校园文章分享系统
设计说明书

项目组成员：

陈果（12331002）

邓小康（12331005）

高钰（12331006）

孟泽宇（12331015）

陈家志（12331003）

黄建（12331008）

目录

引言

本项目来源于本小组的系统分析与设计课程项目。

1 系统框架

本系统基于 MVC 框架开发。系统分为三层，分别为：表示层、业务层以及数据层。 该结构具有清晰的依赖关系，表示层依赖于业务层，业务层调用数据层。本文使用包的形式描述系统各层之间的依赖关系。

- 表示层

表示层是用户与系统交互的界面，负责获取用户的请求和信息， 展示系统的操作结果给用户。表示层的模块包括：用户查看文章、用户检索文章、管理员发布文章、管理员管理文章。

- 业务层

业务层是系统业务逻辑的核心。它负责接收用户的请求和信息，调用数据层的数据，执行系统的业务逻辑操作，并最终将操作结果返回给用户。本系统业务层的模块包括：用户查看文章、用户检索文章、管理员发布文章、管理员管理文章。

- 数据层

数据层负责系统实体对象的数据访问，本系统数据层的模块包括：用户、管理员、文章。

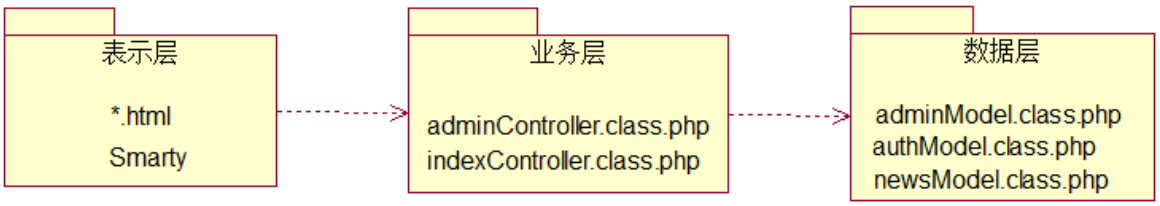

图 1 系统架构图

2 关键抽象

系统关键抽象即系统实体类图，系统实体类图描述了系统中的类及其相互之间的各种关系。它反映了系统中包含的各种对象的类型以及对象间的各种静态关系，主要描述了系统实体层中各实体类的属性及其相互的关系，是对实体层中各

模块的描述。

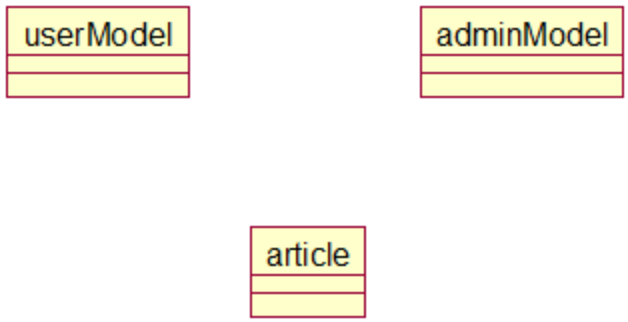

图 2 系统关键抽象

3 用例分析

3.1 管理员登录用例分析

3.1.1 管理员登录用例功能描述

管理员可以利用这一功能登入系统管理界面，对文章进行管理。

3.1.2 管理员登录用例交互过程

⑴管理员权限用户访问系统登录界面，输入管理员用户名和密码，单击提交。

⑵登录界面逻辑层取到用户名和密码信息后，调用检查密码的业务逻辑，与后台数据库交互，进行用户名和密码的校验。

⑶如果校验正确，表示层将管理员管理页面呈现给用户，否则，提示登录失败信息。

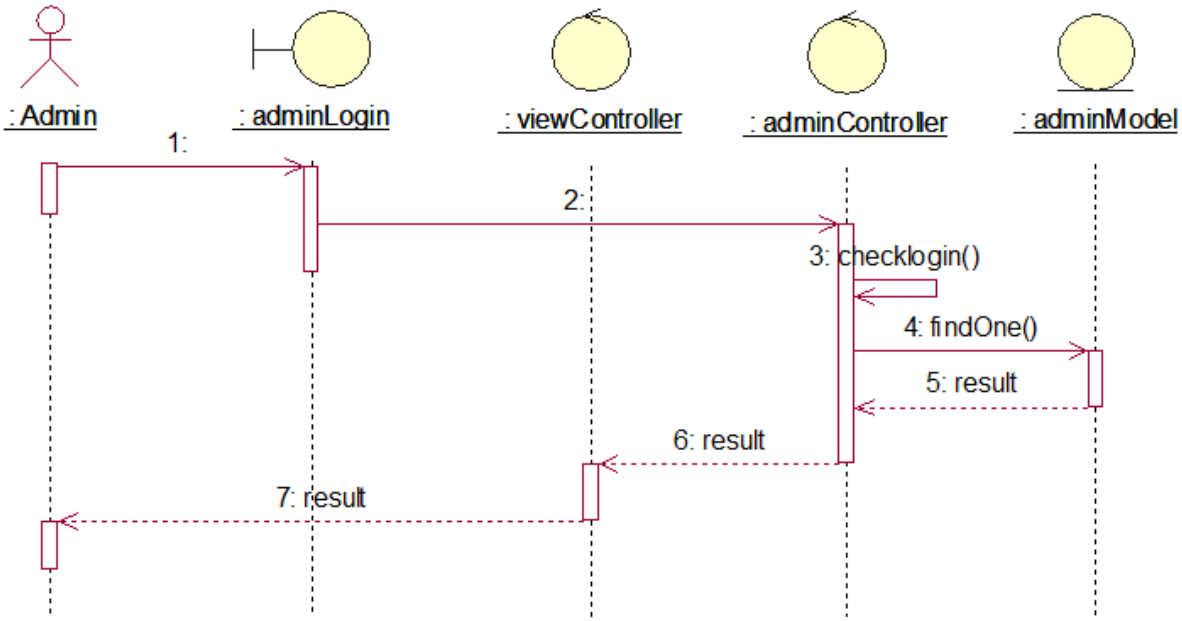

图 3 管理员登录用例时序图

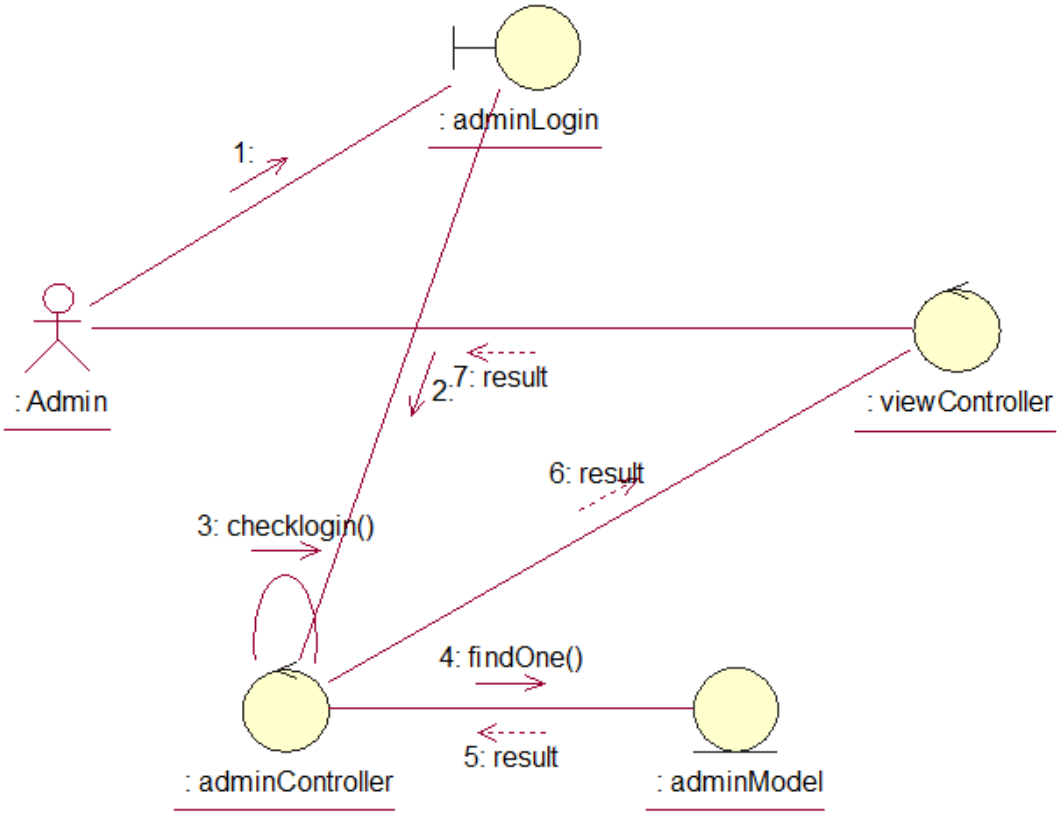

图 4 管理员登录用例协作图

3.1.3 管理员登录用例的类分析和设计

边界类：用例中，边界类为管理员登录界面（adminlogin.html）。该页面以

表单为主，用于获取用户名和密码。

业务逻辑类：管理员控制类（adminController.php）和视图类（viewController.php)。管理员控制类调用 checklogin()检查用户输入的用户名和密码是否在数据库内，且是否匹配，若合法，则调用视图类（viewController.php）将管理界面（manager.html）呈现给管理员。

实体类：管理员实体类（adminModel.php)。该类实现与数据库的交互，将登录页面获取的用户名和密码通过 checklogin()函数传递给管理员实体类，与数据库交互，查询用户名和密码是否合法并且匹配。

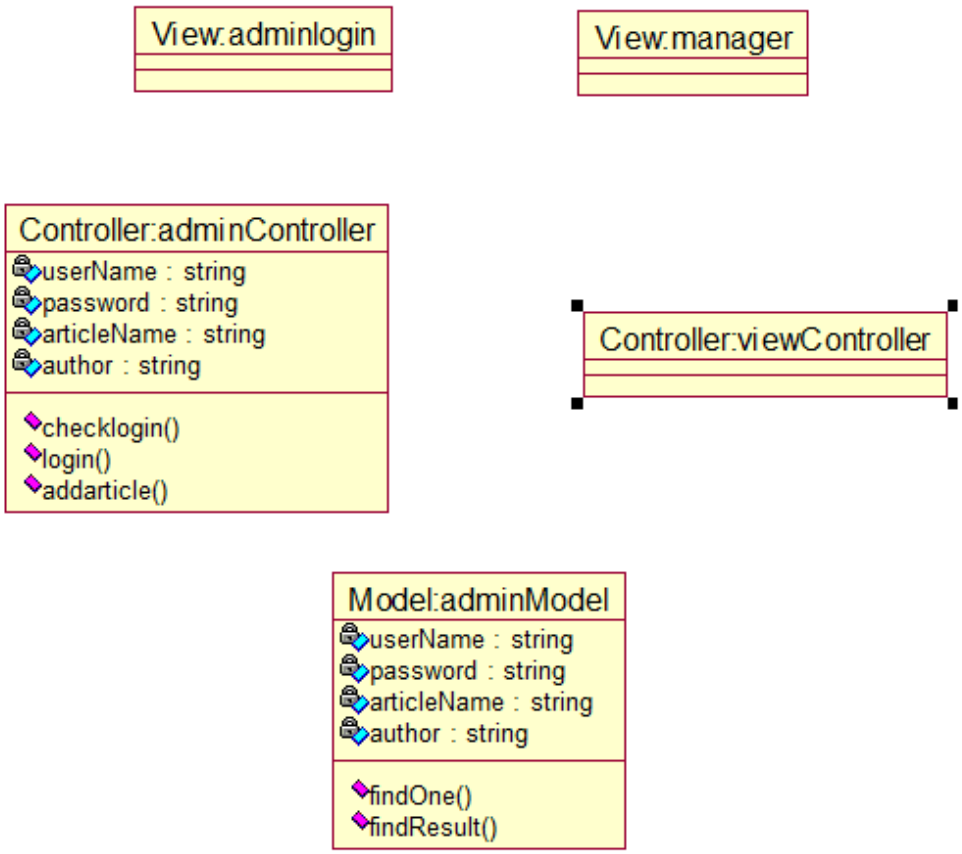

图 5 管理员登录用例的分析类

3.1.4 管理员登录用例分析类关联关系

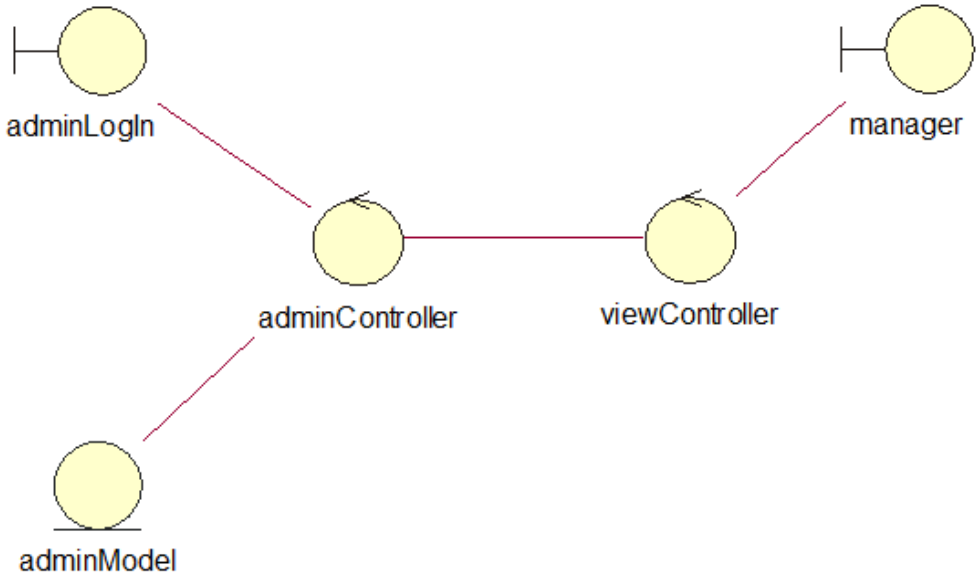

图 6 管理员登录用例分析类关联关系

3.2 管理员发布文章用例分析

3.2.1 管理员发布文章用例功能描述

管理员可以利用这一功能为网站添加文章。

3.2.2 管理员发布文章用例交互过程

⑴管理员在登录的前提下在文章管理页面单击“添加文章”，进入文章信息填写页面。

⑵管理员将文章信息填写完毕，单击提交，将文章信息传递给业务逻辑层。

⑶业务逻辑层检查文章的题目是否为空。如果为空，系统提示用户文章标题不能为空；如果不为空，业务逻辑层与数据层交互，将文章信息添加到数据库中，持久化文章对象。

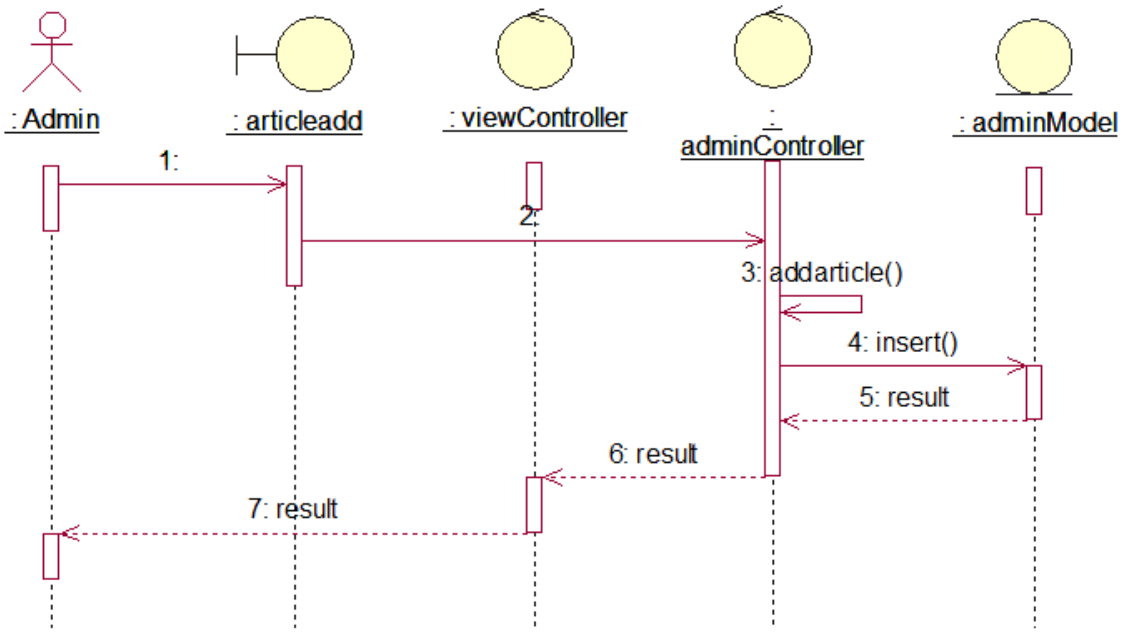

图 7 管理员发布文章用例时序图

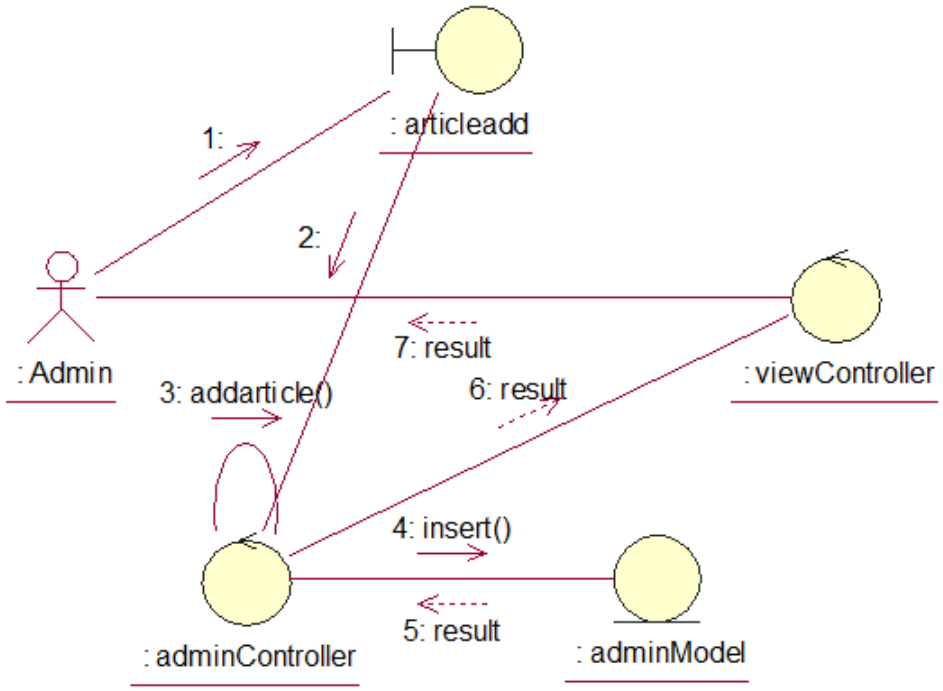

图 8 管理员发布文章用例协作图

3.2.3 管理员发布文章用例的类分析和设计

边界类：用例中，边界类为文章信息填写页面（articleadd.html）和文

章管理页面(manager.html)。文章管理页面左侧中部有添加文章选项，管理员在登录状态下单击添加文章，系统将返回一个有一系列添加框图的文章信息填写页面（articleadd.html）给管理员。

业务逻辑类：管理员控制类（adminController.php）和视图类（viewController.php)。管理员控制类调用 AddArticle()函数处理从文章信息填写页面传递的文章信息并将其传递给实体类（adminModel.php)。

实体类：管理员实体类（adminModel.php)。该类实现与数据库的交互，将经过业务逻辑层检查过的文章信息添加入数据库中，并通过视图类（viewController.php）将文章显示给用户。

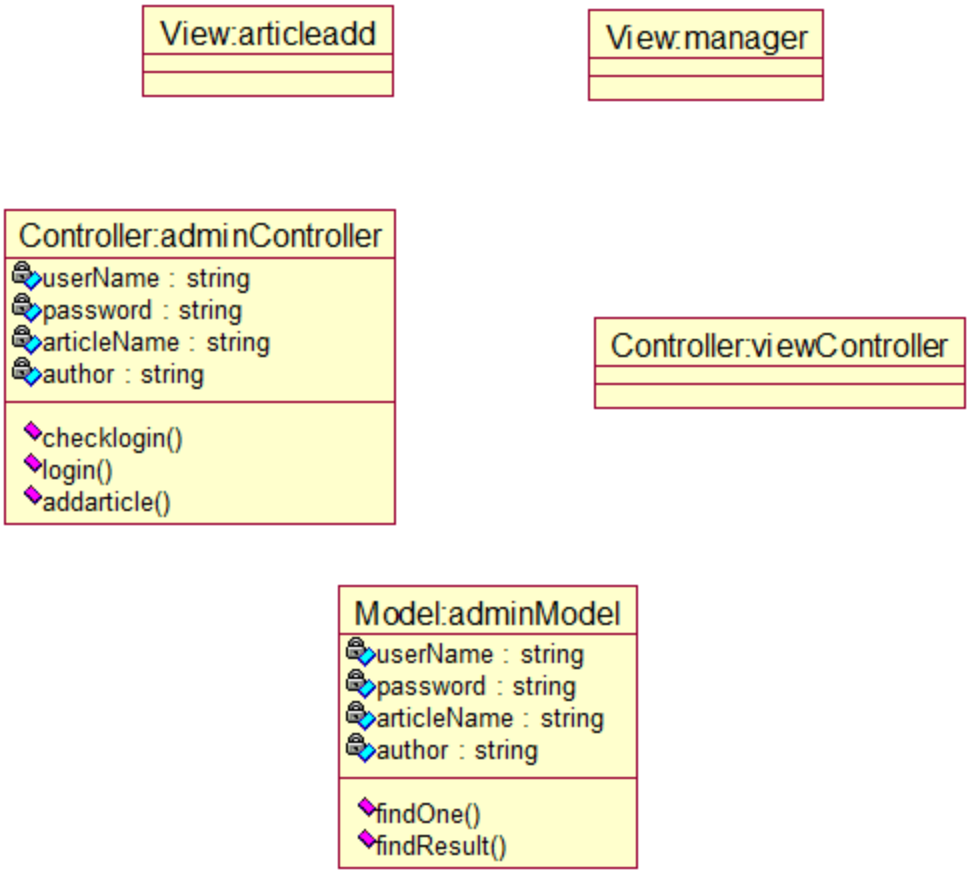

图 9 管理员发布文章用例的分析类

3.2.4 管理员发布文章用例分析类关联关系

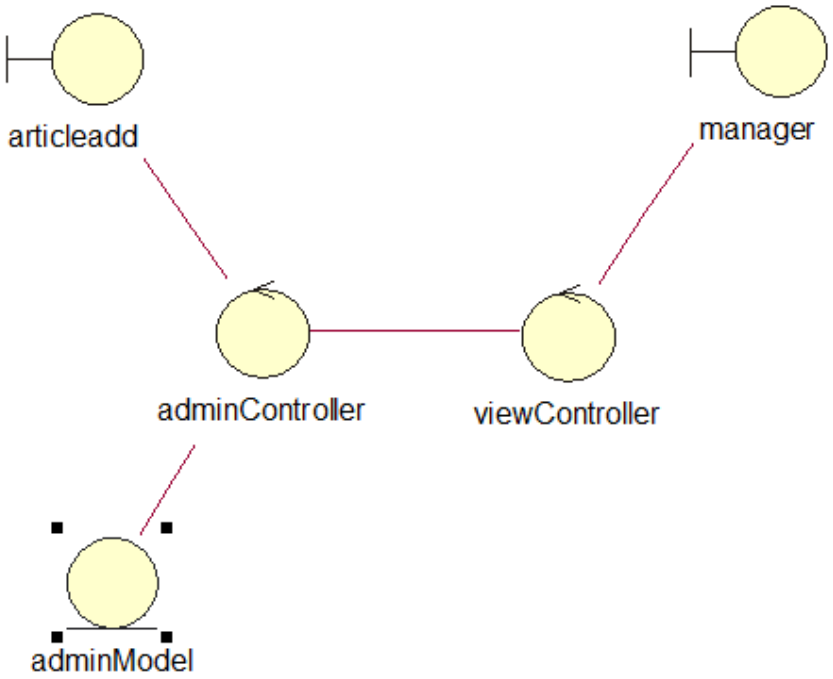

图 10 管理员发布文章用例分析类关联关系

3.3 用户查看文章用例分析

3.3.1 用户查看文章用例功能描述

用户可以利用这一功能查看文章。

3.3.2 用户查看文章用例交互过程

⑴用户在登录的前提下，在文章列表页面中点击所要查看文章题目旁边的“查看文章”的按钮，将文章题目传递给业务逻辑层。

⑵业务逻辑层与数据层交互，在数据库中查找相应的文章，然后将文章内容通过视图类显示给用户。

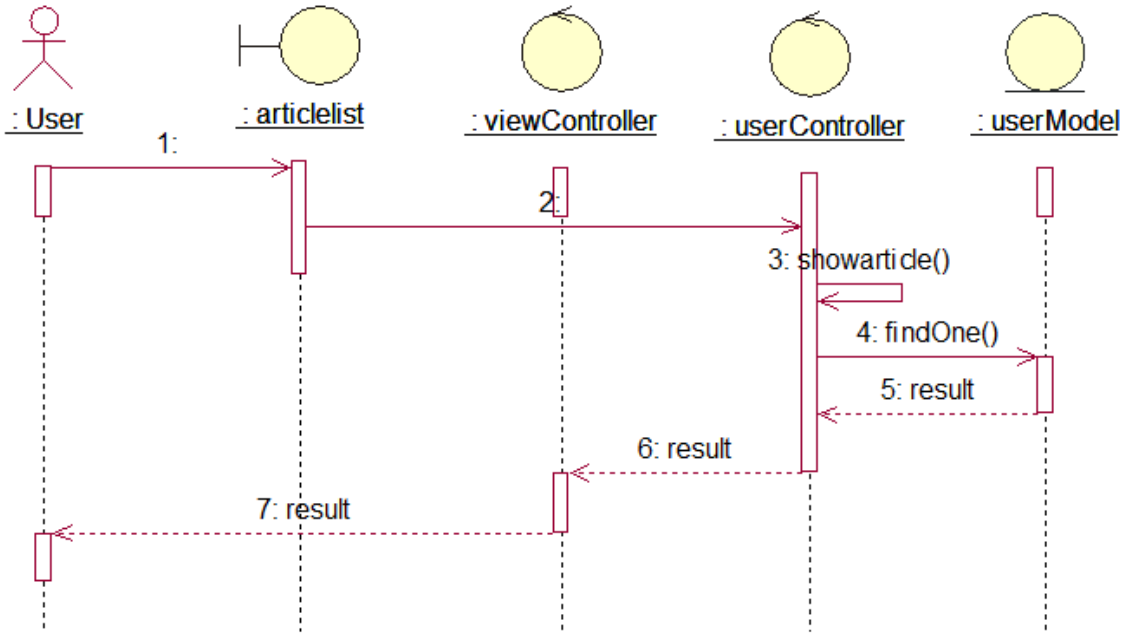

图 11 用户查看文章用例时序图

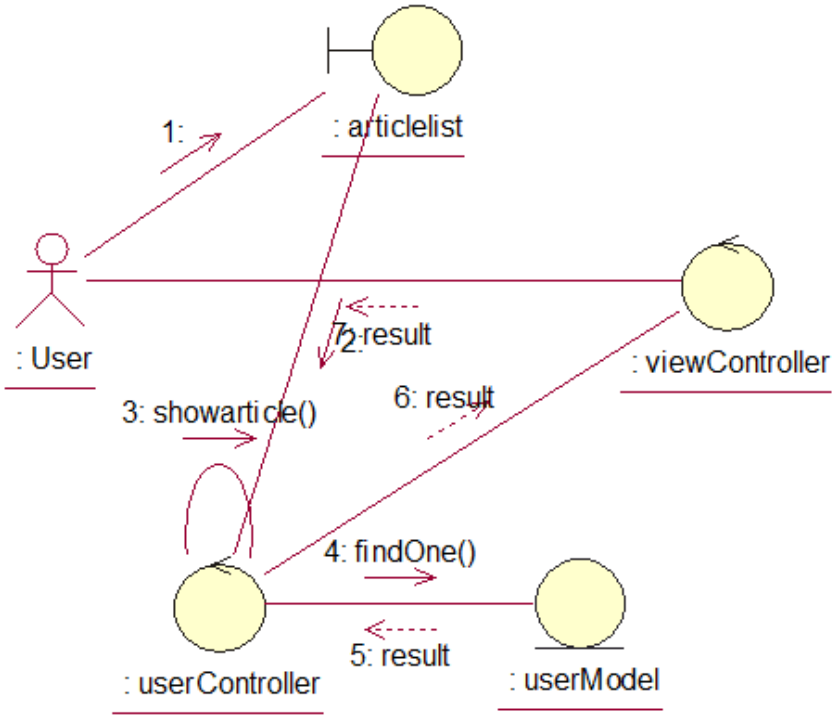

图 12 用户查看文章用例协作图

3.3.3 用户查看文章用例的类分析和设计

边界类：文章列表页面（articlelist.html）和文章详情页面

（articledetail.html）。文章列表页面以列表的形式展现出所有文章的题目、作者和简介信息，文章题目旁边还有一个“查看文章”的按钮。文章详情页面显示所要查看的文章的题目、作者、内容。

业务逻辑类：用户控制类（userController.php）和视图类（viewController.php）。用户控制类调用 showarticle()函数从数据库中取出相应文章内容，然后通过视图类（viewController.php）显示给用户。

实体类：用户实体类（userModel.php）。该类实现与数据库的交互，将文章内容传递给业务逻辑层。

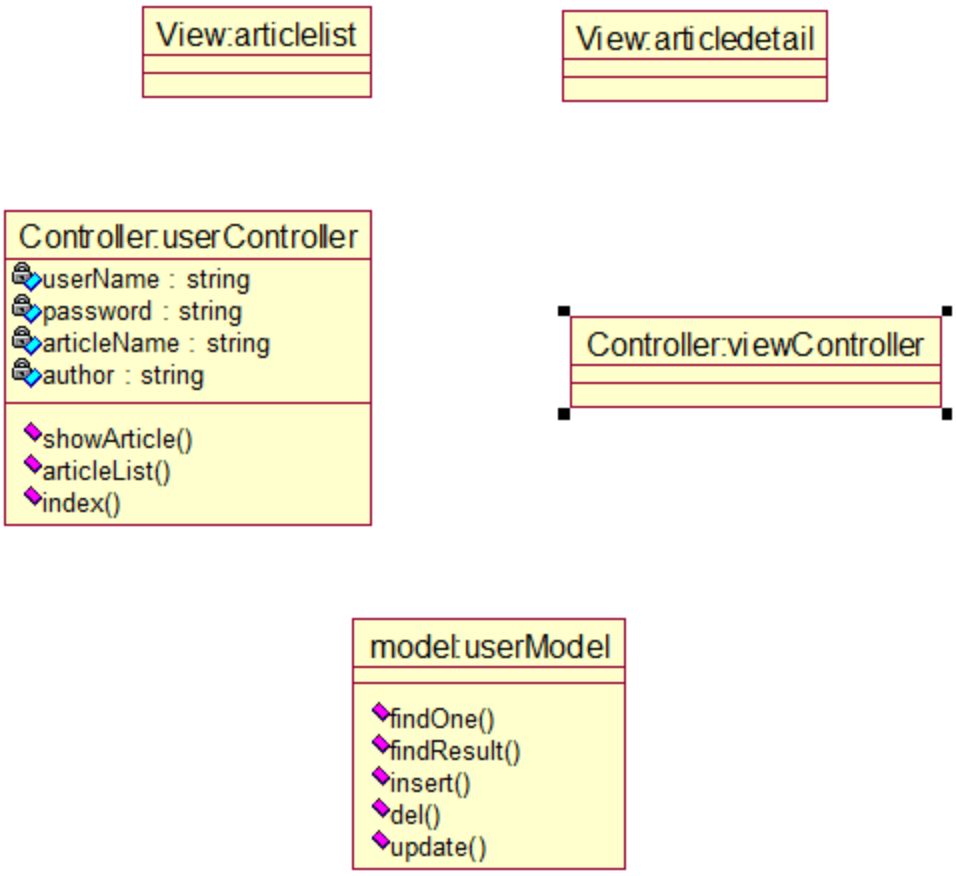

图 13 用户查看文章用例的分析类

3.3.4 用户查看文章用例分析类关联关系

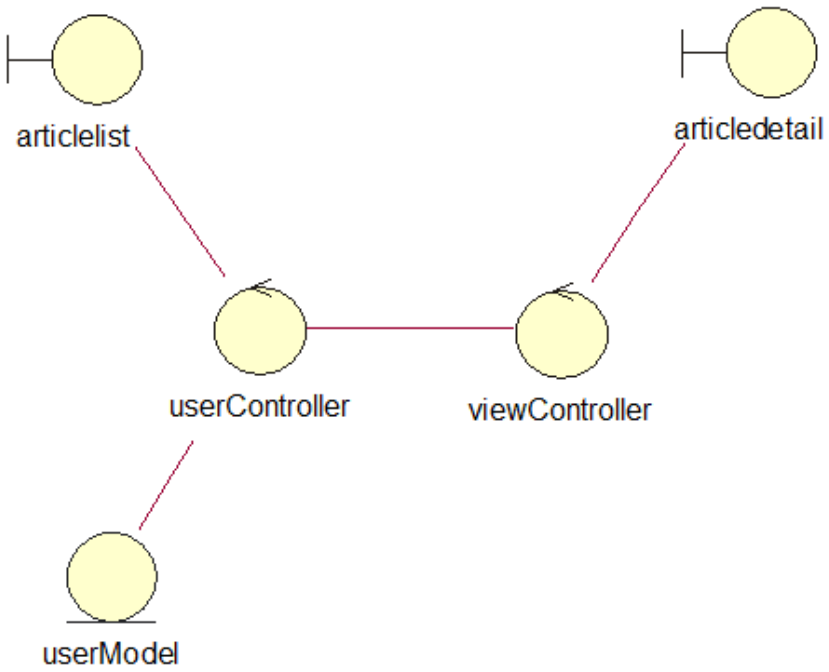

图 14 用户查看文章用例分析类关联关系

4 分析类到分析机制映射

表 1 分析类到分析机制映射图

分析类	分析机制
adminlogin	图形化
articleadd	图形化
manager	图形化
articledetail	图形化
articlelist	图形化
adminController	分布式、安全性
viewController	分布式、安全性
userController	分布式、安全性
adminModel	持久性、安全性
userModel	持久性、安全性

5 系统类图

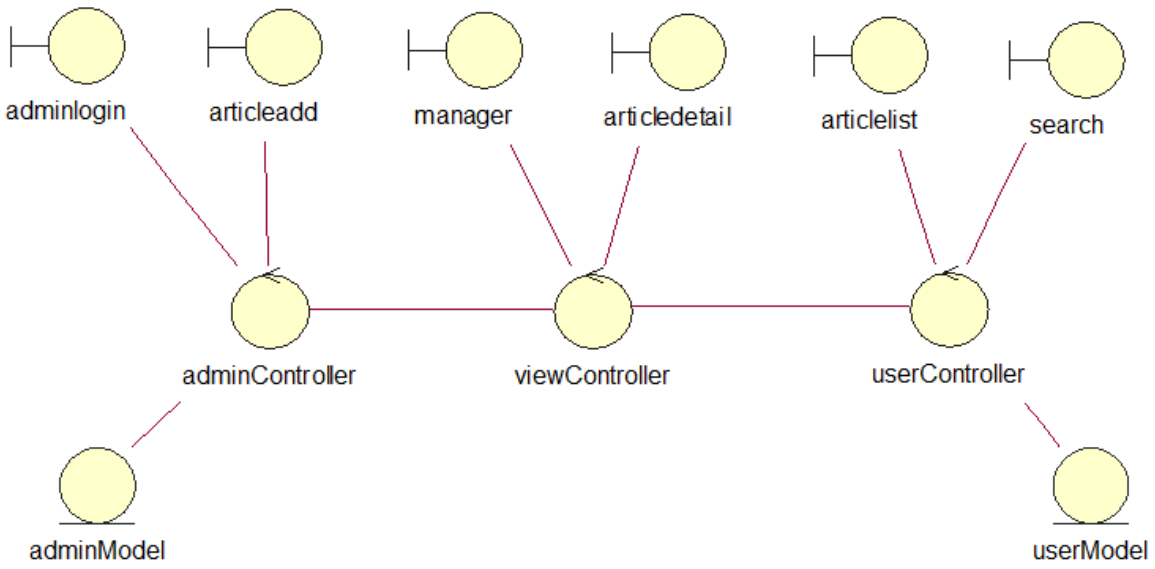

图 15 系统类图

6 子系统设计

6.1 子系统划分

本小节在系统框架、系统关键抽象和分析类的析取的基础上，将系统划分成 3 个逻辑上相对独立的子系统，分别为：管理文章子系统、查看文章子系统、检索文章子系统。每个子系统包含了表示层、业务逻辑层和数据层的类。比如：在管理文章子系统中，包括了表示层的发布文章界面、修改文章和删除文章界面。下图显示了三个子系统的相互依赖关系。

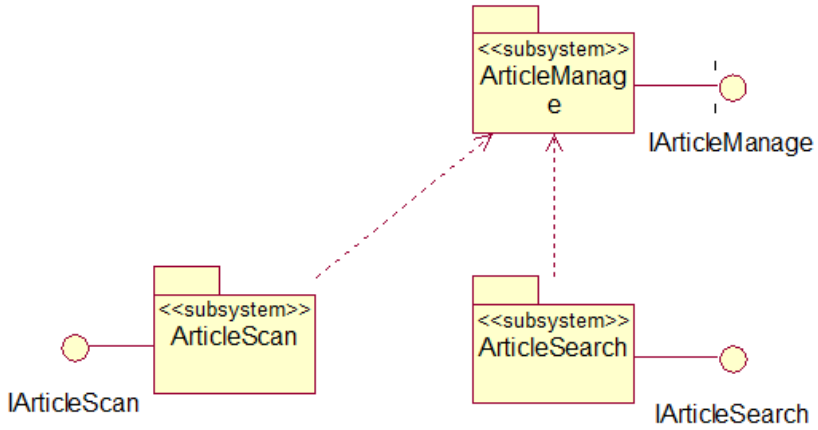

图 16 子系统图

6.2 子系统设计

6.2.1 管理文章子系统

管理文章子系统实现了文章实体的业务逻辑功能，如：发布新文章、修改文章信息、删除文章等功能。

依据系统框架以及文章实体的业务逻辑功能，可将管理文章子系统分成三个子模块，包括：文章操作表单、文章管理和文章实体，分别对应于系统框架的表示层、控制层和实体层。

⑴文章操作表单子模块

对应于系统框架的表示层。该模块主要包含了文章表示层的各种表单，如：发布文章表单，修改文章信息表单和删除文章表单等。

⑵文章管理子模块

对应于系统框架的控制层。该模块主要包含关于文章实体的各种业务逻辑类，如：发布文章、修改文章信息、删除文章等功能。

⑶文章实体子模块

对应于系统框架的实体层。该模块包含了文章实体类，实现数据的持久化。

6.2.2 检索文章子系统

检索文章子系统实现了文章检索的业务逻辑功能。

依据系统框架，可将检索文章子系统分成三个子模块，包括：文章检索表单、文章检索和文章实体，分别对应于系统框架的表示层、控制层和实体层。

⑴文章检索表单子模块

对应于系统框架的表示层。该模块主要包含了文章检索表单（通过一个检索框来操作）。

⑵文章检索子模块

对应于系统框架的控制层。该模块主要包含检索文章功能的各种业务逻辑类，如：按照关键字在数据库中查找文章、按时间排序显示给用户等功能。

⑶文章实体子模块

对应于系统框架的实体层。该模块包含了文章实体类，在数据库中被按照关键字进行匹配。

6.3 分析类到设计元素映射

系统的分析类被映射为 3 个子系统。

表 2 分析类到设计元素映射表

分析类	设计元素
发布文章功能	文章管理子系统
修改文章功能	文章管理子系统
删除文章功能	文章管理子系统
查看文章功能	查看文章子系统
检索文章功能	检索文章子系统

6.4 设计元素到包映射

由于系统的 3 个子系统逻辑上相对独立，故将每个子系统设置为一个包。包中包含了实现该子系统的各种类，包括表示类、控制类和实体类等。

表 3 设计元素到包映射表

设计元素	包
文章管理子系统	文章管理包
查看文章子系统	查看文章包
检索文章子系统	检索文章包

7 具体模块实现与各模块对应代码段

7.1 管理员登录模块

管理员填写管理员信息进行登陆，登陆后可以进入网站后台管理系统，对网站进行管理。

代码示例:

驱动程序代码示例：

```
class PC{
    public static $controller;
    public static $method;
    private static $config;
    private static function init_db(){
        DB::init('mysql', self::$config['dbconfig']);
    }
    private static function init_view(){
        VIEW::init('Smarty', self::$config['viewconfig']);
    }
    private static function init_controllor(){
        self::$controller = isset($_GET['controller'])?daddslashes($_GET['controller']):'
    }
    private static function init_method(){
        self::$method = isset($_GET['method'])?daddslashes($_GET['method']):'index';
    }
    public static function run($config){
        self::$config = $config;
        self::init_db();
        self::init_view();
        self::init_controllor();
        self::init_method();
        C(self::$controller, self::$method);
    }
```

图 17

管理员登录部分主要代码示例：

Controller 层：

```
    public function login(){
        if(!isset($_POST['submit'])){
            VIEW::display('admin/login.html');
        }else{
            $this->checklogin();
        }
    }
```

图 18

```
private function checklogin(){
    if(empty($_POST['username'])||empty($_POST['password'])){
        $this->showmessage('登录失败！', 'admin.php?controller=admin&method=login');
    }
    $username = daddslashes($_POST['username']);
    $password = daddslashes($_POST['password']);
    $authobj = M('auth');
    if($auth = $authobj->checkauth($username, $password)){
        $_SESSION['auth'] = $auth;
        $this->showmessage('登录成功！', 'admin.php?controller=admin&method=index');
    }else{
        $this->showmessage('登录失败！', 'admin.php?controller=admin&method=login');
    }
}
```

图 19

Model 层：

```
function checkauth($username, $password){
    $adminobj = M('admin');
    $auth = $adminobj -> findOne_by_username($username);
    if((!empty($auth))&&$auth['password']==$password){
        return $auth;
    }else{
        return false;
    }
}
```

图 20

View 层：视图层使用 smarty 视图引擎，将 model 层提取出的数据库信息通过 controller 层处理后显示出来。下同。

7.2 管理员添加文章模块

管理员进入文章管理页面，点击添加文章选项，可以在网站添加文章，将信息按要求填写完全之后，成功添加。

代码示例:

```
public function newsadd(){
    if(!isset($_POST['submit'])){
        $data = $this->getnewsinfo();
        VIEW::assign(array('data'=>$data));
        VIEW::display('admin/newsadd.html');
    }else{
        $this->newssubmit();
```

图 21

```
private function newssubmit(){
    extract($_POST);
    if(empty($title)||empty($content)){
        $this->showmessage('请把文章标题、内容填写完整再提交！', 'admin.php?controller=admin&method=newsadd');
    }
    $title = daddslashes($title);
    $content = daddslashes($content);
    $author = daddslashes($author);
    $description = daddslashes($description);
    $newsobj = M('news');
    $data = array(
        'title'=>$title,
        'content'=>$content,
        'author'=>$author,
        'description'=>$description,
        'dateline'=>time()
    );
    if($_POST['id']!=''){
        $newsobj ->update($data, intval($_POST['id']));
        $this->showmessage('修改成功！', 'admin.php?controller=admin&method=newslist');
    }else{
        $newsobj ->insert($data);
        $this->showmessage('添加成功！', 'admin.php?controller=admin&method=newslist');
    }
}
```

图 22

7.3 管理员修改文章模块

在已经添加入文章的基础上，管理员进入文章管理界面后可以选择修改文章操作。

代码示例:

Controller 层：

```
private function getnewsinfo(){
    if(isset($_GET['id'])){
        $id = intval($_GET['id']);
        $newsobj = M('news');
        return $newsobj->findOne_by_id($id);
    }else{
        return array();
    }
}
```

图 23

```
private function newssubmit(){
    extract($_POST);
    if(empty($title)||empty($content)){
        $this->showmessage('请把文章标题、内容填写完整再提交！', 'admin.php?controller=admin&method=newsadd');
    }
    $title = daddslashes($title);
    $content = daddslashes($content);
    $author = daddslashes($author);
    $description = daddslashes($description);
    $newsobj = M('news');
    $data = array(
        'title'=>$title,
        'content'=>$content,
        'author'=>$author,
        'description'=>$description,
        'dateline'=>time()
    );
    if($_POST['id']!=''){
        $newsobj ->update($data, intval($_POST['id']));
        $this->showmessage('修改成功！', 'admin.php?controller=admin&method=newslist');
    }else{
        $newsobj ->insert($data);
        $this->showmessage('添加成功！', 'admin.php?controller=admin&method=newslist');
    }
```

图 24

Model 层：

```
function findOne_by_id($id){
    $sql = 'select * from '.$this->_table.' where id='.$id;
    return DB::findOne($sql);
}
```

图 25

```
    if(!empty($id)){
        $id = intval($id);
        $sql = 'select * from '.$this->_table.' where id='.$id;
        return DB::findOne($sql);
    }else{
        return array();
    }
```

图 26

7.4 管理员删除文章模块

在管理员已经登录并且有文章存在的情况下点击需要删除的文章的右侧删除按钮删除文章。

代码示例:

controller 层：

```
public function newsdel(){
    if($_GET['id']){
        $this->delnews();
        $this->showmessage('删除文章成功！', 'admin.php?controller=admin&method=newslist');
    }
}
```

图 27

```
private function delnews(){
    $newsobj = M('news');
    return $newsobj->del_by_id($_GET['id']);
}
```

图 28

model 层：

```
function del_by_id($id){
    return DB::del($this->_table, 'id='.$id);
}
```

图 29

7.5 用户浏览文章详情模块

用户看到感兴趣的文章标题和摘要点击文章详情，可以查看文章的详情。

代码示例:

controller 层：

```
function newsshow(){
    $data = M('news')->getnewsinfo($_GET['id']);
    VIEW::assign(array('data'=>$data));
    VIEW::display('index/newsshow.html');
}
```

图 30

model 层：

```
function getnewsinfo($id){
    if(!empty($id)){
        $id = intval($id);
        $sql = 'select * from '.$this->_table.' where id='.$id;
        return DB::findOne($sql);
    }else{
        return array();
    }
}
```

图 31

7.6 用户搜索文章模块

用户在搜索框中填入文章标题，单击 search，可以搜索到对应的文章

代码示例:

controller 层：

```
private function search(){
    $key = $_GET['key'];
    $sql = "select * from articles where title like '$key%' order by id";
    $query = mysql_query($sql);
    if($query&&mysql_num_rows($query)){
        while($row = mysql_fetch_assoc($query)){
        $data[] = $row;
    }
    VIEW::assign(array('data'=>$data));
    VIEW::display('index/show.html');
}
```

图 32

model 层：

```
function getnewsinfo($id){
    if(!empty($id)){
        $id = intval($id);
        $sql = 'select * from '.$this->_table.' where id='.$id;
        return DB::findOne($sql);
    }else{
        return array();
    }
```

图 33

8 系统运行时架构设计

本部分主要以管理员登录、管理员发布文章、用户查看文章 3 个用例为例来分别说明它们在运行时的架构设计。

8.1 管理员登录用例运行时架构设计

AdminApplication 进程对应于 AdminLogin 边界类的实现，

AdminLoginProcess 进程对应于 AdminController 控制类的实现，AdminDatabaseAccess 进程对应于 AdminModel 实体类的实现。每个进程跟对应的实现类都是一对一的关系。

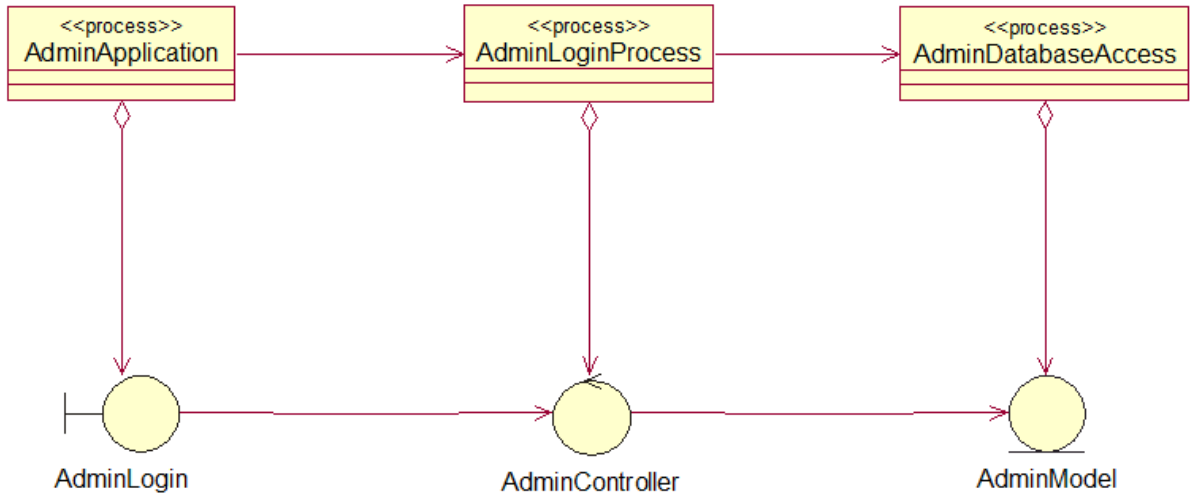

图 34 管理员登录用例运行时架构

8.2 管理员发布文章用例运行时架构设计

AdminApplication 进程对应于 articleadd 边界类的实现，ArticleAddProcess 进程对应于 AdminController 控制类的实现，ArticleDatabaseAccess 进程对应于 AdminModel 实体类的实现。每个进程跟对应的实现类都是一对一的关系。

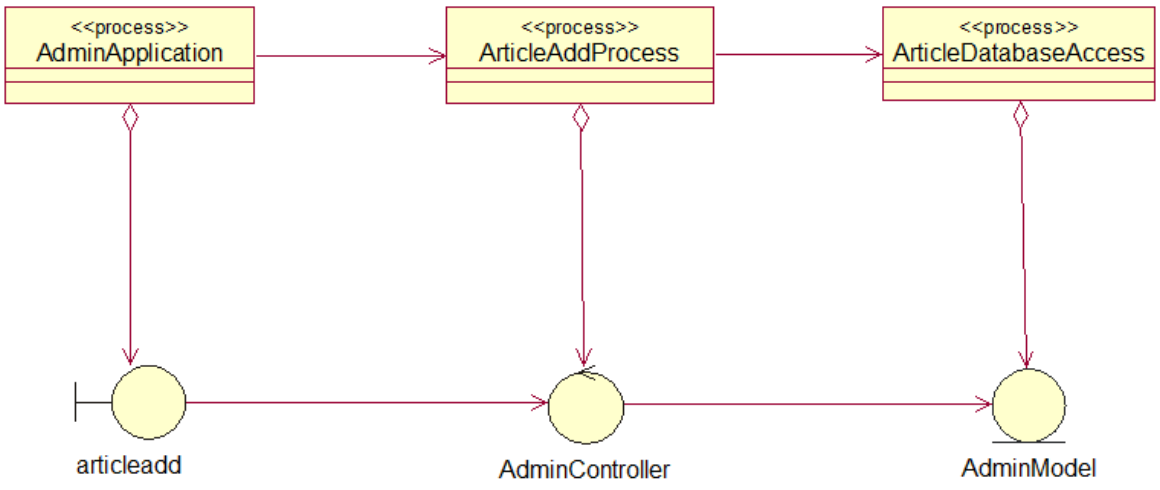

图 35 管理员发布文章用例运行时架构

8.3 用户查看文章用例运行时架构设计

UserApplication 进程对应于 articlelist 边界类的实现，ArticleReadProcess 进程对应于 UserController 控制类的实现，ArticleDatabaseAccess 进程对应于 UserModel 实体类的实现。每个进程跟对应的实现类都是一对一的关系。

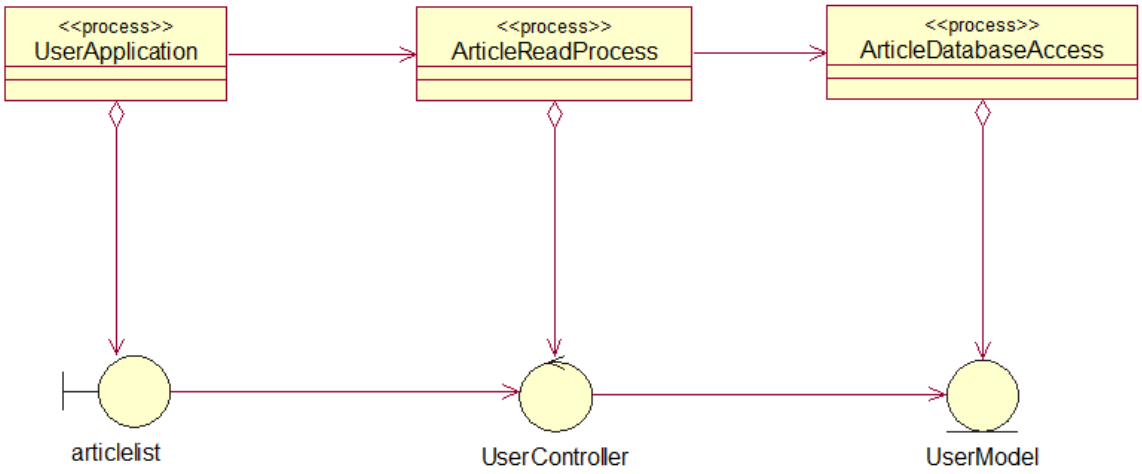

图 36 用户查看文章用例运行时架构

9 系统部署图

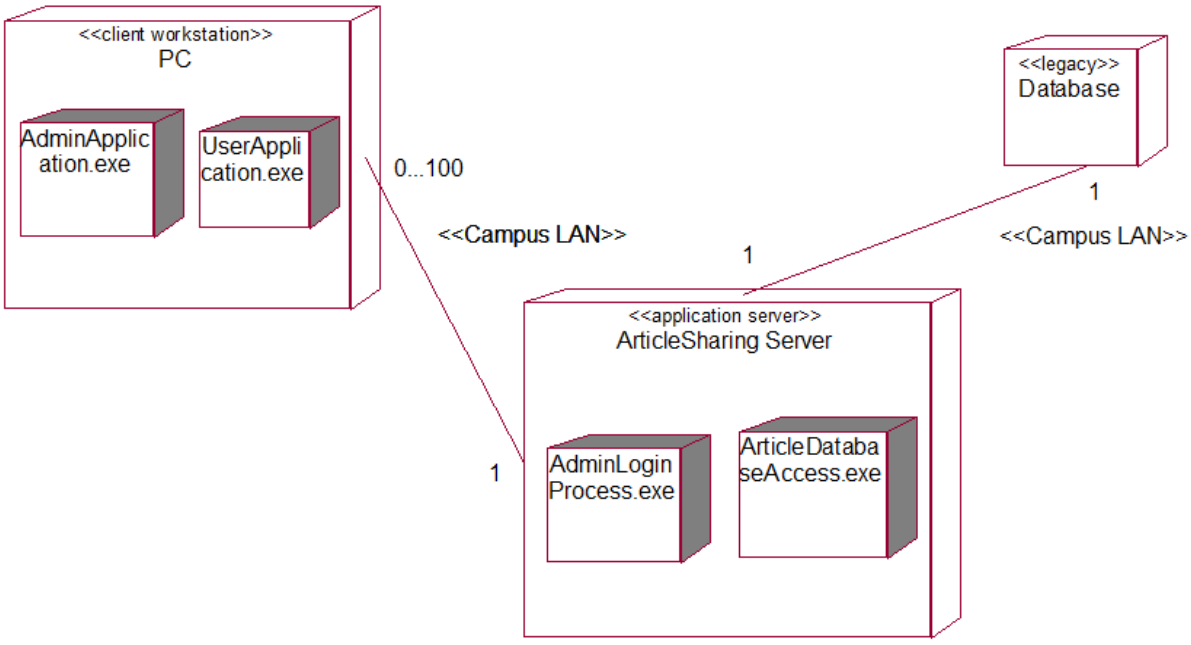

图 37 系统部署图

10 ER 图

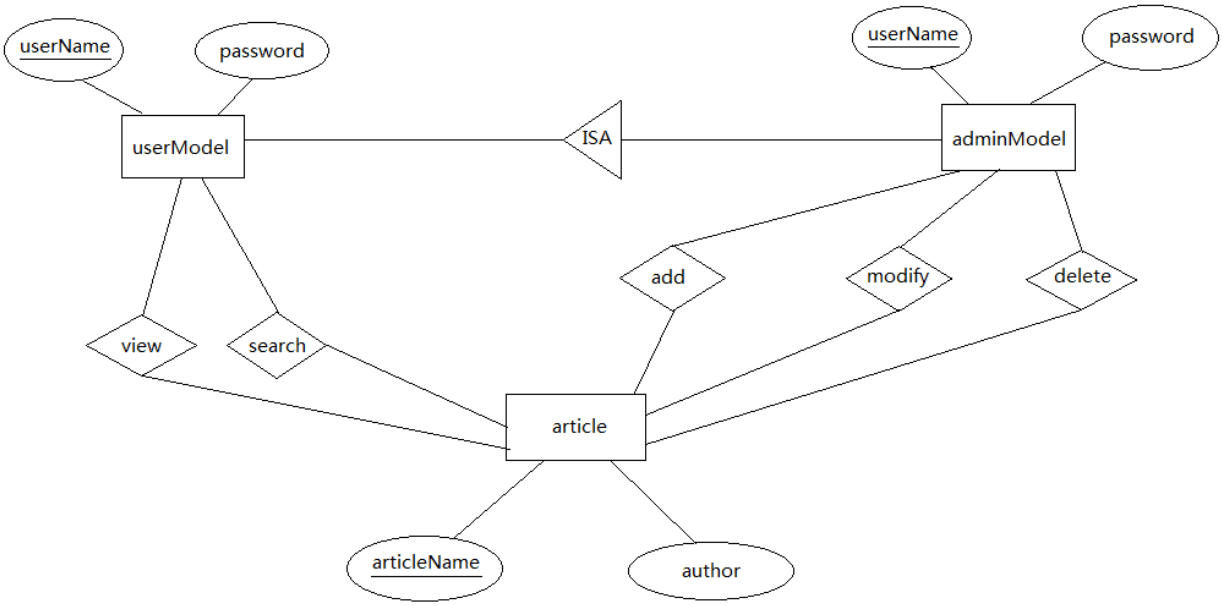

图 38 ER 图

11 技术机制

表 4 技术机制表

开发环境	Windows
开发平台	Apache
前台框架	JQuery + Smarty
后台框架	基于 thinkPHP 思想的自建框架
部署环境	Windows
语言	PHP5 + HTML + CSS + JAVAscript
数据库	MySQL5
开发工具	WampServer + Notepad++

本项目采用李开涌老师在《PHP 与 MVC 开发实战》中教授的基于 ThinkPHP 思想的自建 MVC 框架，Model 为数据库访问层，主要用于和数据库的交互与数据交换，View 层为图像层使用了 PHP 开源图形引擎 Smarty 做为前台页面展示以及 html 模板控制，Control 为业务逻辑层，用于处理 Model 层的数据并与 View 层交互。

服务器我们选择了 Apache，当前主流的服务器平台可选的有 Apache 和 Nginx，但是相比于 Nginx，Apache 更容易掌握，也适合于学生使用，因此我们选择了较为简单的 Apache。

后台框架选择了使用自建框架，一方面，李开涌老师介绍的这种自建框架具备了 ThinkPHP 的大多数优点，同时还给予了使用者极大的自由来定制出自己的框架，减少冗余功能。另一方面在搭建整个框架的过程中，小组成员对于 PHP 的框架设计有了更深的理解。

前台选择了比较常见的“黄金搭配”Html + CSS + JavaScript，因为其不仅功能强大，而且简单易用。

数据库选择了开源免费的关系型数据库 MySQL5。

我们的系统基本上全部使用免费的开源语言，一方面对于学生来说最大限度的节约了成本，另一方面，大量的开源资源可以让我们的开发工作最大程度的简化，让我们站在巨人的肩膀上摘苹果。

12 参考引用说明

本项目的部分网页样式采用开源模板网页的 css 样式，但是网页内容为本小组原创。我们还引用了第三方开源软件 Smarty 作为视图引擎，后台框架采用了李开涌老师在《PHP 与 MVC 开发实战》中介绍的自定义 PHP 框架。