**基于用户兴趣推荐的个性化**

**阅读APP详细设计说明书**

**Chenxu小组**

**2018年03月**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 文档修改情况记录 | | | | |
| 版本 | 修改状态 | 修改日期 | 修改摘要 | 修改人 |
| 1.0 | 创建 | 2018-03-20 | 创建详细设计文档1.0 | 陈煦 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目 录**

[1 引言 1](#_Toc29067)

[1.1 编写目的 1](#_Toc25196)

[1.2 名称解释 1](#_Toc16650)

[2 总体设计 1](#_Toc477)

[2.1 设计方法 1](#_Toc6159)

[2.2 软件结构 2](#_Toc16228)

[2.2.1 总体流程图 2](#_Toc28627)

[2.2.2 总体结构 2](#_Toc32429)

[2.2.3 关注模块 2](#_Toc30558)

[2.2.4 发现模块 3](#_Toc19582)

[2.2.5 消息模块 3](#_Toc17078)

[2.2.6 文章模块 4](#_Toc9656)

[2.2.7 我的模块 4](#_Toc30818)

[3 模块设计 5](#_Toc26442)

[3.1 关注模块 5](#_Toc6156)

[3.1.1 需求概述 5](#_Toc14020)

[3.1.2 流程图 5](#_Toc2902)

[3.1.3 信息展示模块 5](#_Toc28774)

[3.1.4 推荐关注模块 12](#_Toc25980)

[3.2 发现模块 16](#_Toc25178)

[3.2.1 需求概述 16](#_Toc27194)

[3.2.2 流程图 17](#_Toc22175)

[3.2.3 文章推荐模块 17](#_Toc18195)

[3.1.4 专题推荐模块 24](#_Toc20417)

[3.3 消息模块 31](#_Toc30409)

[3.3.1 需求概述 32](#_Toc4332)

[3.3.2 私信模块 32](#_Toc23940)

[3.4.3 评论模块 39](#_Toc20126)

[3.4.4 喜欢模块 43](#_Toc5596)

[3.4.5 关注模块 44](#_Toc18256)

[3.4 文章模块 47](#_Toc29566)

[3.4.1 需求概述 47](#_Toc5026)

[3.4.2 流程图 48](#_Toc2428)

[3.4.3 阅读模块 48](#_Toc4240)

[3.5.4 写作模块 58](#_Toc29737)

[3.5 我的模块 61](#_Toc8665)

[3.5.1 需求概述 62](#_Toc32383)

[3.5.2 流程图 62](#_Toc8605)

[3.5.3 用户管理模块 63](#_Toc3710)

[3.5.4 文章管理 76](#_Toc211)

# **引言**

## 编写目的

此设计说明书是APP开发的主要依据，描述程序系统结构，并进行程序设计，为开发人员提供工作基准文件，并对后续阶段的工作起指导作用。

预期读者为设该系统的系统分析员、程序员、测试员、实施人员等。

## 名称解释

|  |  |
| --- | --- |
| **名称** | **解释** |
|  |  |

# 总体设计

根据功能对系统进行划分，可分为关注模块、发现模块、消息模块、写作模块以及个人管理模块。

## 设计方法

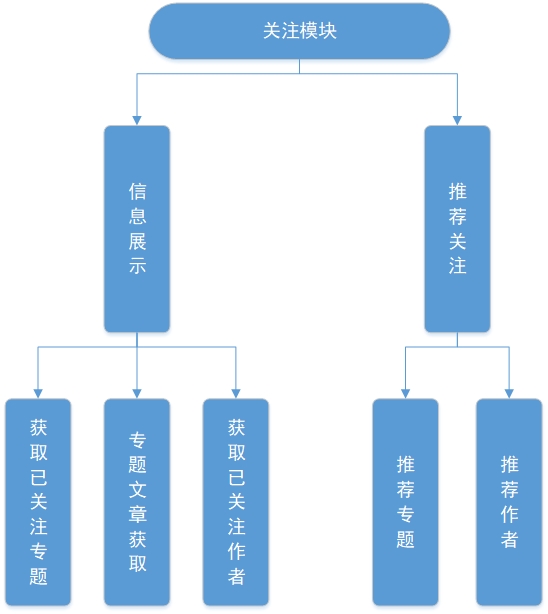
本APP采用传统的软件开发生命周期方法和敏捷开发相结合，采用自顶向下，逐步求精的面向对象软件设计方法。

## 软件结构

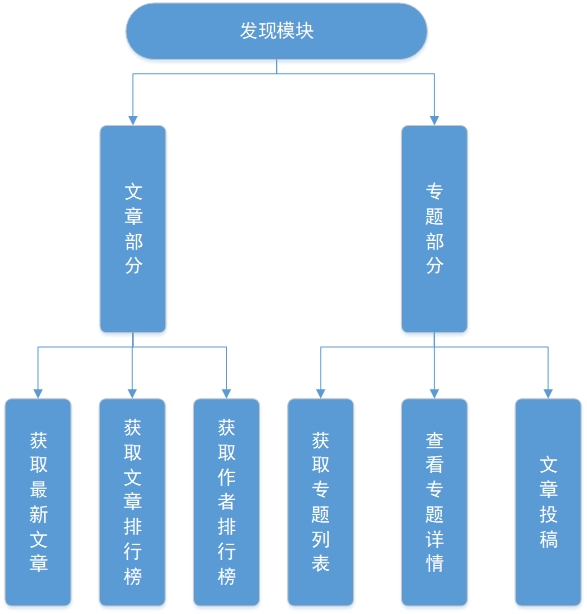
### 总体流程图

### 总体结构

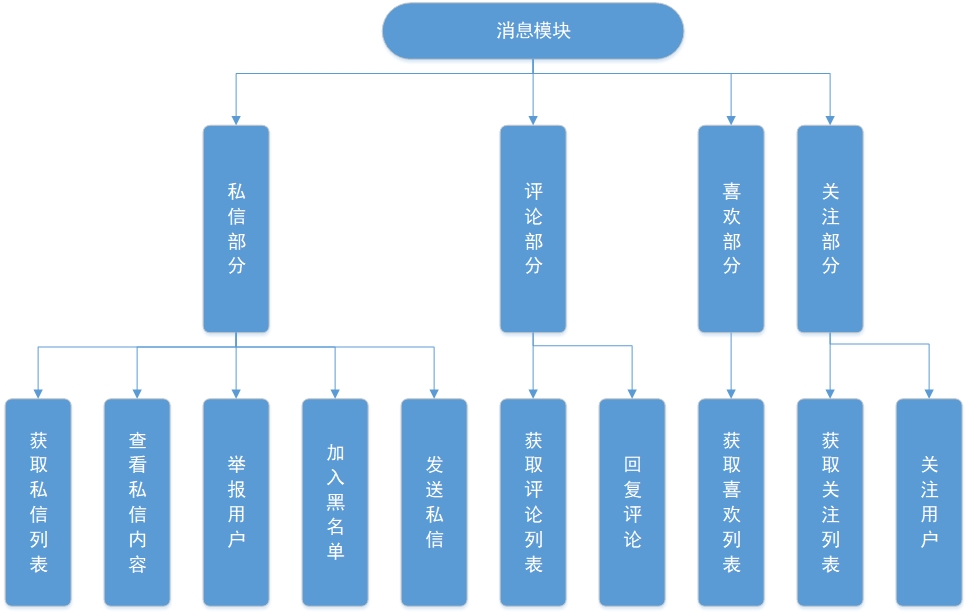
### 关注模块



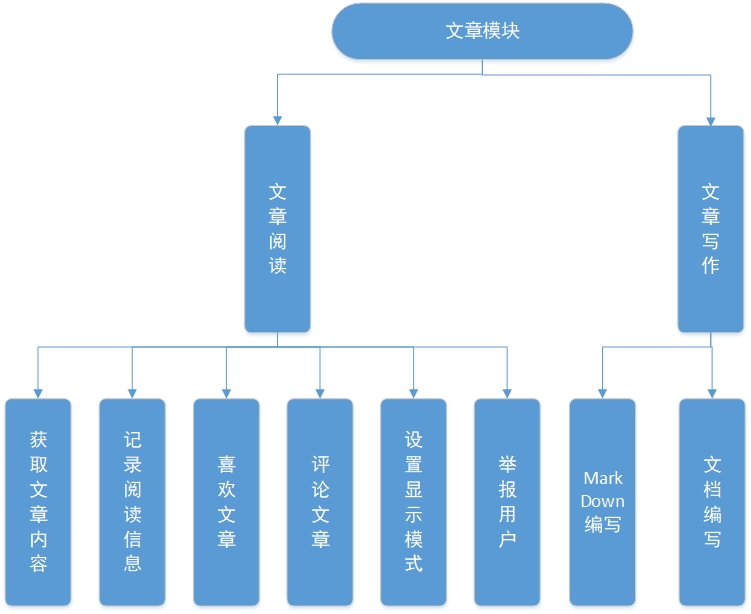
### 发现模块



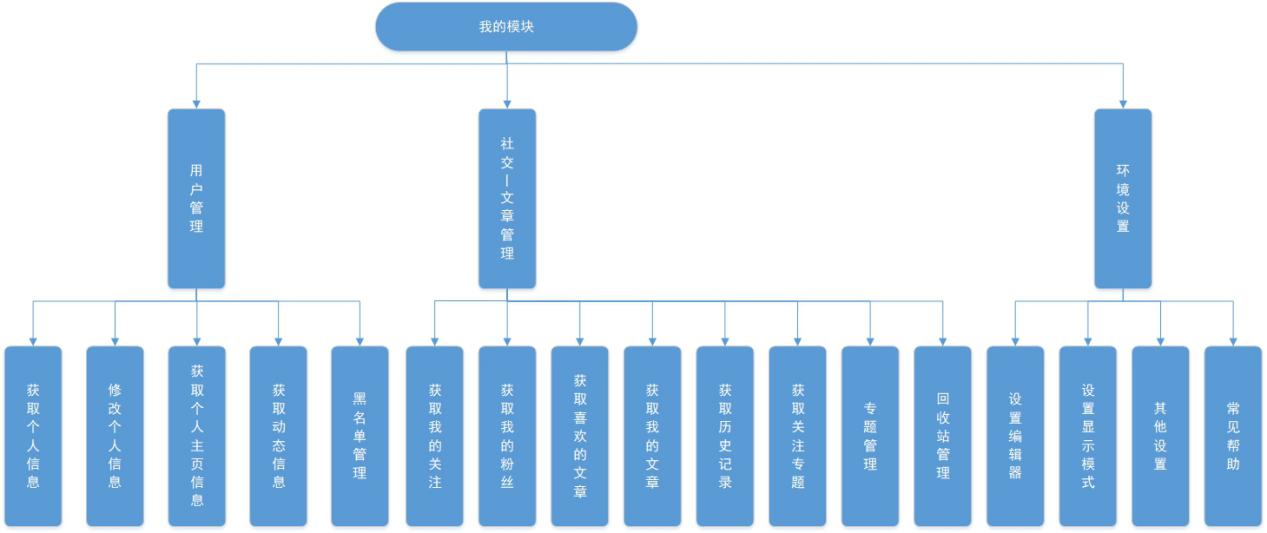
### 消息模块



### 文章模块



### 我的模块



# 模块设计

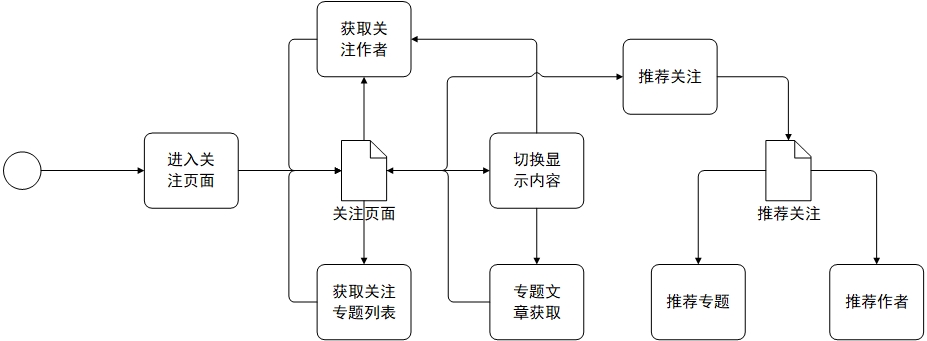
## 关注模块

关注模块分为两部分：信息展示以及推荐关注。

### **需求概述**

1. 信息展示：用户可以查看已关注专题的内容，例如旅游、美食。点击相应的专题可以查看该专题下的所有内容。除了专题之外，用户还可以查看已关注作者的所有文章。
2. 推荐关注：运用算法以及历史记录来分析用户喜爱的专题与作者，并推荐给用户这些专题与作者。

### 流程图



### **信息展示模块**

1. 具体描述：

关注界面显示当前用户已关注的所有专题以及作者，用户通过切换不同的专题来查阅相关专题下的文章，或者查看指定用户撰写的文章。

1. 具体流程：
2. 获取已关注专题
3. 模块描述

用户进入关注界面时，自动获取当前获取已关注的所有专题，并显示在界面的正上方。

1. 输入、输出

输入：用户账号。

输出：专题号、专题名称。

1. 数据库设计

### 专题关注表 rsubjfocus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Vsfid | Int | 否 | 主键 | 主键，唯一标识专题关注 | 专题关注号 |
| Ksid | Int | 否 | 外键 | 外键，专题表 | 专题号 |
| Saccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |
| Irank | Int | 否 |  |  | 排列顺序 |

1. 程序界面



（图：3.1.2.1 专题关注）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *FocusController* | |
| 方法设计 | public function getSubjectFocus(){} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的用户账号从 view\_subjectFocus 视图中检索相应的数据并返回给前端。 |

1. 模型类列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

1. 专题文章获取
2. 模块描述

用户可以通过切换专题名来获取该专题下指定数量的文章。

1. 输入、输出

输入：专题关注号、已显示文章数。

输出：作者头像、昵称、文章标题、缩略图、日期、部分内容、评论数、喜欢数以及文章号。

1. 实现过程

切换专题时前端会自动向后台发送请求，请求参数中携带专题关注号和已显示文章数。根据专题关注号到 view\_article 视图中检索出该专题下指定数量的文章（目前暂定6篇）。页面下拉到底端后开始获取新的文章，前端传递的另一个参数已显示文章数就用来过滤已经获取到的文章（mysql：limit 已显示文章数,6）。

1. 程序界面

参考图3.1.2.1专题关注。

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *FocusController* | |
| 方法设计 | public function getSubject() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的专题号和已显示文章数从 view\_articles 视图中检索六条（或少于六条）与专题号匹配的文章。 |

1. 模型列表

|  |  |
| --- | --- |
| *Focus* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

1. 获取已关注作者
2. 模块描述

用户点击作者即可获取当前用户已关注的所有作者，并以列表的形式呈现在界面上。

1. 输入、输出

输入：用户账号、已获取作者数。

输出：作者号、作者头像、昵称、最新一篇文章标题、未读文章数（待定）。

1. 实现过程

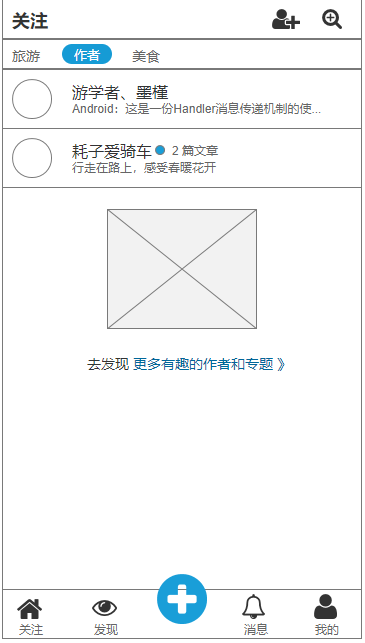
未读文章数有些繁琐，暂定先不做。其他过程同**专题文章获取**。

1. 数据库设计

### 作者关注表 hautfocus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Yafid | Int | 否 | 主键 | 主键，唯一标识作者关注 | 作者关注号 |
| Tautaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 作者账号 |
| Daccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |

1. 程序界面



（图：3.1.2.2作者关注）

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *FocusController* | |
| 方法设计 | public function getAuthors() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的用户账号和已获取作者数从 view\_authorFocus 视图中检索十位作者的信息并返回给前端。 |

C 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

### **3.1.4 推荐关注模块**

1. 具体描述

系统通过分析用户的阅读记录以及关注的专题与作者，推荐符合用户爱好与口味的文章与作者。

1. 具体实现
2. 推荐专题
3. 模块描述

系统自动推荐符合用户喜爱的专题。

1. 输入、输出

输入：无。

输出：专题号、专题照片、名称、文章数、关注数、简介。

1. 实现过程

拟实现的三个方法（仅供参考）：

1.根据专题间的关联性进行推荐——运用数据挖掘中的关联规则。

2.根据阅读文章所属的专题进行推荐——检索用户的历史记录，并判断该文章所属的专题，然后对这些专题出现的次数进行排序，推荐出现最多的专题。

3.推荐热门专题。

1. 程序界面



（图：3.1.3.1推荐专题）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *RecommendController* | |
| 方法设计 |  |
| 处理逻辑 |  |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

1. 推荐作者
2. 模块描述

系统自动推荐符合用户喜爱的作者。

1. 输入、输出

输入：无。

输出：作者账号、作者头像、昵称、个人介绍、最新的两篇文章。

1. 实现过程

拟实现的三个方法（仅供参考）：

1.根据作者间的关联性进行推荐——运用数据挖掘中的关联规则。

2.推荐已关注用户关注的作者。

3.推荐粉丝数较多的作者。

1. 程序界面



（图3.1.3.2推荐作者）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 |  |
| 处理逻辑 |  |

B. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

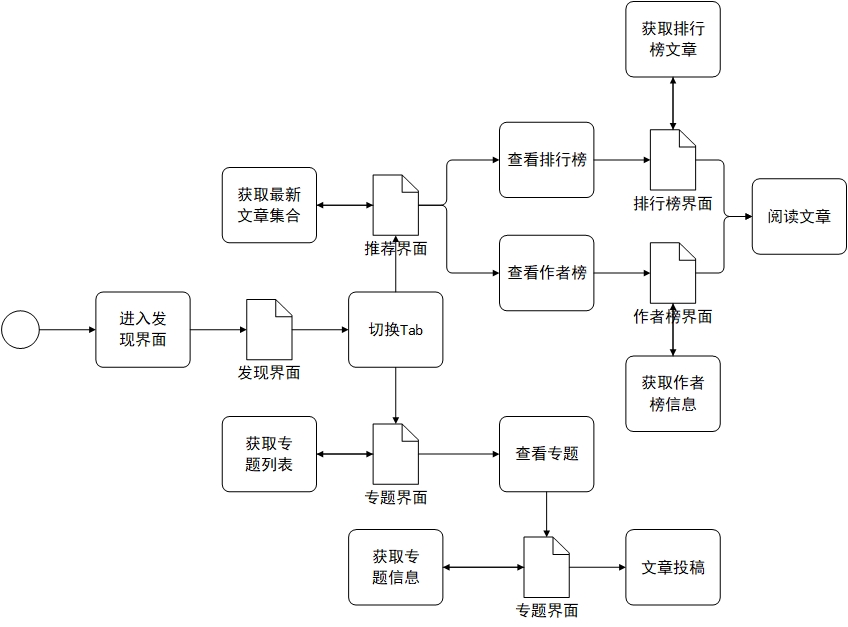
## 发现模块

发现模块是开眼APP的主要模块，而可分为文章部分以及专题部分，提供文章和专题的检索。

### **需求概述**

1. 文章推荐：需求名为推荐，实则并非推荐。文章推荐可细分为三部分，文章排行榜、作者排行榜以及最新文章。文章排行榜显示阅读数最多的文章，作者排行榜显示粉丝数最多的作者。
2. 专题推荐：在该界面可以查看关注数最多的专题，以及最新推出的专题。除此之外，用户也可以给自己喜爱的专题投稿。

### 流程图



### **文章推荐模块**

一．具体描述：

用户进入发现-推荐界面后，可以查看最新的文章，并且下拉屏幕可以刷新获取最新的文章。界面的正上方有文章排行榜和作者排行榜，分别展示阅读数最多的文章以及粉丝数最多的作者。

二．具体流程：

一）获取最新文章

1. 模块描述

当用户进入发现主界面时，会自动获取X篇最新的文章（暂定6篇）。用户也可以手动刷新以获取新的文章。

1. 输入、输出

（1）获取最新文章

输入：无/已获取最新文章的日期。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、所属专题、指定文字数的文章内容。

（2）下滑获取更多文章

输入：已获取文章数。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、所属专题、指定文字数的文章内容。

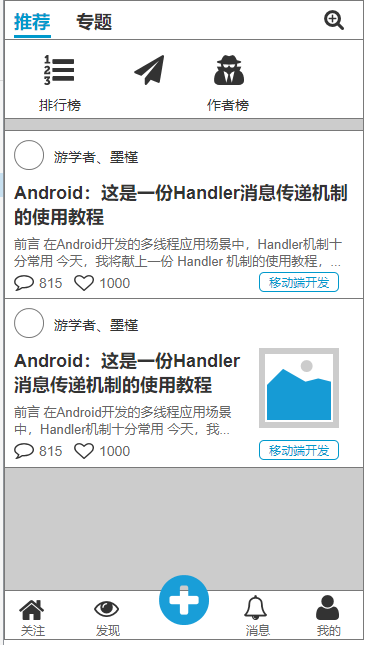
1. 实现流程

初始化加载：用户进入发现主界面，前端自动向后台发起请求，后台返回6篇最新的文章（若不足6篇，则返回全部），。

上拉加载：用户上拉屏幕，主动刷新页面以获取最新的文章。此时前端会传递一个参数“已获取最新文章的日期”，这个日期是最近一次获取最新文章的日期。后台将这个日期作为过滤条件，只有大于该日期的文章才会被认作是比之前获取的文章“更加新”。

下滑加载：用户下滑屏幕，动态加载更多文章，每次加载最多6篇文章。

1. 程序界面



（图3.2.3.1发现主界面）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getNewArticle() {} |
| 处理逻辑 | 1.通用安全性检查；  2.判断前端是否有传“已获取最新文章的日期”这个参数。若不存在，则默认该参数的值为“1970-01-01 00:00”；若存在，则直接使用能够该参数的值来作为过滤条件，从 view\_article 视图中获取满足条件的文章（最多6篇）并返回给前端。 |
| 方法设计 | public function getMoreArticle() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的参数“已获取文章数”从 view\_article 视图中获取文章（最多6篇）并返回给前端。 |

1. 模型类列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）获取文章排行榜

1. 模块描述

用户点击文章排行榜按钮即可进入到文章排行榜界面，此时应用会自动向后台发起请求以获取阅读数最多的文章。

1. 输入、输出

输入：已获取文章数。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、所属专题、指定文字数的文章内容。

1. 实现过程

具体流程同“获取最新文章”中的“初始化加载”和“下滑加载”。

1. 程序界面



（图3.2.3.2文章排行榜）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getArticleByRank() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的参数“已获取文章数”从 view\_article 视图中获取文章（最多6篇、按照阅读数降序排列）并返回给前端。 |

B. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

三）获取作者排行榜

1. 模块描述

用户点击作者排行榜按钮即可进入到作者排行榜界面，此时应用会自动向后台发起请求以获取粉丝数最多的作者。

1. 输入、输出

输入：已获取作者数。

输出：作者头像、昵称、最新的一篇文章标题。

1. 实现过程

具体流程同“获取最新文章”中的“初始化加载”和“下滑加载”。

1. 程序界面



（图3.2.3.3作者排行榜）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 | public function getAuthorByRank() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的参数“已获取作者数”从 view\_author 视图中获取作者（最多10人、按照粉丝数降序排列）并返回给前端。 |

B. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

### **3.1.4 专题推荐模块**

一．具体描述

用户通过切换左上角的Tab，即可进入到专题界面。在该界面，用户可以查看最热门的专题以及最新创建的专题。令人兴奋的是，用户可以向专题投稿（注意，一篇文章只能向一个专题投稿，且投稿了无法撤销）。

二．具体实现

一）获取专题列表

1. 模块描述

用户进入专题界面自动获取专题列表，包括最热门专题以及最新专题。

1. 输入、输出
2. 获取专题

输入：无。

输出：专题号、专题封面、专题名称。

1. 下滑获取更多

输入：已获取专题数。

输出：专题号、专题封面、专题名称。

1. 实现过程

初始化加载：用户刚进入界面，前端就会向后台发起请求。相较于文章部分，专题部分要返回两部分的内容，最热门的专题（存放在一个数组中）以及最新的专题（存放在另一个数组中）。

下滑加载：下滑加载只需要获取最新专题即可，获取方式同“获取最新文章”的“下滑加载”，即根据前端传递的“已获取专题数”来检索之后的专题。

1. 程序界面



（图3.2.4.1专题界面）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *SubjectController* | |
| 方法设计 | public function getSubjects() {} |
| 处理逻辑 | 1.通用安全性检查；  2.分别从 view\_subject 视图中获取专题（一个按照关注数降序获取，具体数量待定；另一个按照创建日期降序获取，具体数量待定）并返回给前端。 |
| 方法设计 | public function getMoreSubjects() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的“已获取专题数”从 view\_subject 视图中获取指定数量的专题（具体数量待定）并返回给前端。 |

B. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）查看专题详情

1. 模块描述

用户点击感兴趣的专题封面（或专题名称）即可进入到专题详情页面。在该页面会显示有关专题的各项信息，以及该专题下最热门的文章以及最新投稿的文章。

2. 输入、输出

1. 获取专题信息

输入：专题号。

输出：专题封面、专题名称、专题所属者、专题总计文章数、专题介绍、专题关注数。

1. 获取最热门文章

输入：专题号、已获取文章数。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、所属专题、指定文字数的文章内容。

1. 获取最新文章

输入：专题号、已获取文章数。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、所属专题、指定文字数的文章内容。

1. 实现过程

用户进入专题详情页面时，前端会立即向后台发起请求以获取该专题的信息。同时再发起请求获取最热门文章（默认Tab切换到“最热门”）。

最热门文章和最新文章的处理过程同之前的“获取最新文章”（这一类操作都相同）。

1. 程序界面



（图3.2.4.2专题详情）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *SubjectController* | |
| 方法设计 | public function getSubjectInfo() {} |
| 处理逻辑 | 1.通用性安全检查；  2.根据前端传递的专题号，从 view\_subject 视图中检索相应专题号所对应的信息，并将这些信息返回。 |
| 方法设计 | public function getPopArticleInSub() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的“专题号”和“已获取文章数”从 view\_article 视图中获取指定数量的文章（按阅读量降序排列，具体数量待定）并返回给前端。 |
| 方法设计 | public function getNewArticleInSub() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的“专题号”和“已获取专题数”从 view\_article 视图中获取指定数量的文章（按文章日期降序排列，具体数量待定）并返回给前端。 |

B. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

1. 文章投稿

1. 模块描述

在专题详情页，用户可以投稿自己的文章。需要注意的是，一篇文章只能投稿给一个专题，之后无法更改，只有删除重写再投稿。未投稿的文章在应用中不显示专题名。

1. 输入、输出
2. 获取未投稿文章

输入：作者账号（用户账号）。

输出：文章号、文章标题。

1. 文章投稿

输入：文章号、作者账号（用户账号）。

输出：文章投稿操作的响应代码和响应消息。

1. 实现过程

用户点击“投稿”按钮后，会从下往上弹出一个面板。在该面板上显示当前用户未投稿的文章，此时用户点击要投稿文章后方的“投稿”按钮，即可进行投稿。投稿需要专题所属者进行审核，审核通过方可成功投稿。

1. 程序界面



（图3.2.4.3文章投稿）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getArticleNoCon() {} |
| 处理逻辑 | 1.通用性安全检查；  2.根据前端传递的用户账号，从 view\_article 视图中检索相应用户账号所有的未投稿文章（专题号为0），并将这些信息返回。 |
| 方法设计 | public function contribute() {} |
| 处理逻辑 | 1.通用安全性检查；  2.根据前端传递的“文章号”和“用户账号”修改 larticle表对应数据的专题号。 |

B. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

C. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

## 消息模块

用户可以查看已收到的私信，并决定对私信发送者进行举报、加入黑名单以及回信等操作。用户也可以主动向指定的用户写信，但只能是已关注的用户。除此之外，用户还可以查看评论回复、文章喜欢以及用户关注等相关消息。

### **需求概述**

（1）获取私信：进入网上预约挂号系统，根据地区选择需要挂号的医院。进入医院网站首页，初次登陆系统的用户，需要先注册才能登录系统进行挂号；如果是非初次登录，输入用户名和密码，即可登录。系统可以自动核对用户信息，校验通过后则可进行预约挂号。患者注册时，必须如实填写患者姓名，身份证号，手机号。一个身份证号和手机号只能注册一个用户名。

### **私信模块**

一．具体描述：

查看私信列表以及指定私信的内容、对私信发送者进行举报、加入黑名单以及回信等操作。同样，也允许用户主动向其他用户主动发送私信。

二．具体流程：

一）获取私信列表

1. 模块描述

获取当前用户已收到的所有私信，并以列表的形式呈现在页面上。

1. 输入、输出

输入：用户账号。

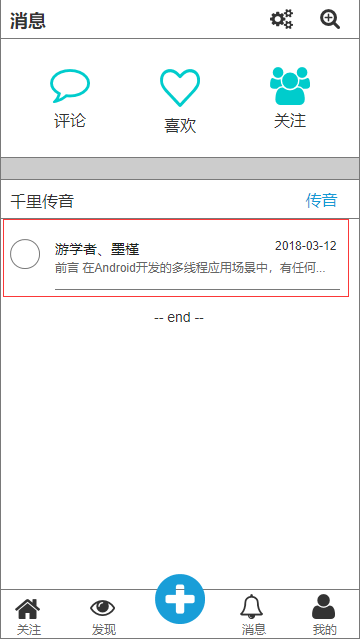
输出：私信发送者头像、昵称、日期、最新的一条消息。

1. 数据库设计

### 私信表 smessage

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Dmid | Int | 否 | 主键 | 自增长，唯一标识私信 | 私信号 |
| Esender | varchar(12) | 否 |  | 发送私信者的账号 | 发送者账号 |
| Rdate | char(16) | 否 |  | 日期格式：YYYY-MM-dd HH:mm | 发送日期 |
| Pcontent | varchar(128) | 否 |  | 最多不能超过128个中文 | 私信内容 |
| Oreceiver | varchar(12) | 否 |  | 接收私信者的账号 | 接收者账号 |
| GisRead | char(1) | 否 |  | 默认为0,0：未读,1：已读 | 是否已读 |

1. 程序界面



（图：3.4.2.1 消息主界面）

1. 类设计
2. 类图
3. 控制器列表

|  |  |
| --- | --- |
| *MessageController* | |
| 方法设计 | public function getList() {} |
| 处理逻辑 | 1.判断请求参数的合法性以及身份认证；  2.根据接收者账号从 view\_message 视图中获取发送者账号不重复的所有私信(select distinct)； |

1. 模型类列表

|  |  |
| --- | --- |
| *Message* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）查看私信内容

1. 模块描述

用户点击相应的私信发送者，即可进入到与当前私信发送者交流的界面，在界面初始化时获取与该私信发送者有关的八（待定）条最新私信内容。

1. 输入、输出

输入：发送者账号、接收者账号（用户），当前私信条数。

输出：发送者头像、发送日期、私信内容。

1. 程序界面



（图：3.4.2.2 私信界面）

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *MessageController* | |
| 方法设计 | public function getContent(){} |
| 处理逻辑 | 1.判断请求参数的合法性以及身份认证；  2.根据发送者、接受者账号从 view\_message 视图中查找指定数量（暂定为8条）的私信。 |

C 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

三）举报用户

1. 模块描述

用户可以举报给你发送私信的用户，在举报时需要注明举报的类型以及理由。

1. 输入、输出

输入：举报类别、发送者账号、接收者账号（用户）、举报原因。

输出：举报用户操作返回的响应代码和相关的响应消息。

1. 程序界面

参考图：3.4.2.2 私信界面。

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *ReportController* | |
| 方法设计 |  |
| 处理逻辑 |  |

C 模型列表

|  |  |
| --- | --- |
| *Report* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

四）加入黑名单

1. 模块描述

用户可以将当前私信用户加入到黑名单中，加入黑名单之后可以自动屏蔽这些用户的私信。

1. 输入、输出

输入：发送者账号、接收者账号（用户账号）。

输出：加入黑名单操作返回的响应代码和相关的响应消息。

1. 程序界面

参考图：3.4.2.2 私信界面。

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *ReportController* | |
| 方法设计 |  |
| 处理逻辑 |  |

C 模型列表

|  |  |
| --- | --- |
| *Report* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

五）发送私信

1. 模块描述

用户可以主动给已关注的用户发送私信。

1. 输入、输出

输入：发送者账号（用户账号）、接收者账号、私信内容。

输出：发送私信操作返回的响应代码和相关的响应消息。

1. 程序界面

参考图：3.4.2.2 私信界面。

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *ReportController* | |
| 方法设计 |  |
| 处理逻辑 |  |

C 模型列表

|  |  |
| --- | --- |
| *Report* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

### **3.4.3 评论模块**

一．具体描述

用户可以在消息主界面中点击“评论”图标从而进入到评论界面，在该界面可以查看到有关当前用户的所有评论。用户也可以对评论进行回复。

二．具体实现

一）获取评论列表

1. 模块描述

用户进入到评论界面，界面会自动获取最新的八条评论并呈现在界面上。

1. 输入、输出

输入：接收者账号（用户账号）。

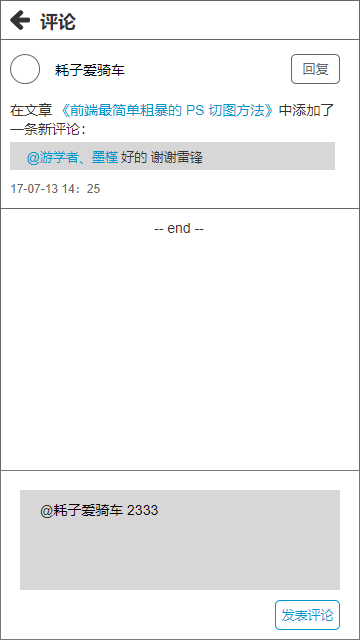
输出：评论者账号、头像、昵称、评论日期、评论内容、评论文章。

1. 数据库设计

### 评论表 dmesscomment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Ecid | Int | 否 | 主键 | 自增长，唯一标识评论记录 | 评论号 |
| Vsender | varchar(12) | 否 |  |  | 发送者账号 |
| Dreceiver | varchar(12) | 否 |  |  | 接收者账号 |
| Odate | char(16) | 否 |  | 评论发布的日期 | 评论日期 |
| Scontent | varchar(128) | 否 |  | 评论最多128个中文 | 评论内容 |
| Particle | Int | 否 | 外键 | 文章表外键 | 评论文章 |
| Rtype | char(1) | 否 |  | 默认0，0：评论，1：回复 | 评论类型 |

1. 程序界面



（图：3.4.3.1评论界面）

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *CommentController* | |
| 方法设计 |  |
| 处理逻辑 |  |

C 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）回复评论

1. 模块描述

用户可以对他人给出的评论进行回复。

1. 输入、输出

输入：发送者账号、接收者账号（用户账号）、评论内容。

输出：回复评论验证操作的响应代码和响应消息

1. 程序界面

参考图：3.4.3.1评论界面

1. 类设计

A 类图

B 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 |  |
| 处理逻辑 |  |

C 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

D 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

### 3.4.4 喜欢模块

一．具体描述

用户可以在消息主界面中点击“喜欢”图标从而进入到喜欢界面，在该界面可以查看到有关当前用户的所有喜欢。

二．具体实现

一）获取喜欢列表

1. 模块描述

用户进入到喜欢界面，界面会自动获取最新的八条喜欢并呈现在界面上。

1. 输入、输出

输入：接收者账号（用户账号）。

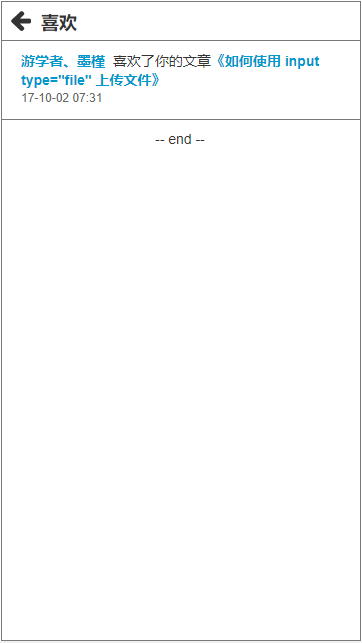
输出：发送者昵称、喜欢日期、喜欢文章。

1. 数据库设计

### 喜欢表 hmesslike

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Tlid | Int | 否 | 主键 | 自增长，唯一标识喜欢记录 | 喜欢号 |
| Ssender | varchar(12) | 否 |  |  | 发送者账号 |
| Ireceiver | varchar(12) | 否 |  |  | 接收者账号 |
| Wdate | char(16) | 否 |  | 喜欢发布的日期 | 喜欢日期 |
| Varticle | Int | 否 | 外键 | 文章表外键 | 喜欢文章 |

1. 程序界面



（图3.4.4.1 喜欢界面）

1. 类设计

### 3.4.5 关注模块

一．具体描述

用户可以在消息主界面中点击“关注”图标从而进入到关注界面，在该界面可以查看到有哪些用户关注了当前用户。用户也可以关注这些用户。

二．具体实现

一）获取关注列表

1. 模块描述

用户进入到关注界面，界面会自动获取最新的八条关注并呈现在界面上。

1. 输入、输出

输入：接收者账号（用户账号）。

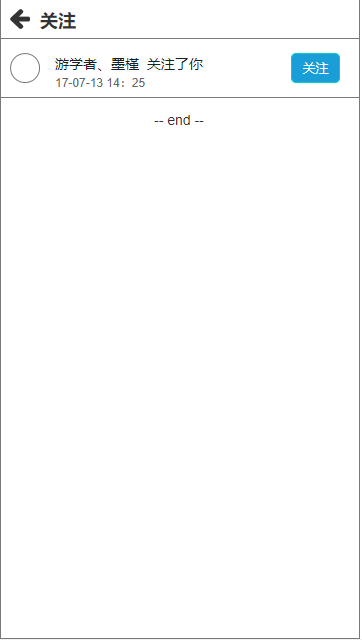
输出：发送者昵称、头像、关注日期。

1. 数据库设计

### 关注表 cmessfocus

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Ufid | Int | 否 | 主键 | 自增长，唯一标识关注记录 | 关注号 |
| Xsender | varchar(12) | 否 |  |  | 发送者账号 |
| Greceiver | varchar(12) | 否 |  |  | 接收者账号 |
| Qdate | char(16) | 否 |  | 关注发布的日期 | 关注日期 |

1. 程序界面



（图3.4.5.1 关注界面）

1. 类设计
2. 关注用户

1. 模块描述

用户可以对关注自己的用户进行关注。

1. 输入、输出

输入：发送者账号，接收者账号（用户账号）。

输出：关注用户操作返回的响应代码和相关的响应消息。

1. 程序界面

参考图3.4.5.1关注界面

1. 类设计

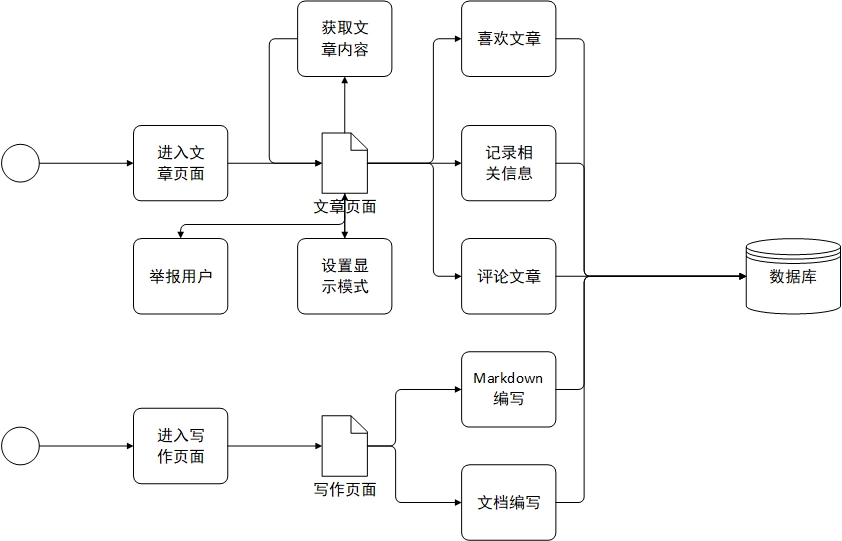
## 文章模块

将关注、发现、我的等模块中有关文章的内容单独抽离出来成为一个完整的模块。文章模块主要分为两部分：阅读部分以及写作部分。

### **需求概述**

1. 文章阅读：无论用户通过何种途径进入到文章页面，都能够完整地将文章的图文呈现。系统提供多种阅读显示模式，用户可以自行进行切换。除了阅读之外，用户还可以对文章进行评论、对他人的评论进行评论，或者将文章标记为喜欢。如果文章中有不良信息，用户可以举报该用户，令其删除该文章。
2. 文章写作：用户可以输入普通文档格式的文章，也可以使用 markdown 语法书写 markdown 文章。文章中可以内嵌图片以及链接，字体的大小与颜色也都可以进行相应的修改，至于是否能编写表情，仍待商榷。

### 流程图



### **阅读模块**

一．具体描述：

关注界面显示当前用户已关注的所有专题以及作者，用户通过切换不同的专题来查阅相关专题下的文章，或者查看指定用户撰写的文章。

二．具体流程：

一）获取文章内容

1. 模块描述

用户进入指定文章界面时，自动获取该文章的内容（作者信息、文章信息、文章标题与正文），并显示在界面中。

1. 输入、输出

（1）获取文章相关内容：

输入：文章号、用户账号。

输出：文章标题、文章正文、文章所属专题、文章日期、喜欢数、评论数、阅读数、作者头像、昵称。

（2）获取评论

输入：文章号、已获取文章数。

输出：五条文章评论。

1. 实现过程

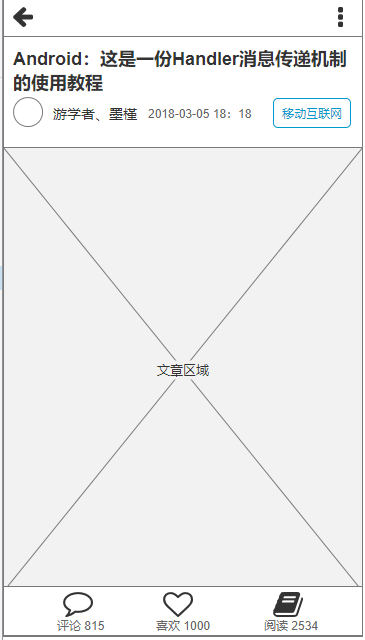
页面加载时，前端向后台发起“获取文章相关内容”请求。后台收到前端发送的文章号，根据文章号从 view\_article 视图中获取指定文章信息，并返回给前端。前端收到后台返回的信息将这些信息渲染到页面上，同时记录下文章正文+顶部的高度。当用户下滑页面到指定高度时（通过计算），向后台发送“获取评论”请求，后台再将评论按照顺序依次返回给前端。

1. 数据库设计

### 文章表 larticle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Xaid | varchar(22) | 否 | 主键 | 主键，账号+YYMMHHmmss | 文章号 |
| Psid | Int | 是 | 外键 | 外键，专题表，默认0 | 所属专题 |
| Vaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 所属作者 |
| Dtitle | varchar(32) | 否 |  | 最长32个中文 | 文章标题 |
| Tcontent | varchar(8192) | 否 |  | 最长8192个中文 | 文章正文 |
| Rdate | char(16) | 否 |  | YYYY-MM-dd HH:mm | 文章日期 |
| Klike | Int | 是 |  | 默认0 | 喜欢数 |
| Wcomment | Int | 是 |  | 默认0 | 评论数 |
| Oread | Int | 是 |  | 默认0 | 阅读数 |
| Kpiture | varchar(32) | 是 |  | 文章中第一张图片路径 | 缩略图 |

1. 程序界面



（图：3.5.3.1 文章主界面）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getArticle(){} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的文章号从 view\_article 视图中检索相应的数据并返回给前端。 |
| 方法设计 | public function getArticleComments(){} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的文章号、已获取评论数从 view\_comment 视图中检索相应的数据并返回给前端。 |

1. 模型类列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）记录阅读信息

1. 模块描述

该部分有两项操作，首先修改文章的阅读数，使其加一。第二项操作，将当前文章添加到当前用户的阅读历史记录中。注意，当前模块与“获取文章内容”模块都是在页面加载时就自动执行，其先后顺序为：“记录阅读信息”>“获取文章内容”。也就是说，先执行记录阅读信息，后执行获取文章内容（这么做的原因在于阅读数的变化）。

1. 输入、输出

输入：无。

输出：无。

1. 实现过程

后台收到前端请求后，根据“获取文章内容”传递的文章号到数据表 article 中修改阅读数，使其+1。推荐使用存储过程或者函数以减少后台与数据库之间的交互次数。

1. 数据库设计

### 阅读历史记录表 ohistory

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Chid | Int | 否 | 主键 | 主键，唯一标识历史记录 | 历史记录号 |
| Gaid | Int | 否 | 外键 | 外键，文章表 | 文章号 |
| Aaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |
| Pdate | char(16) | 否 |  | YYYY-MM-dd HH:mm | 阅读日期 |

1. 程序界面

参考图3.5.3.1文章主界面

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function modifyReadCount() {} |
| 处理逻辑 | 根据getArticle()方法传递的文章号，修改 larticle 表中相应文章号的阅读数，使其自增一。 |
| *UserController* | |
| 方法设计 | public function addRedHistory() {} |
| 处理逻辑 | 将getArticle()传递的文章号、用户账号，以及获取的系统时间添加到 ohistory 表中。 |

1. 模型列表

|  |  |
| --- | --- |
| *Focus* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

三）喜欢文章

1. 模块描述

用户点击文章主界面底部的爱心按钮，即可将本篇文章标记为喜欢。同样，用户也能够取消喜欢的文章。可以通过爱心按钮的颜色，来辨别当前文章是否已标记为喜欢。空心表示未喜欢，实心表示喜欢。

1. 输入、输出

输入：文章号、用户账号、作者账号。

输出：喜欢操作返回的响应代码和响应消息。

1. 实现过程

后台在收到前端的请求后，需要先将当前的文章的喜欢数+1，然后将该文章添加到当前用户的喜欢文章列表中，最后再给作者发送私信，告诉他“你的文章被某某用户喜欢了”。

若是取消喜欢，则先将当前文章的喜欢数-1，然后将该文章从当前用户的喜欢文章列表中移除。注意，取消喜欢不需要给作者发送私信（毕竟不太友好）。

1. 数据库设计

### 用户喜欢表 fuserlike

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Hulid | Int | 否 | 主键 | 主键，唯一标识用户喜欢 | 用户喜欢号 |
| Taid | Int | 否 | 外键 | 外键，文章表 | 文章号 |
| Rautaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 作者账号 |
| Vaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |

1. 程序界面

参考图3.5.3.1文章主界面

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function addArticleLike() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的文章号将 larticle 表中相应数据的喜欢数+1。 |
| 方法设计 | public function removeArticleLike() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的文章号将 larticle 表中相应数据的喜欢数-1。 |
| *UserController* | |
| 方法设计 | public function addUserLike() {} |
| 处理逻辑 | 将addArticleLike()传递的文章号、用户账号和作者账号添加到fuserlike 表中。 |
| 方法设计 | public function removeUserLike() {} |
| 处理逻辑 | 根据removeArticleLike()传递的文章号、用户账号，从 fuserlike表中将指定的数据删除。 |
| *MessageController* | |
| 方法设计 | public function addMessageLike() {} |
| 处理逻辑 | 将 addArticleLike()传递的文章号、用户账号和作者账号以及自定义的文字添加到 hmesslike 表中。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

四）评论文章

1. 模块描述

用户可以自由地评论文章，也可以对已有的评论进行评论。若对文章评论，则默认@作者；对已有评论进行评论，则@评论的用户。

1. 输入、输出

输入：文章号、评论用户（作者）账号、当前用户账号、评论内容、评论类型（0：对文章评论，1：对评论进行评论）。

输出：评论操作返回的响应代码和响应消息。

1. 实现过程

后台在收到前端的请求后，需要先将当前的文章的评论数+1，然后给作者（评论用户）发送私信，告诉他“某某用户评论了你XXX文章，说了‘哔哔哔’”。

1. 程序界面



（图3.5.4.1文章评论）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function addArticleComment() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的文章号将 larticle 表中相应数据的评论数+1。 |
| *MessageController* | |
| 方法设计 | public function addMessageComment() {} |
| 处理逻辑 | 将 addArticleComment()传递的文章号、用户账号和作者（已评论用户）账号、评论内容、评论类型添加到 hmesscomment 表中。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

五）设置显示模式

1. 模块描述

用户可以设置文章的字体大小以及修改阅读模式（白天模式和夜晚模式）。这部分完全由前端来实现，不需要前后端进行交互。

六）举报用户

1. 模块描述

参考消息模块中的举报用户。

### **3.5.4 写作模块**

一．具体描述

用户点击APP下方菜单栏的加号按钮，即可进入到写作界面。写作模式分为两种，普通文档以及 markdown 文档。关于文档模式切换放置在哪个界面仍待商榷。

二．具体实现

一）文档编写

1. 模块描述

用户能够在指定的区域内输入文字、图片。文字可以进行加粗、斜体等操作，也可以标题花处理。图片会自动压缩（尺寸以及像素比），符合界面的大小。

1. 输入、输出

（1）图片上传：

输入：图片的二进制数据（文章号，取决于实现过程中采用的办法）。

输出：图片在服务器上的地址。

（2）文章上传：

输入：文章标题、正文、作者账号（用户账号）、文章号（待定，取决于采用的办法）。

输出：文档上传操作的响应代码和响应消息。

（3）放弃写作：

输入：文章号或图片地址集合（取决于实现过程中采用的办法）。

输出：无。

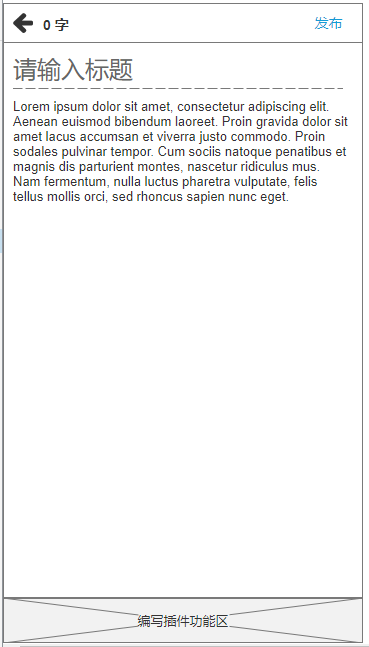
1. 实现过程

首先需要明确的一点是文章和图片是分开上传。问题的关键在于图片上传的时机。若是在最终上传文章的时候，先上传图片后上传文章，这会让用户多等待一段时间。若在用户插入图片时就上传图片，一旦用户最终放弃编写文章，那么上传的图片就会占据服务器的磁盘空间，造成不必要的浪费。目前的方向：1.加快最终上传的速度；2.插入图片时即上传，在用户放弃上传文章时，回收这片文章已上传的图片。

办法一：文章号不要自增长，而是通过某种方式生成。在用户编写文章的时候就拟生成一个文章号给当前用户。如果用户确实发表了这篇文章，就把这个文章号作为真正的文章号。若用户不发表，就废弃这个文章号，并把与当前文章号关联的图片全部删除。

办法二：前端记录下所有图片的地址，在用户放弃编辑的时候将这些图片的地址发送给后台，让后台删除这些图片。

1. 程序界面



（图：3.5.4.1写作）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function writePlain() {} |
| 处理逻辑 | 1.通用性安全检测；  2.将前端传来的数据插入到 article 表中。 |
| 方法设计 | public function uploadPicture() {} |
| 处理逻辑 | 1.通用性安全检测；  2.将前端传来的二进制图片保存到服务器指定的文件夹内，然后返回给前端图片地址。 |
| 方法设计 | public function giveUpWriting() {} |
| 处理逻辑 | 1.通用性安全检测；  2.处理方式参考实现过程中的两个办法。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）markdown 编写

1. 模块描述

Markdown 编写与普通文档编写大致一致，除了前端动态实现代码转换为文本稍有困难之外，在交互方面反而要容易一些，因为 markdown 的图片只能是引用形式，因此就不需要涉及到图片上传。

1. 输入、输出

输入：文章标题、正文、作者账号（用户账号）、文章号（待定，取决于采用的办法）。

输出：文档上传操作的响应代码和响应消息。

1. 程序界面

参考图3.5.4.1写作。

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function writeMarkdown() {} // 考虑是否要将markdown和普通文档合并成一个方法。 |
| 处理逻辑 | 1.通用性安全检测；  2.将前端传来的数据插入到 article 表中。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

## 我的模块

我的模块为系统四大模块之一，主要用于用户管理，例如修改用户信息、管理黑名单、阅读记录、已编写的文章和专题等。其次是对系统的配置，例如设置显示模式、默认编辑器等。

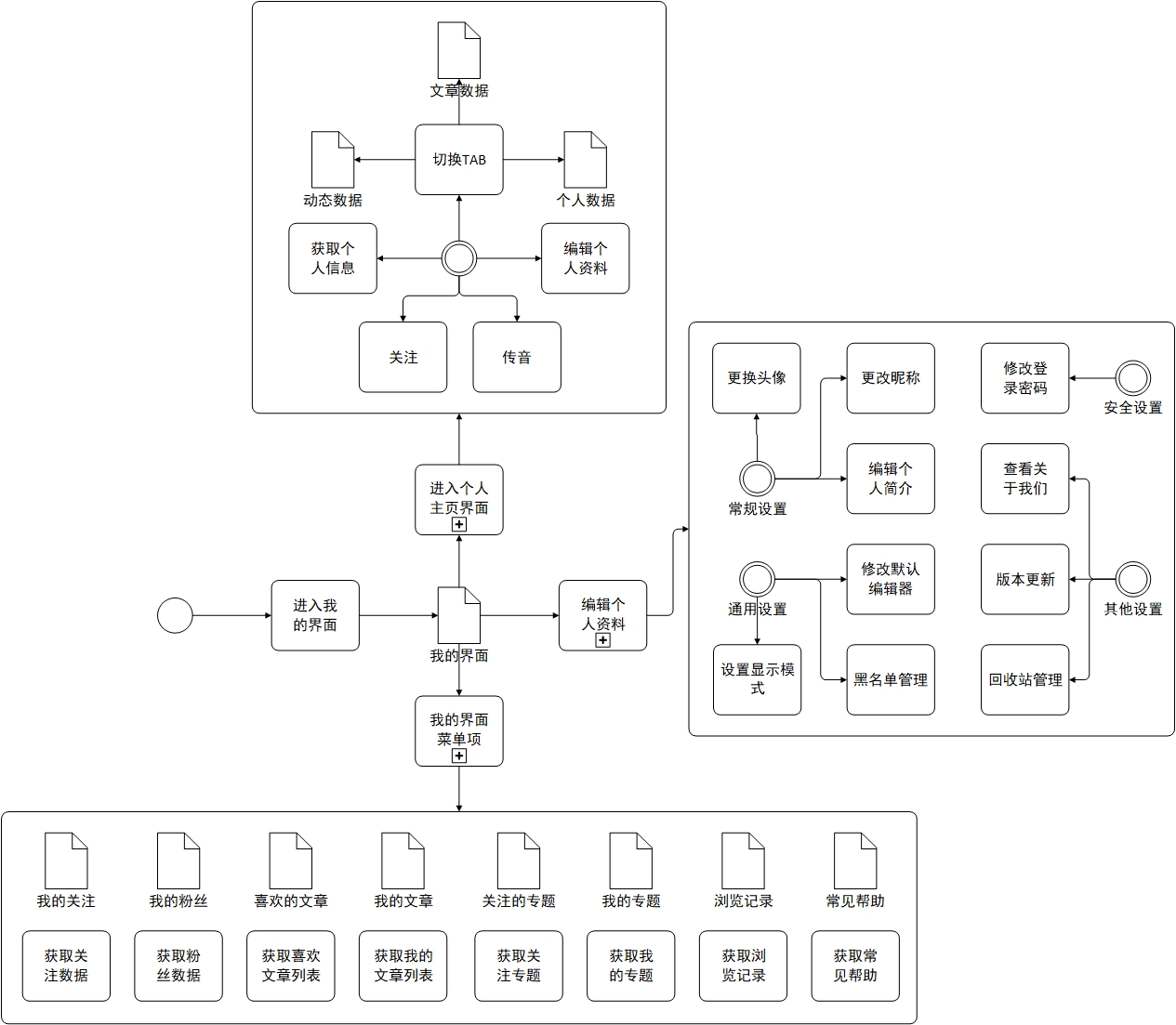
### **需求概述**

（1）用户管理：用户能够在“我的”模块中修改自己的昵称、头像和个人简介，也能够修改密码。用户能够查看个人主页的信息，例如关注数，粉丝数以及文章数，还有自己的阅读历史记录。

（2）文章、专题管理：用户可以查看自己编写文章与专题，并且可以自行创建专题。除此之外，还可以查看喜欢的文章以及关注的专题。

（3）应用设置：用户可以修改应用的显示模式和默认编辑器。

### 流程图



### **用户管理模块**

一．具体描述：

关注界面显示当前用户已关注的所有专题以及作者，用户通过切换不同的专题来查阅相关专题下的文章，或者查看指定用户撰写的文章。

二．具体流程：

一）获取个人信息

1. 模块描述

用户进入指定“我的界面”时，自动获取当前用户的信息，包括用户昵称、头像该文章的内容（作者信息、文章信息、文章标题与正文），并显示在界面中。

1. 输入、输出

输入：用户账号。

输出：用户昵称、头像、个人简介、我的关注数、我的粉丝数、喜欢文章数、我的文章数、关注专题数。

1. 实现过程

页面加载时，前端主动向后台发起请求。后台收到前端发送的用户账号，从 view\_user 视图中获取指定文章信息，并返回给前端。

1. 数据库设计

### 用户表 ruser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Raccount | varchar(12) | 否 | 主键 | 主键，唯一标识用户表 | 用户账号 |
| Ipassword | varchar(20) | 否 |  | 6-20位长度，任意字符 | 登录密码 |
| Hnickname | varchar(10) | 否 |  | 最多10个中文 | 用户昵称 |
| Uhead | varchar(30) | 是 | 默认 | 记录图片在服务器的路径 | 用户头像 |
| Jbrief | varchar(48) | 是 | 默认 | 最多48个中文 | 个人简介 |
| Hcountfocus | Int | 是 | 默认 | 默认为0 | 关注数 |
| kcountfans | Int | 是 | 默认 | 默认为0 | 粉丝数 |
| Ycountlike | Int | 是 | 默认 | 默认为0 | 喜欢数 |
| Xcountarticle | Int | 是 | 默认 | 默认为0 | 文章数 |
| Ecountsubject | Int | 是 | 默认 | 默认为0 | 专题数 |
| Rindexback | varchar(30) | 是 | 默认 | 主页背景在服务器的路径 | 主页背景 |
| Bcountwords | Int | 是 | 默认 | 默认为0 | 总文字数 |
| Gcountliked | Int | 是 | 默认 | 默认为0 | 被喜欢数 |

1. 程序界面



（图：3.6.3.1 我的主界面）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 | public function getUser(){} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的用户账号从 view\_user 视图中检索相应的数据并返回给前端。 |

1. 模型类列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）修改个人信息

1. 模块描述

该部分又可分为四部分，分别为修改用户头像、用户昵称、个人简介以及登录密码。

1. 输入、输出

（1）修改用户头像

输入：用户账号、二进制图片数据。

输出：头像在服务器上的路径。

（2）修改用户昵称

输入：用户账号、用户昵称。

输出：修改用户昵称操作的响应代码和响应消息。

（3）修改个人简介

输入：用户账号、个人简介。

输出：修改个人简介操作的响应代码和响应消息。

（4）修改登录密码

输入：用户账号、原密码、新密码。

输出：修改登录密码操作的响应代码和响应消息。

1. 实现过程
2. 修改用户头像：后台收到前端发送的 base64 格式的图片，将其转码后写入到指定文件夹内，然后将该路径先更新到用户表中，最后将图片访问的路径返回给前端。
3. 修改用户昵称：后台收到前端发送的新用户昵称，根据用户账号修改用户表中的原有用户昵称。
4. 修改个人简介：后台收到前端发送的新个人简介，根据用户账号修改用户表中的原个人简介。
5. 修改登录密码：后台收到前端发送的用户账号和旧密码，先进行比对。比对正确后再将密码更改为新密码。
6. 程序界面



（图3.6.3.2编辑个人资料）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 | public function modifyUserHead() {} |
| 处理逻辑 | 1.通用安全性检测。  2.接收前端传递的图片数据，写入到服务器的指定文件夹内。  3.将写入图片的路径保存到用户表的用户头像列中。  4.将新的用户头像路径返回给前端。 |
| 方法设计 | public function modifyNickName() {} |
| 处理逻辑 | 1.通用安全性检测。  2.接收前端传递的用户昵称，更新用户表指定用户账号的昵称。 |
| 方法设计 | public function modifyUserBrief() {} |
| 处理逻辑 | 1.通用安全性检测。  2.接收前端传递的个人简介，更新用户表指定用户账号的简介。 |
| 方法设计 | public function modifyPassword() {} |
| 处理逻辑 | 1.通用安全性检测。  2.接收前端传递的用户账号和原密码，进行正确性比对。  3.比对正确后，更新用户表指定用户账号的密码；失败，则返回失败的原因。 |

1. 模型列表

|  |  |
| --- | --- |
| *Focus* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

三）获取个人主页信息

1. 模块描述

用户可以进入自己的个人主页，也可以进入到他人的个人主页，在功能上略微有些不同（个人主页：编辑个人资料；他人主页：关注+传音）在个人主页能够查看当前用户的个人信息，例如关注数、粉丝数等。

1. 输入、输出

输入：用户账号。

输出：用户头像、昵称、个人简介、关注数、粉丝数、总喜欢数、书写总文字数、是否已经关注（取决于个人主页的拥有者）、背景图。

1. 实现过程

根据前端传递的用户账号从 view\_user 视图中检索相应的用户信息，并返回给前端。

1. 程序界面



（图3.5.3.1个人主页界面）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 | public function getUserIndex() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的用户账号从 view\_user 视图中检索相应用户信息，并返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

四）获取动态信息

1. 模块描述

个人主页的动态栏目记录了当前用户最近的一些行为，例如喜欢了某某文章、关注了某某作者等。

2.输入、输出

输入：用户账号、已获取动态数。

输出：动态类型、链接、内容。

1. 实现过程

前端向后台提供用户账号，后台根据用户账号到 view\_action 视图中进行检索，获取该用户账号的动态，每次查询获取指定数量的动态（具体数量待定）。

1. 数据库设计

### 动态表 maction

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Eaid | Int | 否 | 主键 | 主键，唯一标识动态表 | 动态号 |
| Ukey | varchar(22) | 否 |  | 作者账号或文章号 | 链接 |
| Rvalue | varchar(32) | 否 |  | 作者名文章标题或专题名 | 内容 |
| Vaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |
| Htype | Int | 否 |  | 0:关注1:喜欢2:发表3:订阅4:评论 | 动态类型 |

1. 程序界面



（图3.5.3.2个人主页—动态）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *UserController* | |
| 方法设计 | public function getAction() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的“用户账号”和“已获取动态数”从 view\_action 视图中检索相应动态记录，并返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

五）获取文章信息

1. 模块描述

参考文章部分的“获取我的文章”。

六）黑名单管理

1. 模块描述

用户可以将黑名单列表中的用户从黑名单中移除。

1. 输入、输出

输入：用户账号、已获取黑名单用户数。

输出：黑名单用户头像、昵称。

1. 实现过程

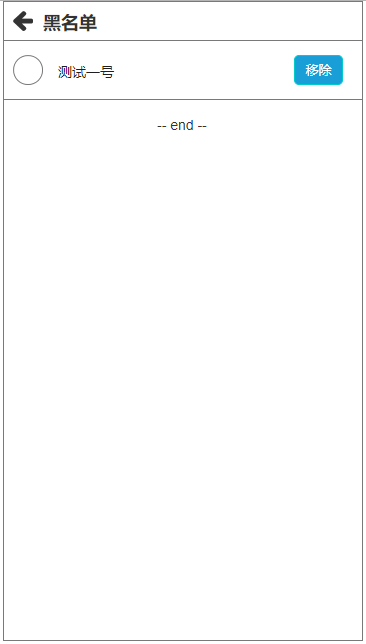
前端向后台提供用户账号，后台根据用户账号到 view\_blacklist 视图中进行检索，获取该用户账号的黑名单用户，每次查询获取指定数量的黑名单用户（具体数量待定）。

1. 数据库设计

### 黑名单表 nblacklist

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据格式 | 空 | 约束 | 说明 | 中文解释 |
| Tbid | Int | 否 | 主键 | 主键，唯一标识黑名单表 | 黑名单号 |
| Hblackaccount | varchar(12) | 否 |  | 加入黑名单的用户 | 黑名单账号 |
| Vaccount | varchar(12) | 否 | 外键 | 外键，用户表 | 用户账号 |

1. 程序界面



（图3.5.3.3黑名单）

1. 类设计
2. 控制器列表

|  |  |
| --- | --- |
| *BlacklistController* | |
| 方法设计 | public function getBlacklist() {} |
| 处理逻辑 | 1.通用安全性检测；  2.根据前端传递的“用户账号”和“已获取黑名单数”从 view\_blacklist 视图中检索相应黑名单用户，并返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *User* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

### **3.5.4 文章管理**

一．具体描述

该部分主要负责查阅各项记录，例如我的关注、我的文章等等内容。

二．具体实现

一）获取我的关注

1. 模块描述

用户点击“我的”主界面的“我的关注”，即可查看当前用户关注的用户列表。

1. 输入、输出

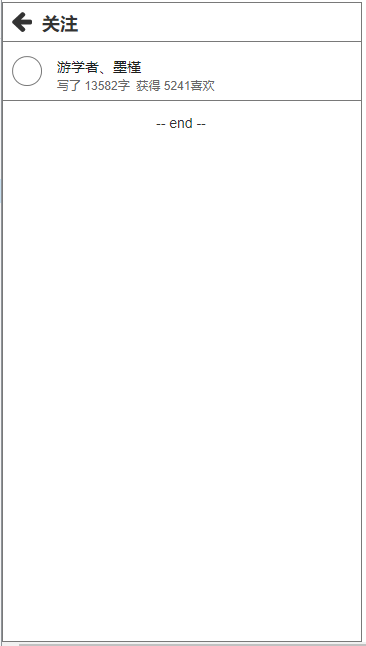
输入：用户账号、已获取关注数。

输出：作者头像、昵称、总文字数、被喜欢数、作者账号。

1. 实现过程

前端传递用户账号和已获取关注数给后台，后台根据这两个条件从 view\_userfocus 视图中检索，将检索到的数据返回给前端。

1. 程序界面



（图：3.5.4.1我的关注）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *FocusController* | |
| 方法设计 | public function getUserFocus() {} |
| 处理逻辑 | 1.通用性安全检测；  2.根据前端传递的两个参数从 view\_userfocus 视图中进行检索，并将检索结果返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

二）获取我的粉丝

1. 模块描述

用户点击“我的”主界面的“我的粉丝”，即可查看关注当前用户的用户列表。

1. 输入、输出

输入：用户账号、已获取粉丝数。

输出：作者头像、昵称、总文字数、被喜欢数、作者账号。

1. 实现过程

同“获取我的关注”。

1. 程序界面



（图：3.5.4.2我的粉丝）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *FocusController* | |
| 方法设计 | public function getUserFans() {} |
| 处理逻辑 | 1.通用性安全检测；  2.根据前端传递的两个参数从 view\_userfocus 视图中进行检索，并将检索结果返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

三）获取喜欢的文章

1. 模块描述

用户点击“我的”主界面的喜欢的文章”，即可查看当前用户喜欢的文章。

1. 输入、输出

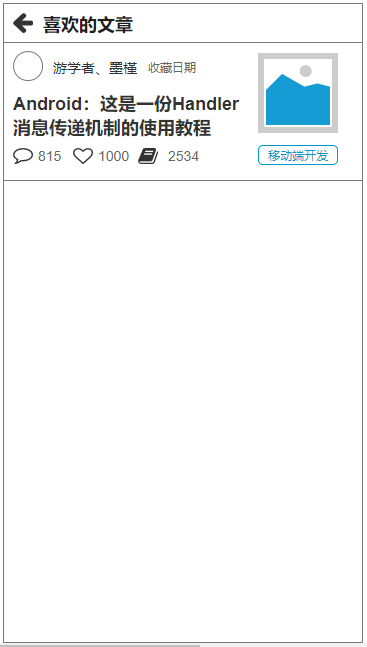
输入：用户账号、已获取文章数。

输出：作者头像、昵称、文章标题、评论数、喜欢数、阅读数、文章号、缩略图、专题名称、专题号、收藏日期。

1. 实现过程

同“获取我的关注”。

1. 程序界面



（图：3.5.4.3喜欢的文章）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getLikedArticle() {} |
| 处理逻辑 | 1.通用性安全检测；  2.根据前端传递的两个参数从 view\_userfocus 视图中进行检索，并将检索结果返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

四）获取我的文章

1. 模块描述

用户点击“我的”主界面的“我的文章”，即可查看当前用户所写的文章列表。

1. 输入、输出

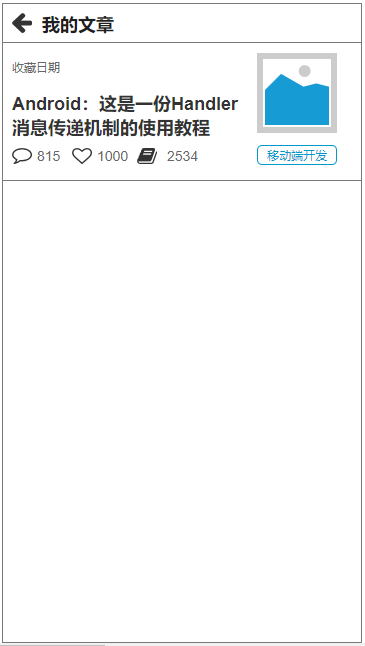
输入：用户账号、已获取文章数。

输出：文章标题、评论数、喜欢数、阅读数、文章号、缩略图、专题名称、专题号、文章日期。

1. 实现过程

同“获取我的关注”。

1. 程序界面



（图：3.5.4.4我的文章）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getUserArticle() {} |
| 处理逻辑 | 1.通用性安全检测；  2.根据前端传递的两个参数从 view\_article 视图中进行检索，并将检索结果返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |

五）获取历史记录

1. 模块描述

用户点击“我的”主界面的“历史记录”，即可查看当前用户的历史记录。

1. 输入、输出

输入：用户账号、已获取记录数。

输出：文章标题、评论数、喜欢数、阅读数、文章号、缩略图、专题名称、专题号、阅读日期。

1. 实现过程

同“获取我的关注”。

1. 程序界面



（图：3.5.4.5历史记录）

1. 类设计

A. 控制器列表

|  |  |
| --- | --- |
| *ArticleController* | |
| 方法设计 | public function getHistory() {} |
| 处理逻辑 | 1.通用性安全检测；  2.根据前端传递的两个参数从 view\_history 视图中进行检索，并将检索结果返回给前端。 |

1. 模型列表

|  |  |
| --- | --- |
| *Appointment* | |
| 方法设计 |  |
| 属性配置 |  |

1. 程序清单

| 文件路径 | 文件 | 文件说明 |
| --- | --- | --- |
|  |  |  |