第一章

1.1 项目背景

1.2 主要研究内容

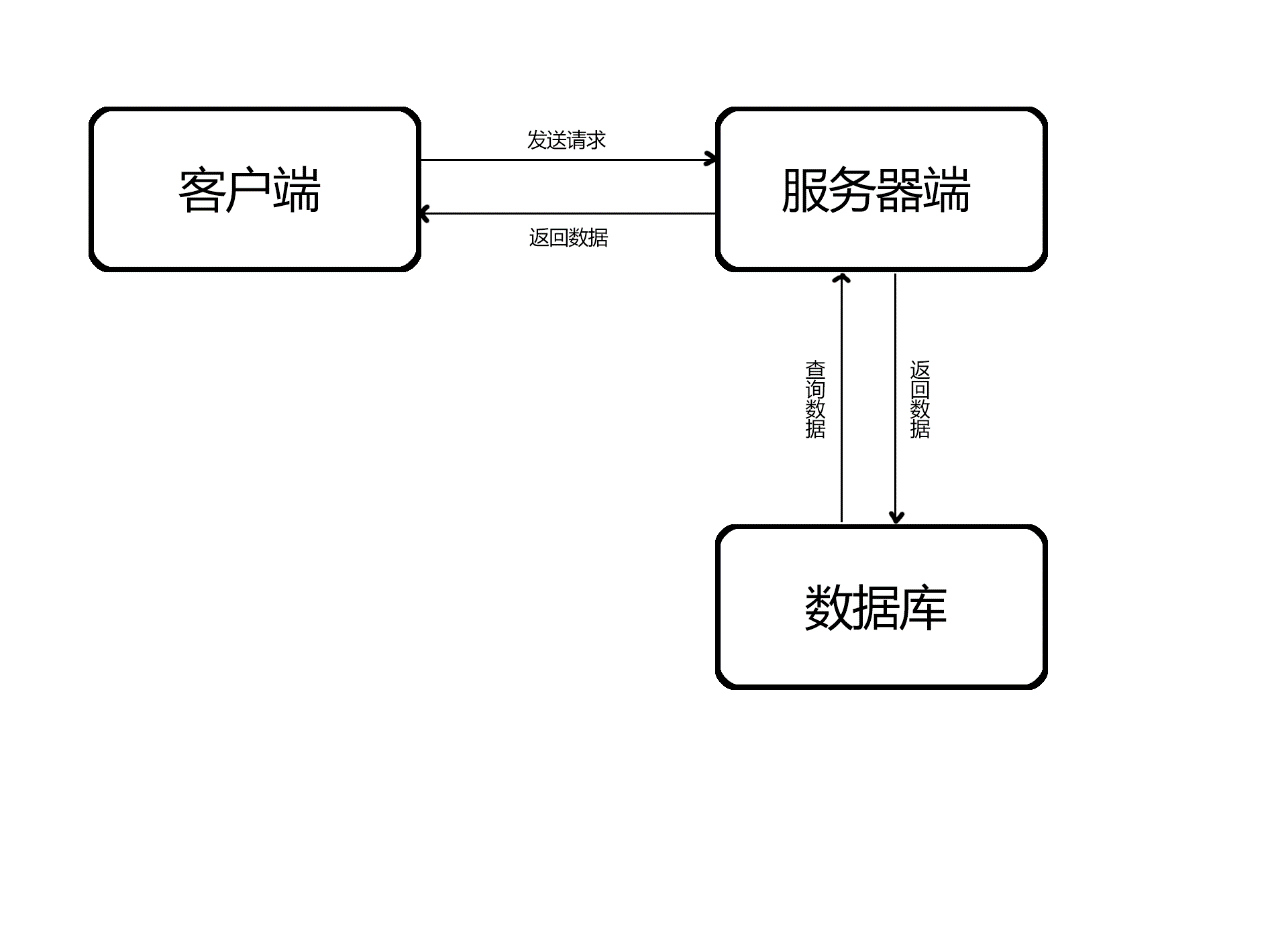
1.3 组织架构

系统分析

1.3.1可行性分析

1.3.4 数据流图

Web 应用分为客户端和服务器端，因此数据在客户端（即浏览器）、服务器（即 Node 服务器）以及数据库（即 MongoDB）之间流动；



第二章 系统关键技术 //本章建议不超过6页，多进行文献引用

该系统基于是 B/S 架构的应用，主要开发语言为 JavaScrpt,使用 Vue.js 搭建前端 View 层及 Controller 层，

Stylus 对 CSS 样式进行预处理，webpack 对静态资源进行打包、管理；express 和 nodeJs 结合搭建后端 Model 层，进行业务处理等，使用非关系型数据库 MongoDB 及其框架 Mongoose 对数据进行管理。

2.1 Vue.js //以简练的语言描述它的特点以及选用该技术的原因

2.2 Vuex

2.3 axios

2.4 Stylus //以简练的语言描述它的特点以及选用该技术的原因

2.5 Echart

2.6 Webpack

伴随着移动互联的大潮，当今越来越多的网站已经从网页模式进化到了 Webapp 模式。它们运行在现代的高级浏览器里，使用 HTML5、 CSS3、 ES6 等更新的技术来开发丰富的功能，网页已经不仅仅是完成浏览的基本需求，并且Webapp通常是一个单页面应用，每一个视图通过异步的方式加载，这导致页面初始化和使用过程中会加载越来越多的 JavaScript 代码，这给前端开发的流程和资源组织带来了巨大的挑战。

前端产品的交互是基于浏览器，资源通过增量加载方式运行到浏览器端，在开发环境组织好这些碎片化的代码和资源，并且保证它们在浏览器段快速、优雅的加载和更新，就需要一个模块化系统。

模块系统主要解决模块的定义、依赖和导出。

已有的模块系统：

1. <script> 标签

最原始的 JavaScript 文件加载方式，弊端也显而易见：容易造成变量命名冲突；文件只能按照 <script> 标签的书写顺序加载；开发人员需要手动解决文件之间的依赖关系；随着项目规模增大、时间积累，代码库容易变得混乱不堪。

② CommonJS

该规范的核心思想是模块通过 require 方式同步加载所依赖的其他模块，通过 exports 或 module.exports 导出指定接口。但是，在浏览器资源是异步加载的，不适合同步模块加载的方式；不能并行加载多个模块；

③ AMD

虽然可以在浏览器中异步、并行加载多个模块，但却提高了开发成本， 模块定义方式语义不顺畅，代码阅读和书写比较困难

上面提到的模块仅仅是指 JavaScript 模块文件，但在前端开发中，还涉及到 CSS、图片、字体、HTML 模板等众多资源。如果这些资源都可以视作模块，通过 require 的方式加载，将带来优雅的开发体验。

Webpack 是前端资源模块化管理和打包工具。它可以将许多松散的模块按照依赖和规则打包成符合生产环境部署的前端资源。还可以将按需加载的模块进行代码分隔，等到实际需要的时候在异步加载。在编译的时候，对整个代码进行静态分析，分析出各个模块的类型和依赖关系，将不同类型的模块交给适配的加载器 loader 处理。通过 loader 的转换，任何形式的资源都可以视作模块，如 CommonJs 模块、AMD 模块、ES6 模块、CSS、图片、JSON、Coffescript、LESS 等。

2.7 Node.js

Node.js是基于Chrome V8引擎的JavaScript运行环境，V8 引擎是目前速度最快的 JavaScript 引擎，JavaScript 程序在 V8 引擎下的运行速度媲美二进制程序，因此 Node.js 程序的即使编译执行速度与本地代码的执行速度相当。

Node.js使用了一个事件驱动、非阻塞式I/O的模型，使其轻量又高效。

Node.js 的设计思想是以事件驱动为核心，提供的巨大多数 API 都是基于事件的、异步的风格。采用事件驱动、异步编程，为网络服务而设计。优势在于，充分利用了系统资源，执行代码不会因阻塞而等待，有限的资源可以用于其他的任务。这样的设计非常适合于后端的网络服务编程，Node.js 的目标也在于此。

服务器的并发连接最大数量往往是 Web 应用程序架构的瓶颈，传统的后台语言采用的是阻塞式函数，阻塞式函数会导致资源浪费和时间延迟。Node.js 非阻塞模式的I\O处理，使其拥有在相对低系统资源耗用下的高性能与出众的负载能力。Node.js轻量高效，可以认为是数据密集型分布式部署环境下的实时应用系统的完美解决方案。

Node.js使用了包管理器npm来对模块进行管理，包管理器npm是全球最大的开源库生态系统。【Node.js 内建了 HTTP，Socket，TLS，FTP，DNS等网络应用模块，能快速开发出各种网络服务和网络服务器的组合，而不需要额外的服务器（如Apache、Tomcat、Ngnix等）支持。】。开发应用时，我们可以根据应用的需求【Node.js文档http://nodejs.cn/】

2.8 MongoDB

[MongoDB](http://baike.baidu.com/link?url=gWFYOJ6GwclQFR70ZjCpeKoilmoVHkhLlmDUKBQuCUYLmEudt7WeP7SuWB44XlDnIk1amyVURwcuTKWcofb6Yq)是一种分布式文档存储型数据库，是介于关系数据库和非关系数据库之间的产品；在非关系数据库当中功能最丰富，最像关系数据库。它存储的是类似 JSON 的BSON格式（JSON 的二进制），因此可以存储比较复杂的数据类型。MongoDB的特点是高性能、易部署、易使用，存储数据非常方便。而最大的特点是它支持的查询语言非常强大，其语法类似于面向对象的查询语言，可以实现类似关系数据库单表查询的绝大部分功能，还支持对数据建立索引。

第三章 系统需求分析

3.1 项目整体概述

3.2 系统功能性需求分析

3.2.1 总体功能性分析

3.2.2 查看教研信息功能分析

3.2.3 编辑发布教研信息功能分析

3.2.4 教研活动信息统计功能分析

1.3.2需求分析

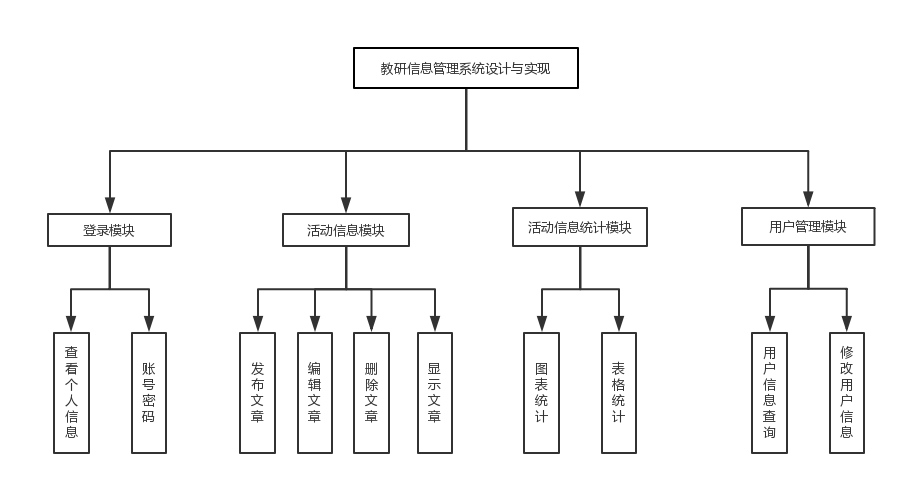
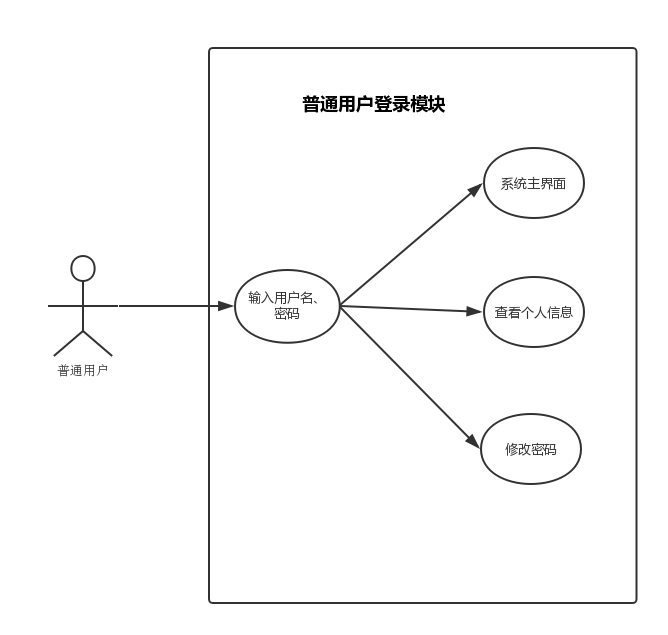


图2.1系统架构图

Fig 2.1 System Function Diagram

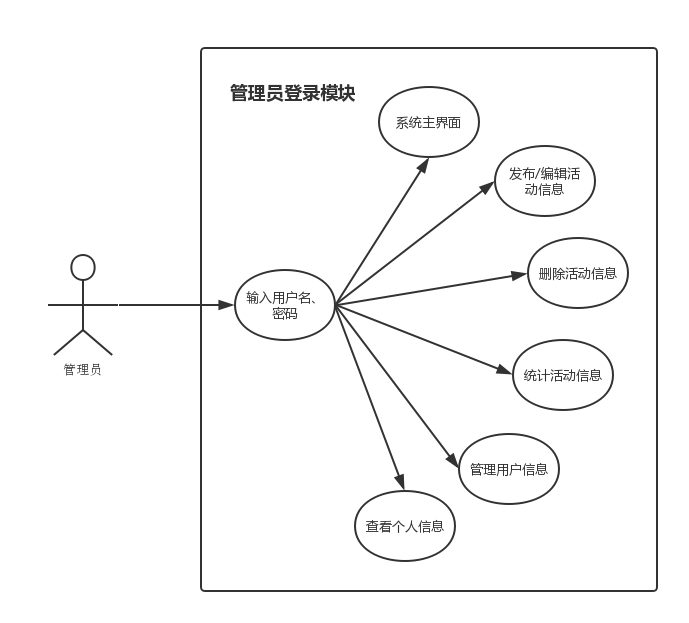
1.3.2.1 登录界面

本系统共分为普通用户、普通管理员、院级管理员、系统管理员四种角色。普通用户可以浏览【以及评论】管理员发布的信息，登录后可查看自己的参与活动信息；普通管理员可以发布、修改、统计自己所属教研室的信息【删除评论】；院级管理员可以发布、修改、统计自己所属学院的信息，对所属学院普通管理员进行管理；系统管理员拥有最高权限，可以发布、修改、统计信息，对所有管理员进行管理。



普通用户登录用例图

Domestic Consumer Login Model User Case Diagram



管理员登录用例图

Administrator Login Model User Case Diagram

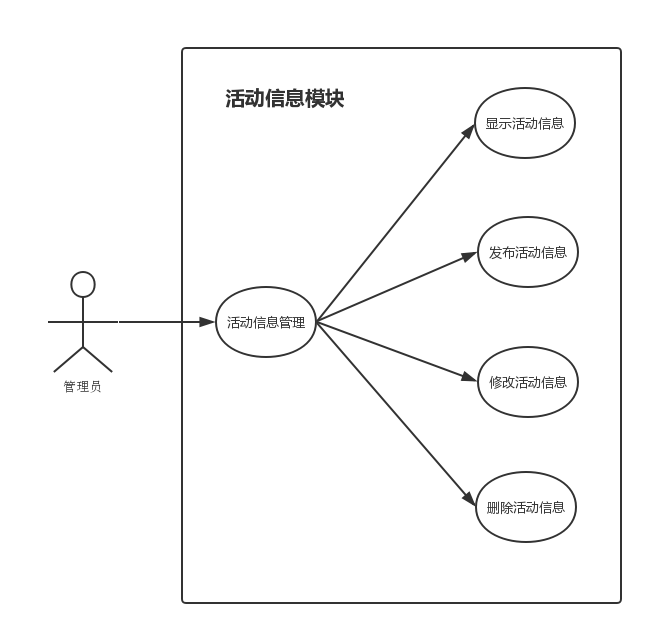
用户在未登录状态下，可浏览已发布的所有信息以及详情。登录后通过等级判断权限，防止非法用户对本系统的恶意、无意修改，保障系统安全。

用户登录后，通过用户权限判断，导航栏显示不同的选项。普通用户有首页、登出、个人中心三个选项；管理员除了普通用户的三个选项外，还有发布和用户管理。其中，用户管理页面显示的内容会根据管理员的等级有所不同。

1.3.2.2 活动信息管理

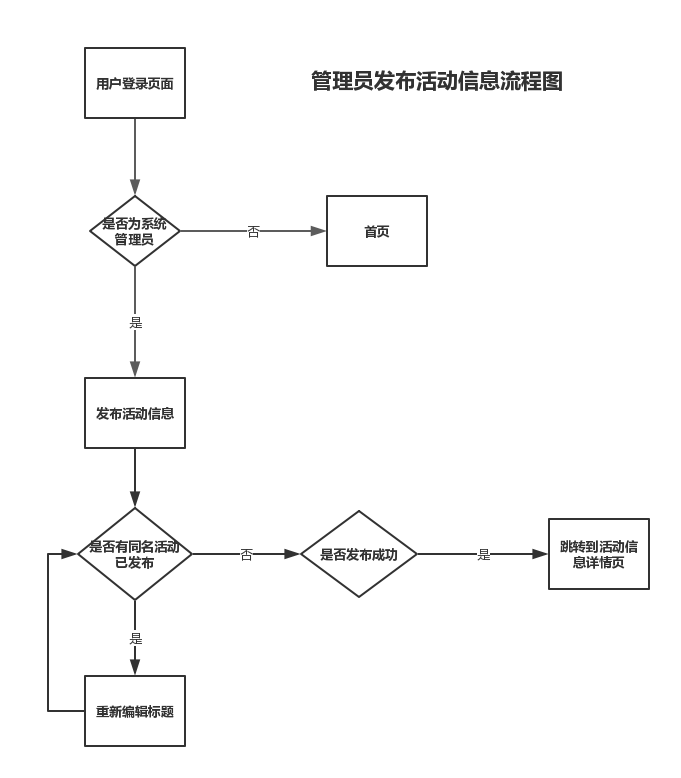
活动信息是本系统的主要内容之一，需要为管理员提供增删改查功能（CRUD）

1. 发布新活动信息：新建活动信息，并存储于数据库中；
2. 编辑活动信息：对已有的活动信息进行修改，如上传活动照片、添加活动参与者等；
3. 显示活动信息：首页显示所有活动的标题、举办时间，内容页面显示活动信息的详情；
4. 删除活动信息：删除已有的活动信息；



活动信息管理模块用例图

Action Message Management Model Use Case Diagram



管理员发布信息流程图

Flow-Process Diagram of Administrator Issue Information

【用户均可在所有活动详情中发表评论及删除自己的评论，管理员可对用户不当评论进行删除】

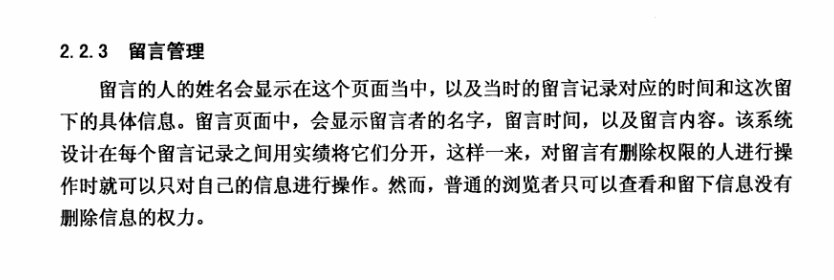
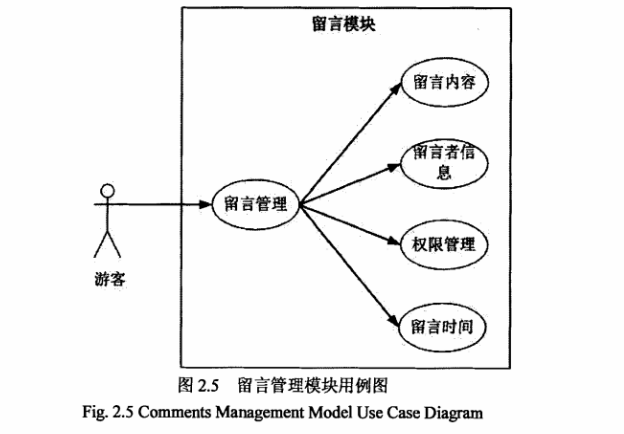
1.3.2.3 活动信息统计管理

本系统的另一个主要内容是管理员对所属教研室（学院、校）的已举办以及将举办的活动进行统计，包括不同类型的活动按年份、月份所举办的次数统计。统计以图表的形式展示【可选择表格展示】，可下载为图片保存于本地。

1.3.2.4 用户管理

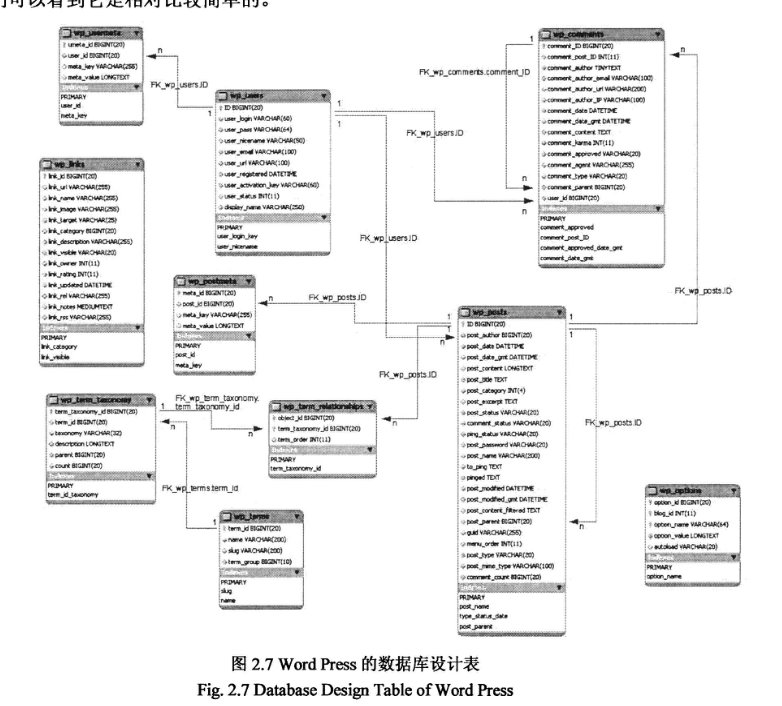
高一级用户对低一级用户的增删改查（CRUD）。权限由高到低为：系统管理员，【院级管理员，】普通管理员，用户。

【1.3.2.3 留言管理】

1.3.3 数据分析

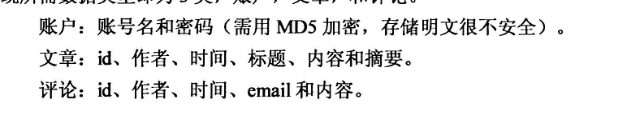
本系统的数据有活动信息、【评论】、用户信息。ER图如下：

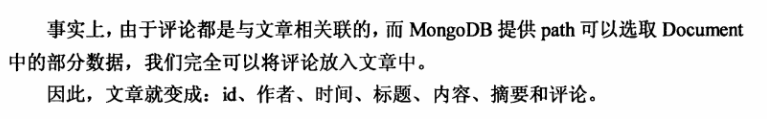


活动信息：id

【评论：】

用户：





3.3 系统非功能性需求分析

3.3.1 可靠性

3.3.2 易用性

3.3.3 安全性

3.3.4 兼容性

第四章 系统设计 //可用图、表介绍设计结果，每个图或表要有相应文字描述

4.1 总体设计 //用层次图表示所设计的各功能模块及模块间调用关系

4.1.1 系统设计目标

4.1.2 系统API设计

4.1.2 系统模块划分

4.1.3 系统整体架构设计

教研管理统计系统主要实现用户浏览信息，管理员对教研信息的编辑发布、教研活动的统计，以及对的用户管理。

系统的功能需求分为用户中心和后台管理两大模块。

用户中心包含教研活动信息及详情浏览、用户信息管理；

后台管理是系统的核心，包含信息编辑修改功能、教研活动信息及用户参与情况的数据可视化统计、用户账号统计管理。

由于该系统主要功能为信息展示，且采用了前后端分离的架构，前端负责 View 层和 Controller 层，后端只负责 Model 层，业务处理等；因此决定做成单页面应用（SPA），通过前端路由控制，实现数据、页面局部更新，使网页加载更快，浏览器性能消耗更低，后台实现对数据库的更改、查询，并提供接口。前后端分离，职责分明，开发、维护成本降低。

4.2 数据库设计 //表示出所设计的各个表

// TODO

至今，数据库技术产生了 SQL 数据库和 NoSQL（非关系数据库的统称）数据库两大类型。MongoDB 是一个基于分布式文件存储的 NoSQL 数据库，存储数据方便、性能高。

4.3 详细设计

//用传统方法设计时，详细设计主要用程序流程图表示主要模块的算法，如果是用面向对象分析方法时，详细设计的内容主要用序列图等UML图来表示出各对象间的交互关系。

单页面（SPA）应用具有一次加载，局部刷新数据的特点，是网页加载更快，适用于展示类网站；而本系统主要功能为信息展示，实时响应要求低，因此选择使用单页面进行开发。同时，实现前后端分离，前端负责 View 层和 Controller 层，通过路由实现数据、页面的更新；后端负责 Model 层，以及对数据的操作等；

…

第五章 系统实现 //用图、文展示系统主要功能的运行情况，即各功能输入输出情况

5.1 查询信息的实现

5.1.1 查询全部教研活动信息的实现

5.1.2 查询具体教研活动详细信息的实现

5.2 编辑发布信息的实现

5.2.1 编辑内容的实现

5.2.2 修改信息的实现

5.3 查询统计信息的实现

5.3.1 以组织为单位查询的实现

5.3.2 以个人为单位查询的实现

//TODO

主界面由标题、时间选择、地点、概要，以及主要容编辑区域组成。

第六章 结论与展望

6.1 总结 //描写开发得到的主要功能，系统实现的效果等。

6.2 展望 //描写本研究的不足之处，以及以后还需要进一步研究的内容