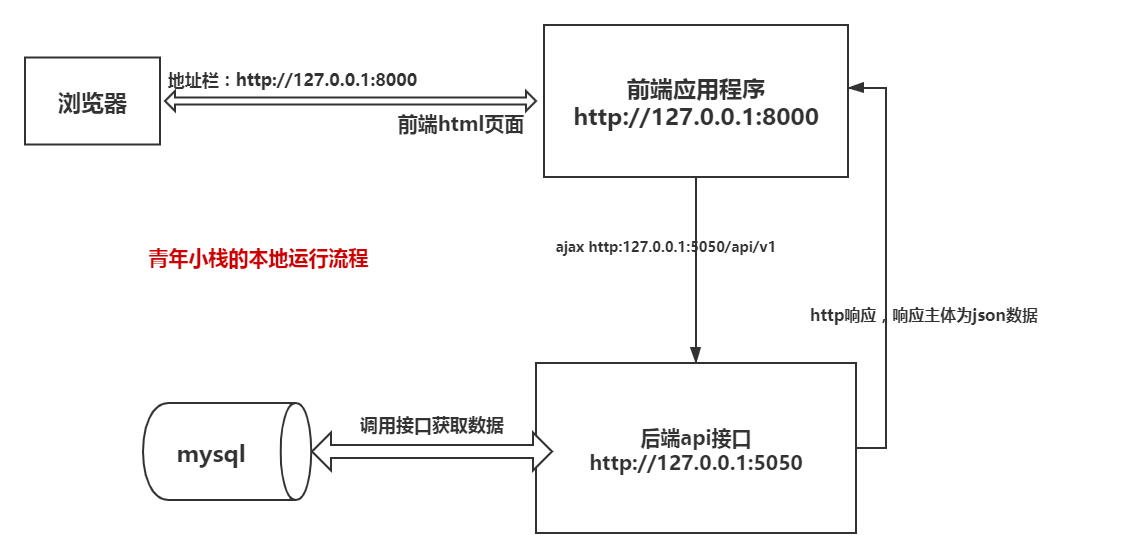


图4-2

在最后又讲述了整个网站开发中很重要的东西—git,分布式版本控制系统，因为在开发过程中，我会遇到很多不是一时半会能解决的问题，可能需要尝试很多遍，但如果每次在编写代码前都需要将项目代码进行拷贝的话，这未免页太浪费时间和空间了，所以就需要git来进行版本控制和代码管理。

# 5 后端api的实现Django

## 5.1 Django简介

### 5.1.1 初识Django

由2005年发布，采用python编程语言编写的开源web框架

Django是一个重量级的web框架，通常配备了以下常用的大部分组件

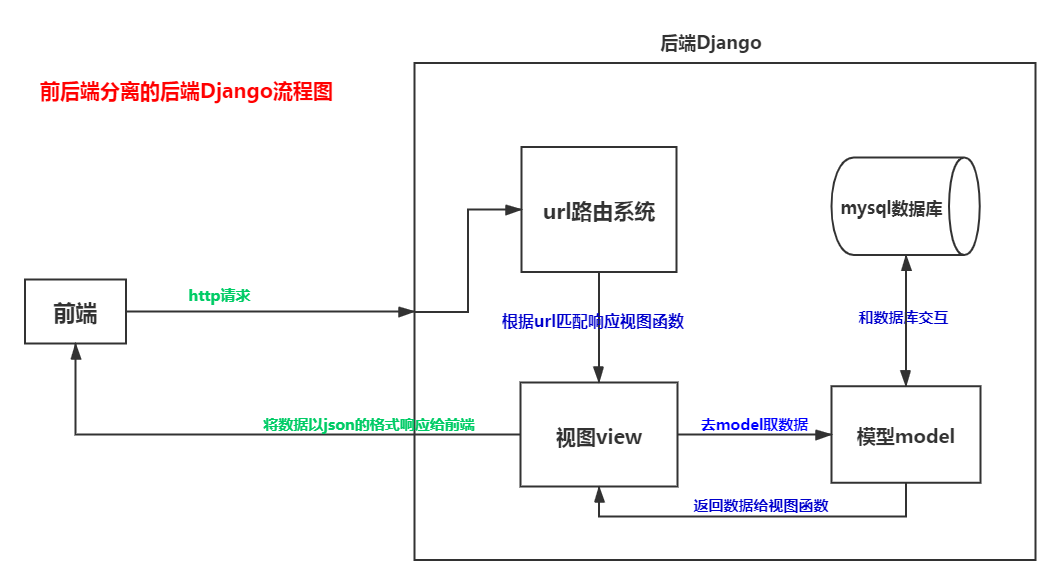
1. 基本配置
2. 路由系统
3. 原生的HTML模板系统
4. 视图view
5. 模型Model，数据库连接以及ORM数据库管理
6. 中间件
7. Cookie和Session
8. 分页
9. 数据库后台管理系统admin

在以上的9中常用的Django组件中，本次网站的后端服务只采用了其中最基本的4种组件，分别是：基本配置、路由系统、视图view、和模型Model

因为是前后端分离的，而后端只需要获取加工数据，并将数据返回给前端，所以用不着Django自带的原生HTML系统，其次网站采用签发token的方式来解决http的无状态特点，所以也用不上Cookie和Session组件，再者网站中并没有管理员一说，只有用户和游客，所以也用不上数据库后台的admin管理系统。[6]

### 5.1.2 前后端分离式的Django运行流程

由于网站采用的是前后端分离式的开发，所以Django的原理图将不再包含从view视图函数去想Template去模板的过程，整个流程即：Django接收到前端的http请求后，将请求交给路由系统，然后路由系统去匹配写好的view视图函数来处理本次请求，视图中一般都会涉及到去model模型取数据，然后model模型再去和数据库交互返回数据给view，最后view视图函数对数据进行处理以json的格式响应给前端，流程图如下：

图5-1

## 5.2 后端程序的全局配置

Django程序需要进行一定的配置才能够正确的使用，而配置文件位于项目文件夹下和项目同名的文件夹下的settings.py中，具体配置如下所述:

1. BASE\_DIR

用于板顶当前项目的绝对路径（此路径是动态计算出来的）,所有文件都可以依赖，取值如下：os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

1. DEBUG

用于配置Django项目的启用模式，取值为：

True：开发环境中使用

False： 项目部署后

1. ALLOWED\_HOSTS

用于设置允许访问到本项目的网络地址列表，取值如下：

[] 空列表便是只有127.0.0.1,或localhost能访问到本程序

[‘\*’]表示所有网络地址都能访问到当前程序

[‘192.168.1.3’]表示只有192.168.1.3这个主机能访问当前项目

1. INSTALLED\_APPS

用于指定当前项目安装的应用列表，即当用命令创建app时需要在

这里进行注册

1. DATABASES

用于指定数据库的配置信息，以极其简单的配置能后让model模型和大多数数据库进行交互，其中连接mysql的数据库配置如下：

DATABASEES={

‘default’:{

‘ENGINE’:’django.db.backends.mysql’,

‘NAME’:’ys’,

‘USER’:’root’,

‘PASSWORD’:’XXXXXX’,

‘HOST’:’127.0.0.1’,

‘PORT’:’3306’

}

}

1. ROOT\_URLCONF

用于配置根基url

1. 添加mysql支持

1>.sudo pip3 install pymysql

2>.修改项目中的init.py文件加入如下内容提供pymysql引擎支持

import pymysql

pymysql.install\_as\_MySQLdb()[7]

以上基本上是在项目文件夹创建完成后根据网站需要做的一些配置改动，以下是之前提到过的用来解决前后端分离造成的跨域问题的配置：

<https://blog.csdn.net/liuanpingfirst/article/details/86684029>

## 5.3 路由系统及视图函数绑定

### 5.3.1 根级路由和应用子路由

当django接收到请求的url后应该转给什么应用的什么view函数来进行处理就需要用到django自带的路由系统，整个路由系统有主路由和子路由组成，

主路由在项目同名文件夹下的urls.py文件中，如下图：



图5-2



图5-3

### 5.3.2 view视图函数接收请求和作出响应

图5.4、5.5中的utlpatterns是一个路由-视图函数影戏的列表，具体的对应关系有url函数来确定，url（）函数所调用的模块以及语法如下：

模块： from django.conf.urls import url

语法： url(regex, views, name=None)

参数：

1>.regex：字符串类型，允许使用正则表达式

2>.views：指定路径对应视图处理函数的名称

3>.name：为地址起别名。（模板中使用，本网站基本不需要）

## 5.4 和数据库的交互

### 5.4.1 模型（Models）基本概念

模型是一个Python类，是由django.db.models.Model派生出的子类.

一个模型类代表数据中的一张数据表，模型类中的每一个类属性都标数据中的一个字段。模型是数据交互的接口，是操作数据库的方式。

### 5.4.2 Django自带的ORM

1. ORM概念（Object Relational Mapping）对象关系映射，这是一种程序技术，它允许你使用类和对象的方式对数据库进行相应操作。

2. ORM的作用：

1>.建立模型类和表之间的对应关系，通过面向对象的方式操作数据库

2>.根据设计的模型类来生成相应的数据库中的表

3>.可以通过简单的配置来实现不同的数据库

3. ORM的好处:

1>.实现了数据模型与数据库的解耦，屏蔽了不同数据库操作上的差异

通过简单的配置就可以更换数据库，不需要修改代码

2>.通过面向对象变成操作数据库，不需要面向数据库编写代码

对数据库的操作都转化程对类属性和类方法的操作

4. ORM与数据库的对象关系映射图如下：

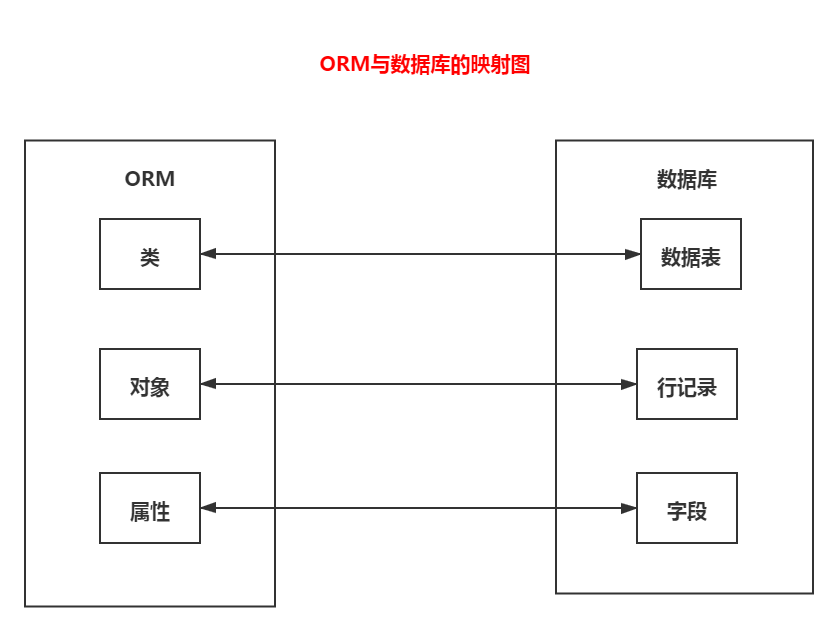


图5-4

### 5.4.3 以模型的设计来实现数据表的创建

文章总结模型（Topic）如下：



图5-5

对应生成的mysql数据表结构如下：

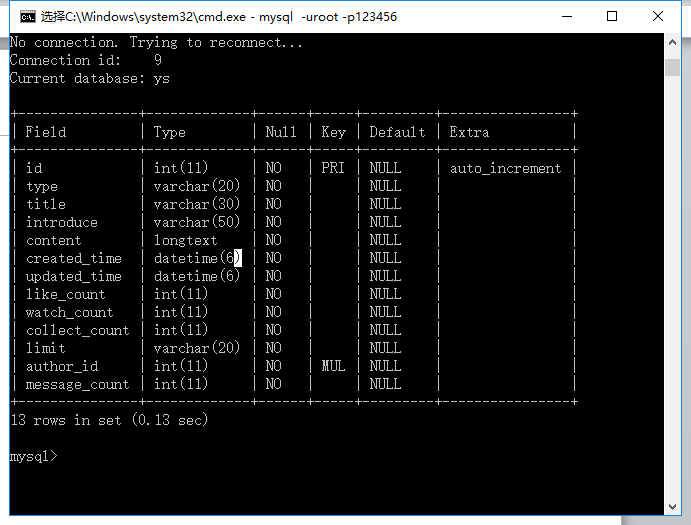


图5-6

### 5.4.4 通过ORM来完成数据库CRUD

CRUD是指在计算处理时的增加（Create）、读取（Read）、更新（Update）和删除（Delete），每个集成自models.Model的模型类，都有一个objects对象会被童颜继承，这个对象叫做管理器对象，数据库CRUD可以通过模型的管理器对象来实现，例如Topic的管理器对象可以用Topic.objects来表示，如下是通过管理器对象来实现的数据库的增删改查操作,以Topic为例：

1. 创建数据记录：

Topic.objects.create(属性1=值1，属性2=值2...)，成功返回创建好的实体对象，失败则会抛出异常

2. 查询数据记录：

在查询数据方面，管理器对象有很多的查询接口，Topic.object.all()

all()查询所有记录，返回一个QuerySet查询对象；get()查询符合条件的单一记录，该方法只能返回一条数据，如果有多余的数据则会抛出

Model.MultipleObjectsReturned异常,如果没有查询到数据则会抛出Model.DoseNotExist异常;filter(属性1=值1，属性2=值2)查询符合条件的多条记录，返回QuerySet容器对象,内部存放MyModel实例.

3. 修改数据记录:

记录的修改分文单个对象以及对QuerySet批量修改，对于单个对象需要先通过get（）方法得到要修改的实体对象，让后用 对象.竖向的方式进行修改；对于QuerySet的批量修改，需要先得到QuerySet记录集，然后用QuerySet.update(属性=值）的方式实现。

4. 删除数据记录：

删除单个模型对象或删除一个查询结果集（QuerySet）中的全部对象都是调用delete()方法，例如：删除单个记录: Topic.objects.get(id=1).delete()

## 5.5 本章小结

Django是一个非常重量级的pyhton web框架，这一章主要介绍了本网站在实现后端api编写过程的中运用的一部分的Django框架技术。先是介绍了Django的基本常用组件，然后再陈述了本网站主要运用的4个组件：基本配置、路由系统、视图view、Model模型。通过这四个组件来实现我想达到的目的，就是在前端发送ajax请求到后端后，通过路由系统找到响应的view函数对其进行解析，然后通过和数据库进行交互得到数据，最后将数据处理加工以json的格式响应给前端。

需要值得一提的是，本网站后端的所有的API设计基本符合RESTful的设计理念。后端的每一个URL接口都表明了api以及版本信息，路径中尽量避免了动词的出现，每个名词都代表着一种的资源；在表现层上以统一的json的格式进行响应；每一种请求方法带代表着响应的操作，GET用来获取资源，POST用来在服务器中创建资源，PUT用来更新服务器资源，DELETE用来从服务器上删除资源。

# 6 前端页面实现Vue

## 6.1 页面技术点分析

### 6.1.1 页面开发三要素

html、css、javascript是web网页开发最基本的要素，其中html作为网页内容的载体，html包含了用户需要浏览的内容，包括文字、图片等，同时，html也是网页的基本机构，可以用来做各种布局；其次css是网页样式的表现，比如：标题字体、背景图片、表框、字体颜色以及复杂一点的变换、过渡等；最后javascript用来实现网页中用户的行为和动作，如果一个网页中只有元素和样式，那么它和一张图片就没什么区别了，这样无法形成良好的用户体验。好的用户体验需要让用户在网页上能够做出非常舒服的人机交互。

### 6.1.2 简化样式的编写bootstrap

在本次网站的页面编写过程中引入了bootstrap的样式库来简化样式的编写，

bootstrap.css是一个集成了很多已经编写好的样式的css文件，bootstrap最令人满意的便是它的响应式布局，但遗憾的是在本次网站开发中并没有运用boot的响应式布局，仅仅是用了一些写好得样式而已，不过值得一提的是在本次的网页中并没有运用精灵图的图标样式，而是引用的来自Glyphicon Halfingsde的字体图标，因为虽然Glyphicons Halfings是收费的，但他们的作者允许Bootstrap免费试用，这就为本次的开发在一些小图标上提供了不少的帮助。

### 6.1.3 使用vue来构建用户界面

vue是一个第三方开发的渐进式的基于MVVM模式的纯前端js框架，它可以在香满园中逐步使用vue的各部分组件和功能，同时亦可以和其他的技术轻松的混搭在一起，并且只需要浏览器就可以运行，不需要任何后端技术的支持，由于其MVVM的设计模式，vue框架在DOM操作上进行了终极简化，自动化开发。MVVM设计模式和旧的前端项目进行比较如下：

旧的前端项目分为三部分：

1.HTML：的定义页面的结构和内容

2.css：定义页面中的内容的样式

3.js：添加页面行为和动作

问题：因为HTML和CSS是静态的，多有的动态变化都需要依赖js来添加，这会导致js的负担非常重要：重复的代码和操作太多。

MVVM重新划分的前端项目：

1.界面（View）：HTML+CSS：支持变量、循环以及分支等程序结构要素

2.数据模型（Model）：一个用来保存所有页面所需变量的对象-data

3.视图模型（ViewModel）：将数据模型（Model）和界面（view）绑定起来，以此来达到数据模型中值如何变化，界面自动跟着变化。[8]

### 6.1.4 spa技术

所谓的spa即单页面web应用（single page web application），整个web应用程序中只有一张页面，通过用户与应用程序的交互来动态更新页面的Web应用程序，在此期间不会重新加载页面。而单页面应用的优点就在于能够让用户有更好的体验，提升用户在于web程序交互式的速度的流畅度；单页面应用程序最根本的优点就是它的高效性，它消耗更少的快带，减小对服务器的压力，也能够于各种架构更好的结合。这里对如何实现spa技术不再详解，主要就是编写不同的页面组件，根据vue-router对象来进行页面组件的切换来达到页面显示不同内容的效果。[9]

### 6.1.5 实现ajax请求-axios

染页面前，需要先访问后端获取数据，在拿到数据后才能根据数据动态的

渲染页面，如何从后端获取数据，就需要在前端用js编写ajax请求来调用后端已经编写好的api接口，以此获取后端响应数据，但用原生的ajax发送请求到获取响应整个过程太过繁琐，且代码量又多，此时需要调用别人已经写好的第三方库axios来对整个流程进行简化。原生的ajax请求流程如下：

1.建立MLHttpRequest对象

2.打开一个连接

3.请求头信息（get可以不用设置）

4.发送请求

5.监听响应

使用axios发送ajax如下：

1.发送get请求，axios.get(url,[config])

2.发送post请求,axios.post(url,[data,config])

3.发送put请求，axios.put(url,[data,config])

4.发送delete请求,axios.delete(url,[config])

由此可以看出axios对ajax进行了终极的封装，这使得前端发送ajax请求变的相当简单。

## 6.2 页面的具体实现

### 6.2.1 前端项目目录结构

本网站的基本的vue脚手架目录结构是使用webpack进行搭建的，搭建项目目录结构<https://blog.csdn.net/Smell_rookie/article/details/91334847>



图6-1

### 6.2.2 各个页面组件的实现效果

1.网站首页显示（登录、注册页）



图6-2

2.网站行动日志页显示效果



图6-3

3.网站写总结页展示效果

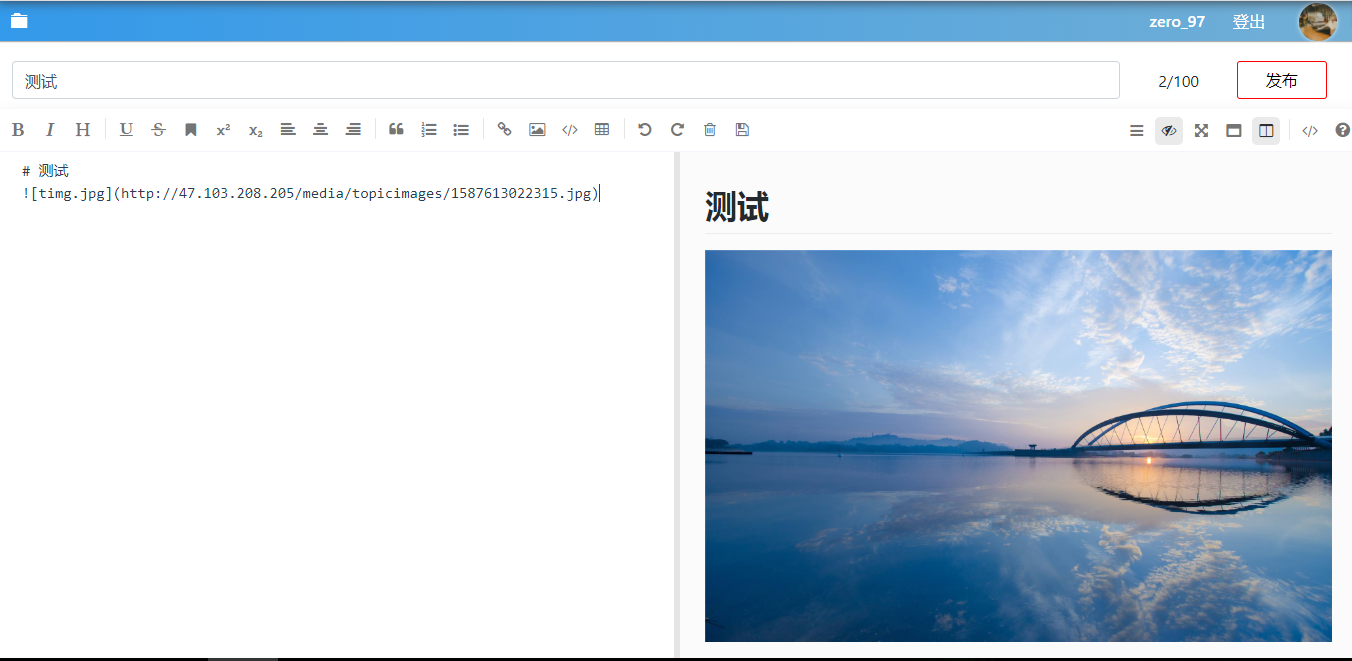


图6-4

4.个人总结页展示效果

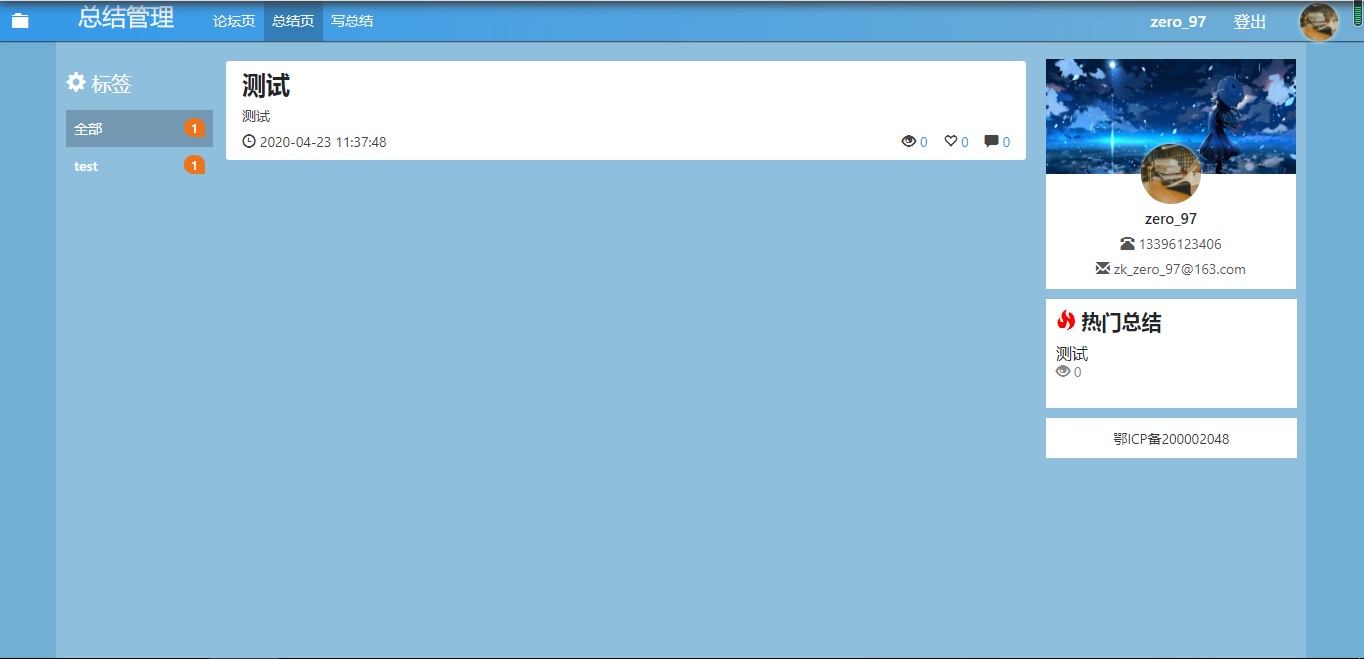


图6-5

5.总结论坛展示效果

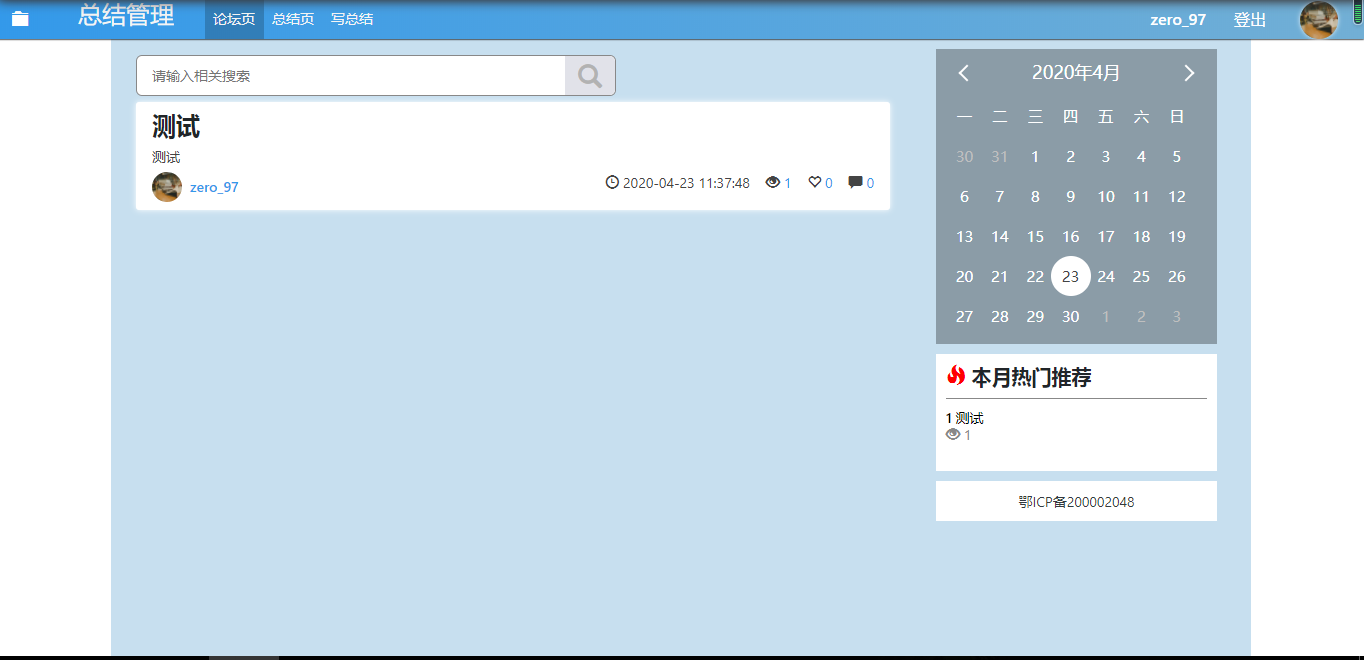


图6-6

6.具体文章页展示效果

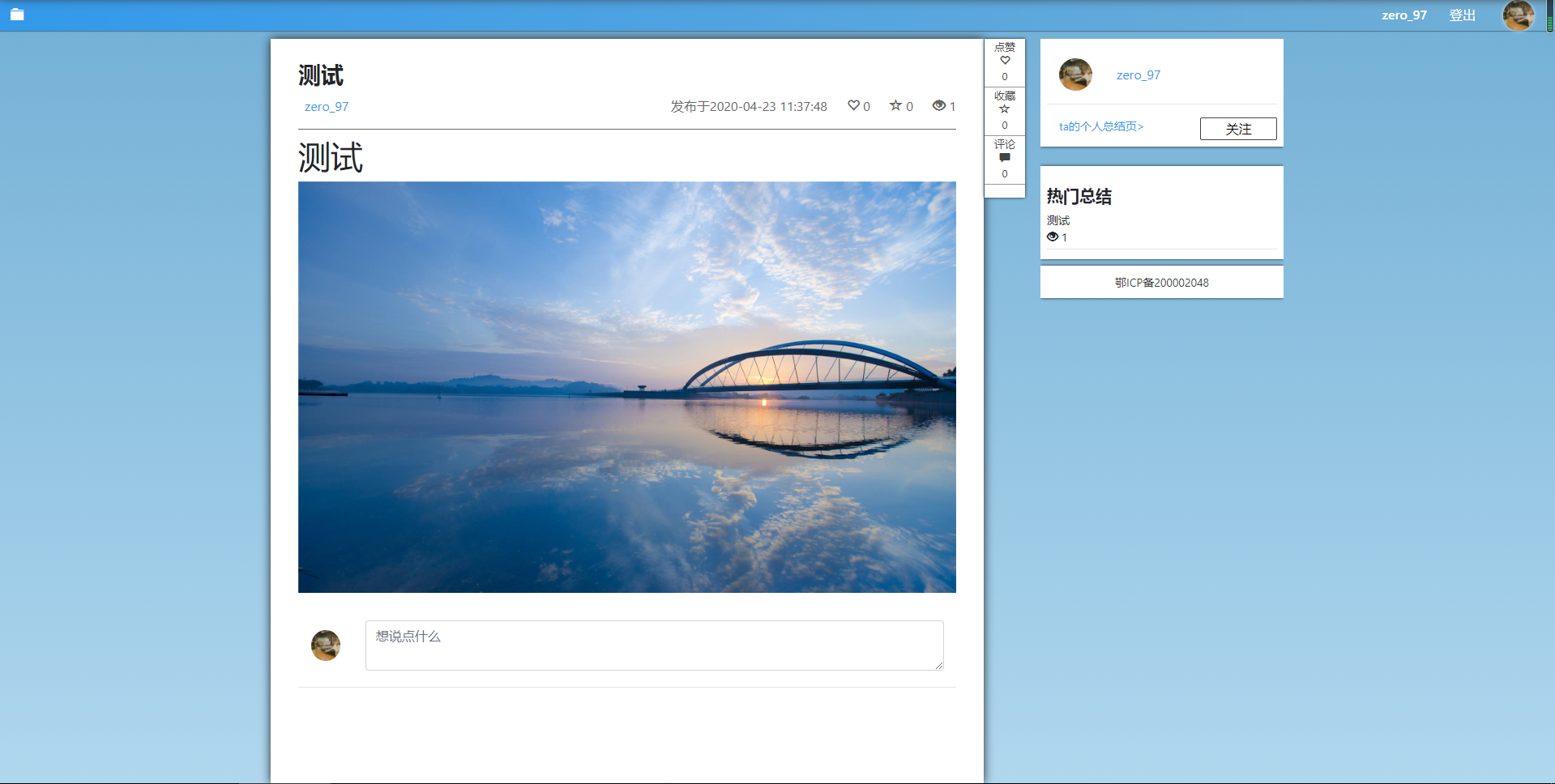


图6-7

7.评论效果展示



图6-8

8.个人中心页效果展示

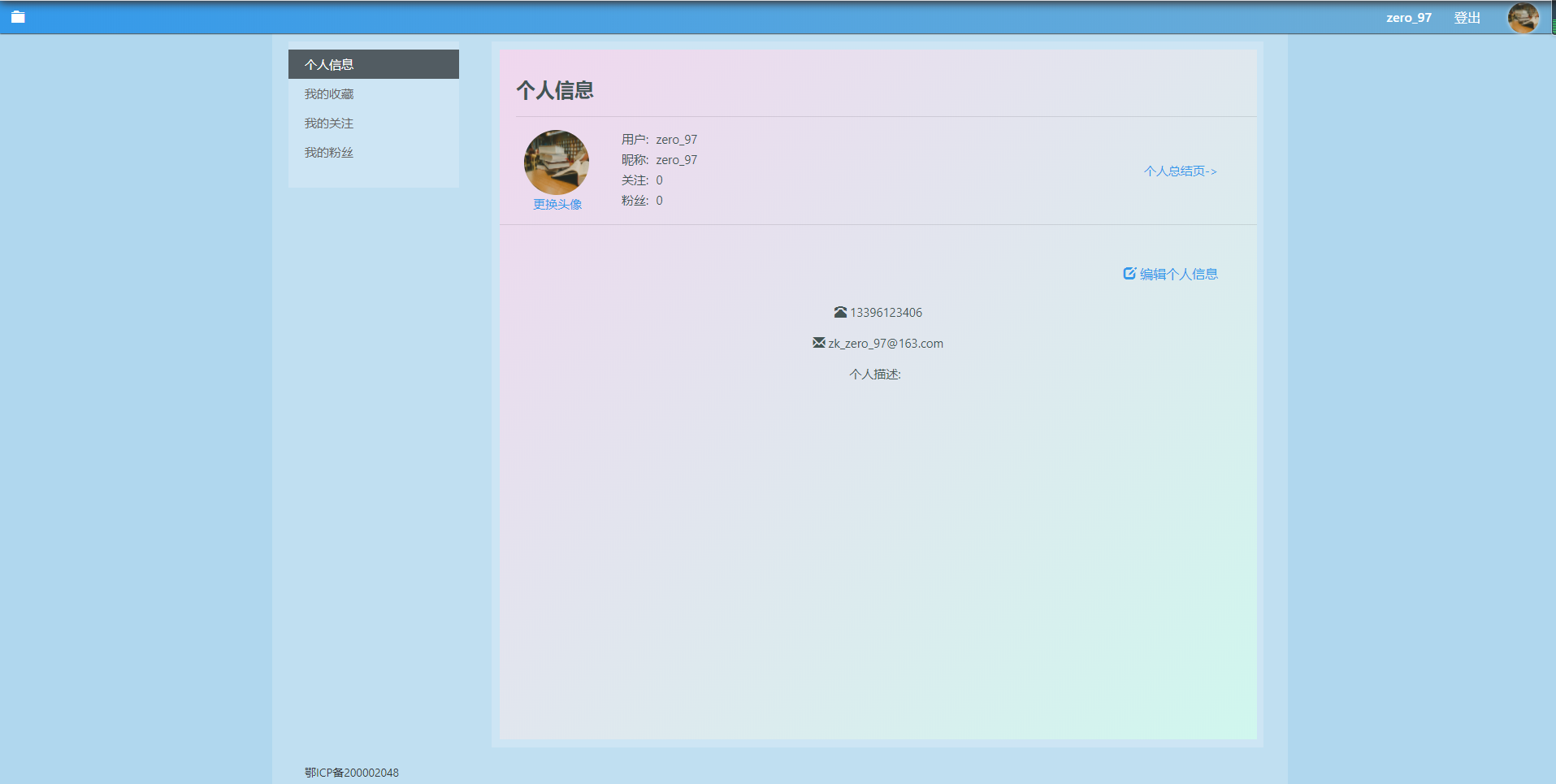


图6-9

9.导航栏效果展示

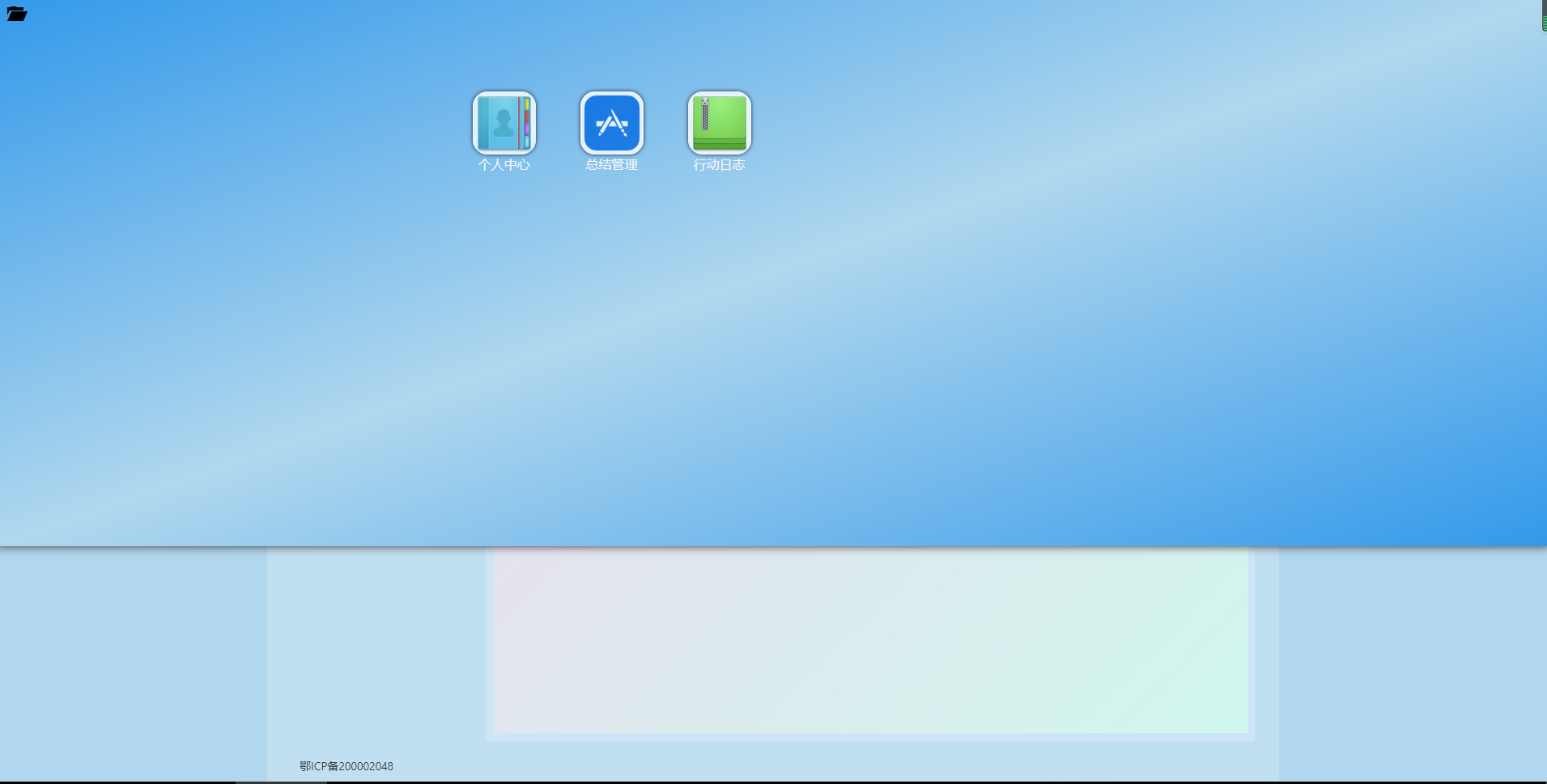


图6-10

## 6.3 前端页面完成后的测试

在前端页面完成后，整个网站的制作基本已经进入到尾声，此时需要对页面上的交互效果进行测试，首先是登录首页，然后注册一个新的账号，发现没问题后，根据程序功能会直接跳转到行动日志页，此时对行动日志页上的相关操作行进测试，添加一个今日待办，修改今日待办，删除今日待办，周待办，月待办，年待办也进行相关操作，发现都没问题后点击导航进入总结管理模块，会直接跳转到个人总结页，因为是新的账号，所以并没有相关总结，点击写总结按钮进入到写总结的页面，对写作功能进行测试，写作完成没问题后点击发布，浏览器会新开一个总结文章页，在文章页进行收藏、点赞、评论以及回复，测试过程中发现都没有问题，最后通过导航来到个人中心模块，在此上传头像，修改信息，查看和取消收藏，以及查看和取消关注。这基本是整个网站的功能流程，对于没有注册的游客来讲，只能访问到总结论坛页，查看一些他人的总结。

## 6.4 本章小结

这一章先是简单介绍了本网站（青年小栈）在前端页面程序编写过程中所用到的技术手段，有基础的html，css和javascript、boot样式库、vue语法、spa单页面应用以及用来发送ajax的第三方库axios。在介绍完技术手段后又展示整个项目的目录结构和最终完成的页面效果，最后以新用户的角度在本地进行测试。整个章节并没有提到任何的代码层面上的东西，并不是意味着前端的代码都很轻松，恰恰相反，往往最简单的样式可能就要研究很长时间，试不同的代码样式，因为在前端页面编写前我并没有事先对页面到底什么样进行设计，这导致了我在进行页面编写的过程中完全凭着自己想象走或者是试各种样式找出自己最满意的，不过比较欣慰的是这个网站本来就是我需要用的，所以我能大致想象出它大概是什么样，开工之前先做出设计这个问题其实很重要，这个很影响整个项目的流程开发，这也算是吃一堑长一智吧。

# 7 网站的上线部署

## 7.1 上线前的准备

在上线前我得知道为什么我需要让我的网站上线，因为如果不上线的话，网

站每次就只能在自己的电脑上通过指定的端口进行登录，这样就没人能够访问到我的网站，并且本机不在的话自己也没办法登录网站，在网站上线后，任何人就可以在任何有网的地方进行访问，这也正是我想要达到的目的，不过上线之前我需要准备一些东西：

1.购买远程服务器，我需要将我的项目跑在上面然后通过公网ip进行访问，因为大多数我们的网络都并非公网ip，即没办法通过网络进行访问。

2.域名，有了域名我才有了通过一个独有的名称来访问网站的前提

3.备案，只有将域名备案后，域名才有效。

4.将远程服务器的ip和域名绑定，这样就能通过域名来访问服务器。

## 7.2 上线后的网站流程分析

在网站上线后那么整个网站的访问流程就需要进行一定的改变，因为远程服务器的默认端口是80，所以就需要一个端口为80的web服务器来对整个网站程序进行代理，本网站的web代理服务器用的是nginx，nginx是一款开源的、高性能的HTTP服务器和反向代理服务器。上线后的网站流程图如下：[11]

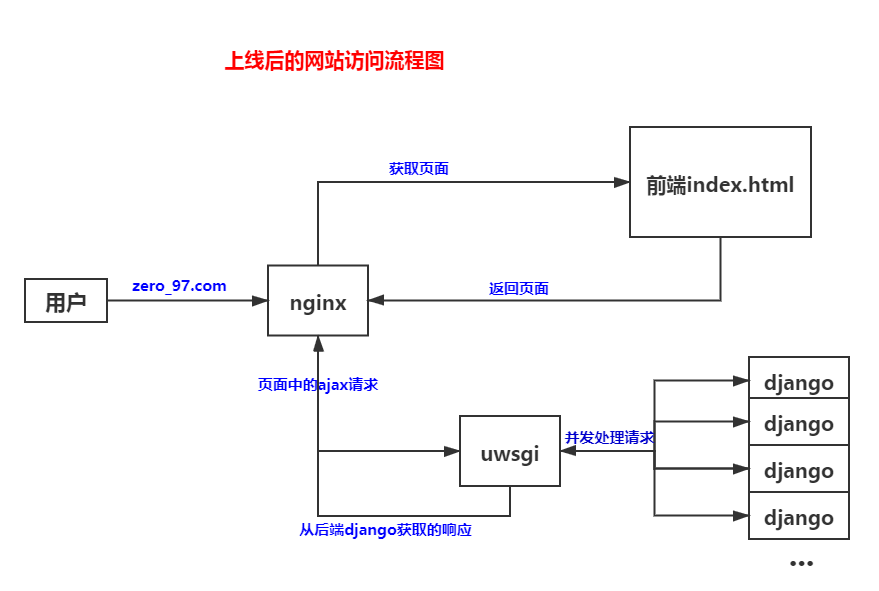


图7-1

## 7.3 正式部署

### 7.3.1 前端vue的部署

1. vue项目打包

npm run build

2. 将打包完成后生成的dist文件夹下的index.html配置到nginx下，以便在访问域名时将页面返给用户,配置如下:

修改nginx的配置文件 /etc/nginx/sites-available/default

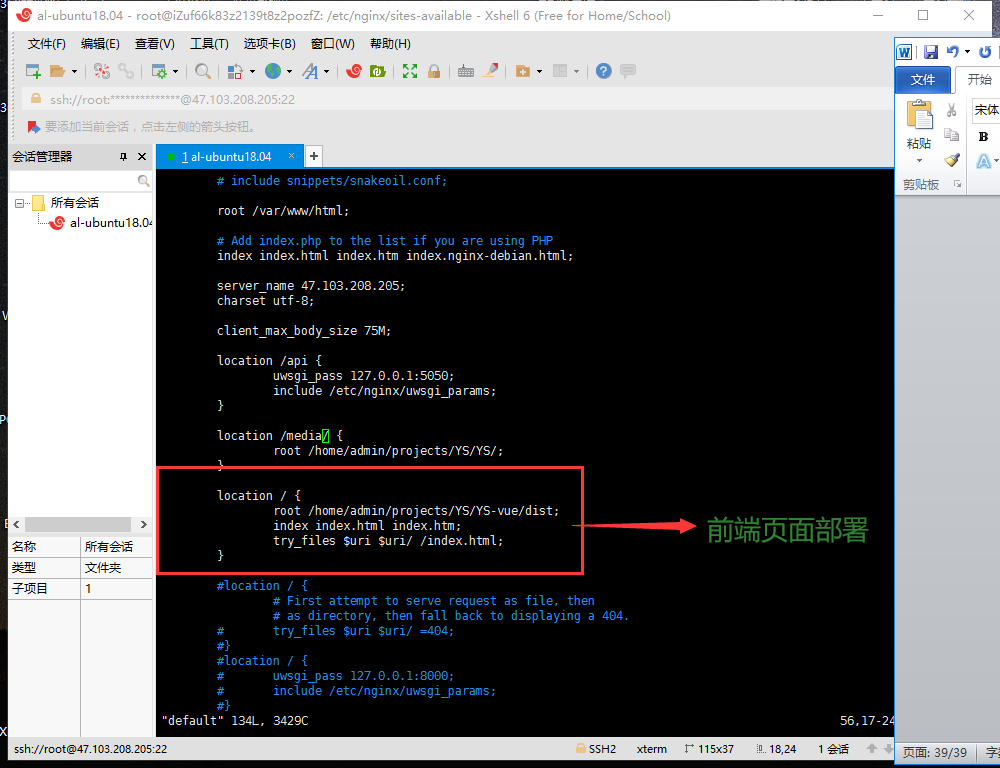


图7-2

### 7.3.2 后端部署

1. WSGI Django工作环境部署

WSGI Web服务器网关接口，是Python应用程序或框架和Web服务器之间的一种接口，被广泛使用。之前使用的python manage.py runserver只是在开发和测试环境中使用，项目上线时完善的代码需要在一个稳定高效的环境中运行，这时可以使用uwsgi，它可以让django等开发的web站点运行在其中，ubuntu下使用sudo pip3 install uwsgi来进行安装，配置文件需要建在后端项目主文件夹下的一个.ini文件中.[10]

启动：sudo uwsgi --ini 项目文件夹/uwsgi.ini

停止：sudo uwsgi –stop uwsgi.pid

uwsgi.ini文件具体配置如下：

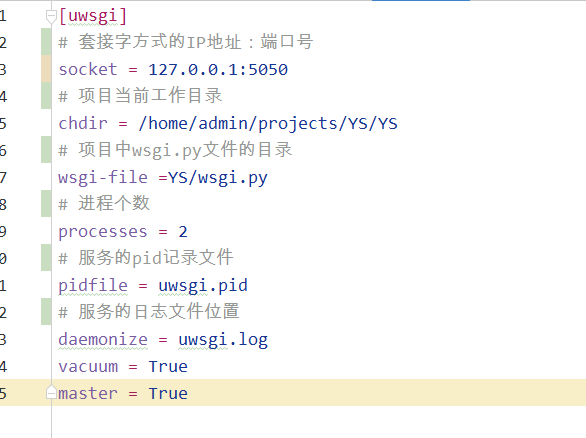


图7-3

2.nginx配置文件中配置uwsgi接口

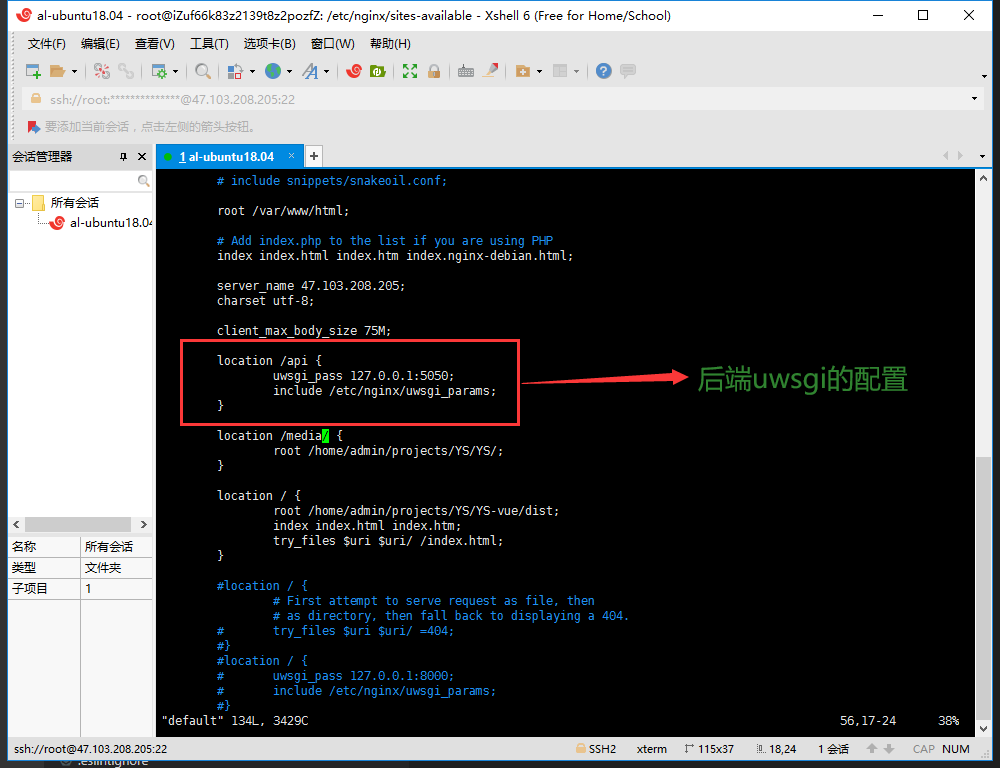


图7-4

### 7.3.3 配置静态文件路径

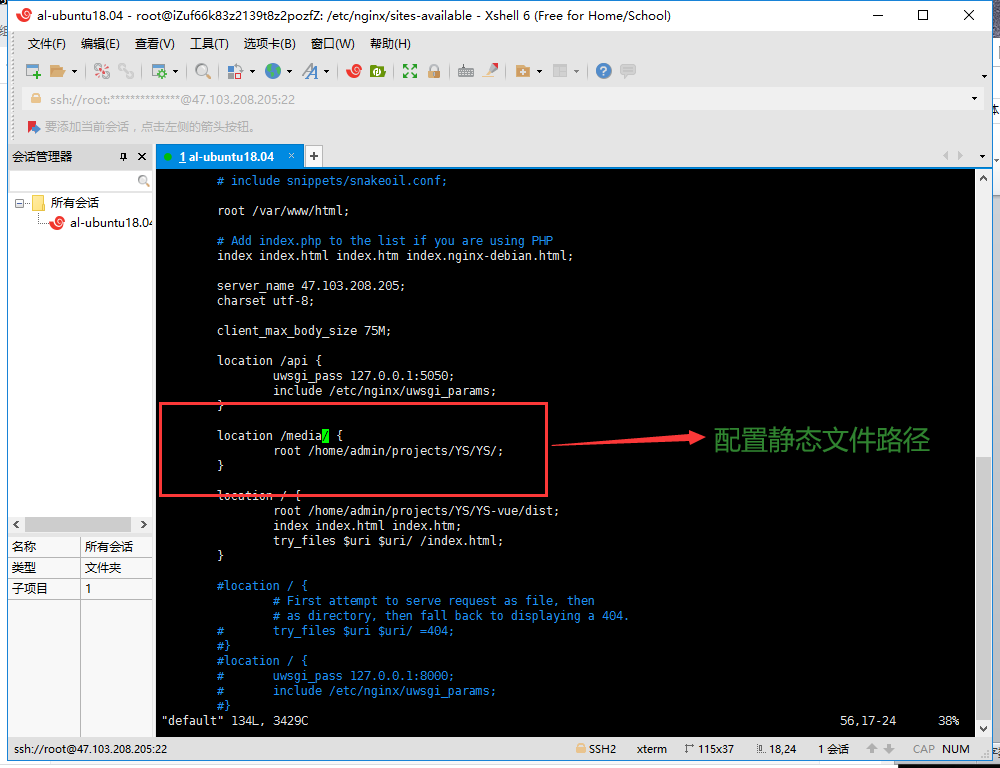


图7-5

## 7.4 总结

在一切配置都完成后，启动uwsgi，重启nginx，到此为止，网站已经可以开始访问了，我自己先去网站以用户的身份走了一趟流程，没有发现什么漏洞，当然一个应用刚诞生在使用的过程中肯定会出现各种的bug，不过对于开发者来说这肯定都不是问题。

以制作网站作为最后的毕业设计主要有两点重要的原因，一是我个人需要一个用来记录我自己行动日志的平台，否则就感觉每天就跟混日子一样不知道干什么，有一个明确的目标做起事来会事半功倍；再者就是我目前就是从事后端工作的程序员，借此来磨练自己的技术、学习更多的知识以及积累经验。我也是在做网站的时候刚接触的js一些比较核心的概念，以及boot、vue都是现学的。

在整个做网站的过程中确实遇到了很多的问题，比如在做文章评论功能的时候我并不知道我的后端要返回一个什么样的数据结构的数据，前端要根据什么样

的数据才能把评论功能做的像酷狗音乐的评论那样，后来经过思考还是做成了自己想要的样子，虽然还是欠缺了些。本次毕业设计最大的收获并不是我做出了一个网站，也不是在开发的过程中我解决了哪些问题，让我感觉收获最大的是我了解到了更多解决问题的方式以及解决问题的经验，还有就是对心性的锻炼在这次做网站的过程中是更加突出的，在一开始做前端页面的时候有好些次都想放弃vue框架直接用最基础的html、css以及js来写，还好经过长时间的摸索坚持下来了，这才让我见识到vue框架各种神奇的思想和功能。

最后在完成网站的上线后，回头望去才发现，原来并没有什么难以理解的东西，只是需要花时间去想、去做罢了。

# 参考文献

[1] 杜现朝,龙怀冰,王清理 等. 基于Web标准的网站设计与开发流程[J]. 计算机工程与设计. 2008-02(28):945-948

[2] 华迎. 网站可用性测试设计[J]. 现代教育技术. 2008-04(01):115

[3] 郑岚. Python访问Mysql数据库[J]. 电脑编程技巧与维护. 2010-03(18):59-61

[4] 姜承尧. Mysql技术内幕:innoDB存储引擎[M]. 北京. 机械工业出版社. 2011

[5] 王建,罗政,张希等. Web项目前后端分离的设计与实现[J]. 软件工程. 2020-04(05):23

[6] 王冉阳. 基于Django和Python的Web开发[J]. 电脑编程技巧与维护. 2009-01(18):56-58

[7] 刘志凯,张太红. Django框架在web开发中的应用[J]. 农业网络信息. 2015-02(26):51-52

[8] 麦冬,陈涛,梁宗湾. 轻量级响应式框架Vue.js应用分析[J]. 信息与电脑（理论版）. 2017-04(08):58-59

[9] 邓雯婷. 基于Vue.js构建单页面GIS应用的方法研究[J]. 科技创新与应用. 2018-05(18):5-10

[10] 王伊,王韶红,刘晋泽等. Vue.js与Django组合框架的网络社交系统单页面架构方案设计[J]. 信息技术与信息化. 2020-02(10):121-123

[11] 田纯青. 利用Nginx实现基于URL的Web负载分配[J]. 现代计算机（专业版）. 2009-07(25):187-191

[12] Šifrer, Tomaž. IMPLEMENTATION OF A WEB CONTENT MANAGEMENT SYSTEM USING THE DJANGO FRAMEWORK[J]. Journal of Behavioral and Experimental Finance,2019

[13] Sheetal Taneja, Pratibha R.Gupta. Python as a Tool for Web Server Application Development[J]. JIMS 8i-International Journal of Information, Communication and Computing Technology(IJICCT),2019

# 致 谢

光阴荏苒，转眼间，四年的本科生涯即将结束，不就我就要从武汉纺织大学这里毕业了、踏上人生新的旅程。回首在母校的4年时光，不禁感慨万分，在这四年里，母校给了我太多美好的回忆，挥手作别之前，我想向一些在大学中认识的人表示真挚的谢意。

首先，感谢这四年来所有教授我知识的老师们，各位老师道德与学术并重，渊博的知识、谦逊的为人让我倍感温馨。永远难忘老师们所传授的理论知识和技术，也难忘老师们在我人生观、价值观上的指导。在此，向各位老师鞠躬致谢。

其次我要感谢我的父母，感谢你们对我这些年来的养育之恩，感谢你们对我这些年的支持。在今后的工作生活中，我一定会努力做一个对社会有用的人。

然后我要感谢一起同窗四年的室友，感谢各位四年来的照顾，是你们让我的大学生活变得更加丰富。和你们一起生活的日子，不管过多久我都不会忘记。

最后我要感谢母校的栽培，感谢数计学院。