**CUGBer 系统**

**设计说明书**

**目录**

[**1. 编写目的** 4](#_Toc71230687)

[**2. 定义** 4](#_Toc71230688)

[**2.1 技术选型** 4](#_Toc71230689)

[**2.2 规约** 5](#_Toc71230690)

[**3. 环境** 5](#_Toc71230691)

[**3.1 开发环境** 5](#_Toc71230692)

[**3.2 运行环境** 5](#_Toc71230693)

[**4. 系统构架设计** 5](#_Toc71230694)

[**4.1 数据构架设计** 5](#_Toc71230695)

[**4.2 开发构架设计** 6](#_Toc71230696)

[**5. 系统功能设计** 7](#_Toc71230697)

[**5.1 博客子模块功能设计** 7](#_Toc71230698)

[**5.1.1 查找博客** 8](#_Toc71230699)

[**5.1.2 创建博客与修改博客** 8](#_Toc71230700)

[**5.2 系统控制子模块功能设计** 9](#_Toc71230701)

[**5.2.1 人工控制** 9](#_Toc71230702)

[**5.2.2 系统监控** 11](#_Toc71230703)

[**5.2.3 自动化维护** 13](#_Toc71230704)

[**5.3 信息发布子模块功能设计** 13](#_Toc71230705)

[**5.3.1 委托** 14](#_Toc71230706)

[**5.3.2 失物招领** 15](#_Toc71230707)

[**5.3.3 文明吐槽** 15](#_Toc71230708)

[**5.3.4 结伴同行** 16](#_Toc71230709)

[**5.3.5 截止时间计时的实现** 16](#_Toc71230710)

[**5.4 用户信息子模块功能设计** 17](#_Toc71230711)

[**5.4.1 个人信息管理** 17](#_Toc71230712)

[**5.4.2 系统登录** 17](#_Toc71230713)

[**5.4.3 token解析器** 18](#_Toc71230714)

[**5.5 用户管理子模块功能设计** 19](#_Toc71230715)

[**5.5.1 增删改系统管理员** 19](#_Toc71230716)

[**5.5.2 管理员权限等级** 19](#_Toc71230717)

[**5.5.2 增删改普通用户** 20](#_Toc71230718)

[**5.5.4 普通用户权限等级** 20](#_Toc71230719)

[**5.6 业务管理子模块功能设计** 20](#_Toc71230720)

[**5.7 用户交互子模块功能设计** 20](#_Toc71230721)

[**6. 视图设计** 21](#_Toc71230722)

[**7. 数据库设计** 22](#_Toc71230723)

[**7.1 概念结构设计** 22](#_Toc71230724)

[**8.2 逻辑结构设计** 23](#_Toc71230725)

[**7.3 物理结构设计** 24](#_Toc71230726)

[**7.4 数据项** 25](#_Toc71230727)

[**7.4.1 系统通知表（system-message）** 25](#_Toc71230728)

[**7.4.2 系统概况表（system-information）** 26](#_Toc71230729)

[**7.4.3 反馈表（feedback）** 26](#_Toc71230730)

[**7.4.4 账户表（account）** 26](#_Toc71230731)

[**7.4.5 用户信息表（usr）** 27](#_Toc71230732)

[**7.4.6 委托表（quest）** 27](#_Toc71230733)

[**7.4.7 博客表（boke）** 28](#_Toc71230734)

[**7.4.8 评论表（comment）** 28](#_Toc71230735)

[**7.4.9 结伴同行表（together）** 29](#_Toc71230736)

[**7.4.10 失物招领表（lose）** 29](#_Toc71230737)

[**7.4.11 文明吐槽表（say）** 29](#_Toc71230738)

[**7.4.12 管理员表（admini）** 30](#_Toc71230739)

[**7.4.13 收藏表（fond）** 30](#_Toc71230740)

[**7.4.14 关注表（attention）** 30](#_Toc71230741)

[**7.4.15 管理员动作日志（log）** 30](#_Toc71230742)

[**8. 系统限制** 31](#_Toc71230743)

[**8.1 数量限制** 31](#_Toc71230744)

[**8.2 时间限制** 31](#_Toc71230745)

[**8.3 大小限制** 32](#_Toc71230746)

**1. 编写目的**

在完成了系统需求分析后，项目正式进入到系统设计阶段。面向对象设计与分析理论指出，系统设计应当为后续的开发提供基本思路，实际开发过程应当贴合最初的设计，并在适当情况下进行微调。

本说明书将在概念上对系统的整体设计进行说明，是需求分析的实现角度考量，原则上并不会对需求分析一味地照本宣科。此外，该模型只是设计模型，作为开发人员的参考，并不代表最终开发的结果会与该设计模型完全一致。本说明书面向的读者为开发人员、系统管理员以及系统运维人员。

**2. 定义**

**2.1 技术选型**

开发手段：

* 视图开发：html + css + JavaScript
* 应用开发：Java
* 缓存数据库：Redis
* 关系型数据库：MySQL

开发框架：

* 视图开发框架：Vuex
* 服务端综合框架：SpringBoot
* 服务端持久层框架：MyBatis
* 服务端视图交互框架：SpringMVC
* 服务端安全框架：SpringSecurity

开发工具：

* 代码版本管理工具：Git（中央仓库：www.gitee.com）
* 服务端项目管理工具：Maven（中央仓库：www.aliyun.com）
* 接口测试工具：Postman
* IDEA 代码编辑器
* WebStorm 代码编辑器

**2.2 规约**

规约约束了开发人员的开发习惯，提供了同一的命名、设计规范，有利于团队合作以及后续的运维开展。具体内容详见《CUGBer系统开发规约》。

**3. 环境**

**3.1 开发环境**

* Windows 10 操作系统
* Centos 7 操作系统
* docker 容器中间件
* MySQL 7.6 关系型数据库
* Redis 6 缓存数据库
* ElasticSearch 7 文档索引数据库
* JDK 1.8 开发者工具

**3.2 运行环境**

* Centos 7 操作系统
* docker 容器中间件
* MySQL 7.6 关系型数据库
* Redis 6 缓存数据库
* ElasticSearch 7 搜索数据库
* Java 8 运行时环境

**4. 系统构架设计**

**4.1 数据构架设计**

引入缓存数据库和搜索中间件，对数据进行以访问频率为依据的分区存储，并用于处理文档检索工作。对于大多数数据而言，需要在关系型数据库中进行持久化，在合适的时候再存入到缓存数据库或搜索中间件。这种架构设计可以有效提高系统的吞吐量，改进系统响应效率。如图2所示

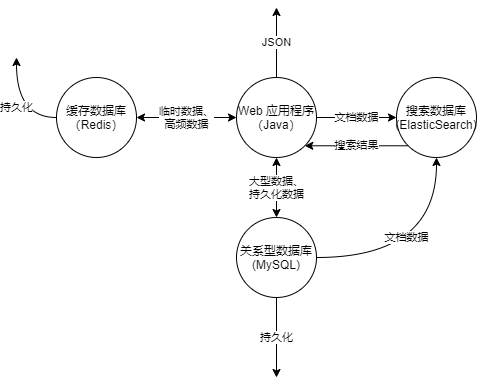


图 1 数据构架设计图

**4.2 开发构架设计**

采用前后端分离的开发模式，旨在将视图和应用程序的开发分离开来，从而有效提高开发效率。在这种开发模式下，服务端可以视为一个数据接口的提供者，而视图等静态资源都存放在前端开发服务器上，外界通过访问前端开发服务器来访问开发网页。

图示

描述已自动生成

图 2 前后端分离开发构架

如图1 所示。在开发阶段的内测客户端需要向两个不同的服务器进行请求，当需要访问页面或进行站内路由时，会向前端服务器发送请求，由前端服务器提供页面资源。而当需要一些用于渲染静态页面的动态数据时，内测客户端又会请求另一个后端服务器，该服务器后隐藏了多个数据源，并通过接口为客户端提供动态数据。客户端的浏览器在接收到动态数据后会自行在客户端进行数据渲染。

图中为了简化构图只显示了一个接口，实际上后端服务器可以提供很多接口用于进行不同的响应。这些接口会以文档的形式展示给前端开发人员，在形式上形同于将这些接口都暴露给了前端服务器。

**5. 系统功能设计**

**5.1 博客子模块功能设计**

博客子模块中包含了增加文章、删除文章、更改文章、发布文章、阅读文章等15个用例，考虑到这些用例之间存在着先后顺序，本说明书给出博客子模块的生命周期，如图3所示。

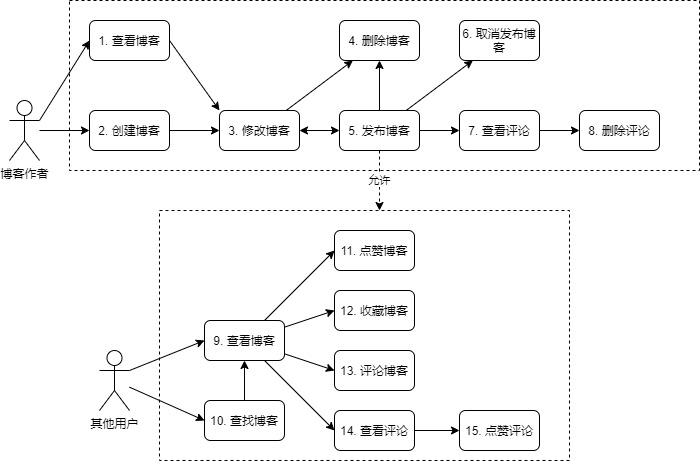


图 3 博客子模块生命周期

上述用例中，多数用例的实现并不复杂，具备一定的模式，具体的参数会在后续的《接口描述说明书》中进行叙述。在此，仅对其中个别有着较为复杂过程或非常规模式的用例提供具体设计。

**5.1.1 查找博客**

对于用户而言，博客系统中较为重要的功能就是查找博客。在这个过程中，用户往往通过输入一串关键字，查找到系统中包含该关键字的博客。该过程意味着进行全文检索，如果使用数据库中的模糊匹配，效率极为低下。因此需要使用到基于 ElasticSearch 的分词检索。

如图4所示为博客查找的时序图。一次检索过程分为两个部分，第一个部分是在数据库中检索，该部分的检索内容为长度短小的标题、简介、关键字等。第二个部分是在 ElasticSearch 中进行检索。该部分会对系统中所有文章进行全文检索，返回全文检索的内容，最后由 Service 层汇总整合为 DTO 返回到 Controller 层，以 JSON 格式传输到 web 视图供用户查看。

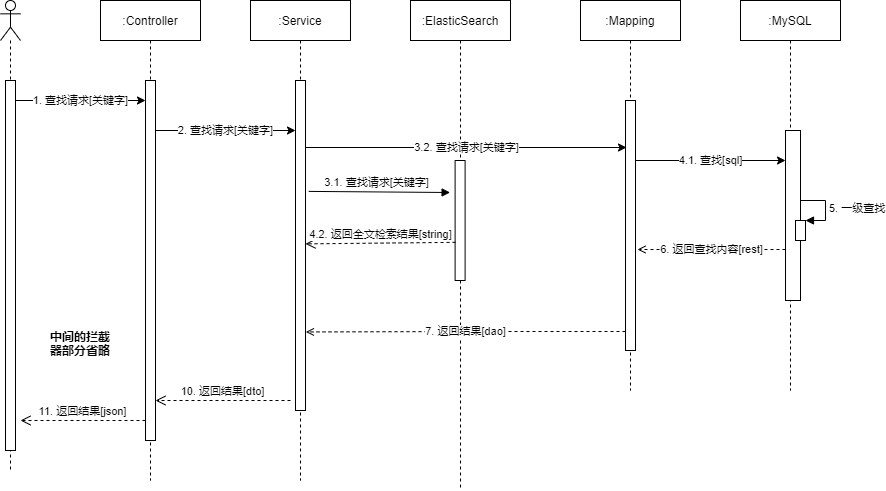


图 4 查找博客时序图

**5.1.2 创建博客与修改博客**

创建博客与修改博客的本质都是对数据源进行某种形式的改变。由于加入了全文搜索中间件，创建或修改一篇文章需要同时修改MySQL和ElasticSearch数据源。

必须要强调的是，需要在ES和MySQL中同时保存数据的做法看似会形成大量冗余数据，但实际上，因为MySQL的数据结构和ES中不一致，但是用户查找时的业务场景不同，因此，这种冗余手段可以节省大量的数据同步开销。

此外，由于ES中数据为文档形式和关系型数据库的数据结构大相径庭，因此，类似于文章id的字段，只能通过字符串拼接的方式存入。

用户修改或添加博客的时序图如图5所示。

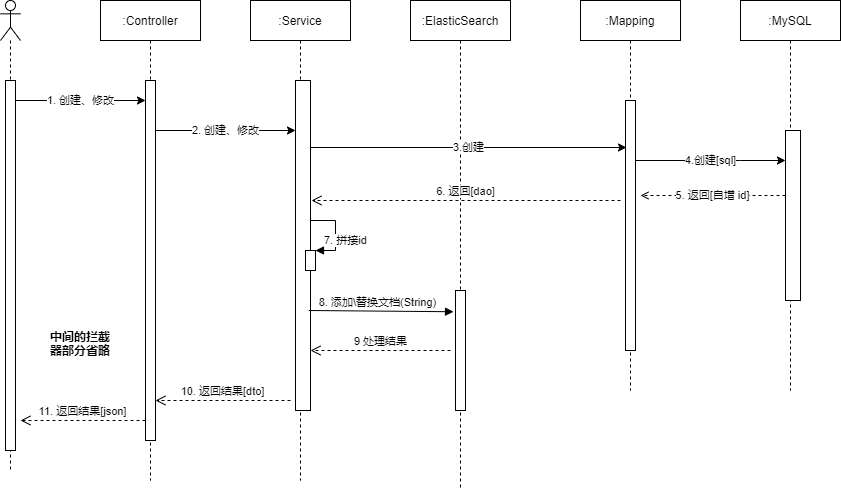


图 5 博客创建-修改时序图

**5.2 系统控制子模块功能设计**

由于在系统需求分析过程中分析的对象面向用户（管理员），因此在用户（管理员）看来，系统控制子模块系统只是用于提供系统概况查看、系统通知等系统控制功能。但实际上，对于整个系统而言，系统控制子模块除了需要管理员参与外，还有自动化部分，即实时统计系统概况、生成系统概况报告、定期持久化系统日志、管理员行为监控、发放系统通知等。

**5.2.1 人工控制**

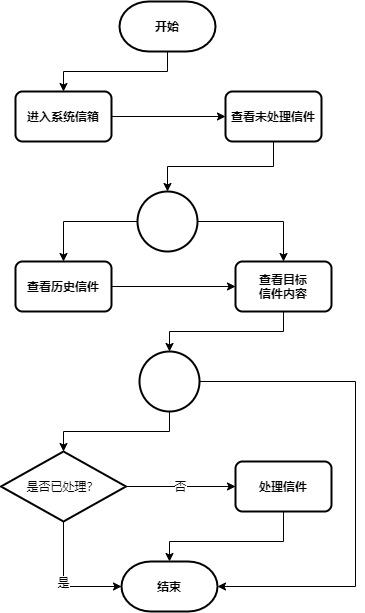
* **功能禁用：**

系统通过拦截器实现部分系统功能的禁用。具有最高权限的系统管理员可以通过系统控制中的“系统禁用”功能选择特定的功能模块进行禁用。一旦功能被禁用，所有用户都无法继续使用系统中该部分的功能。

这往往会导致一系列问题，例如，用户正在书写博客，此时，博客的功能被禁用，那么用户依旧可以在页面内继续书写，但是任何一个有关博客功能的请求都无法得到响应，比如保存。这样的操作可能会引起用户的强烈不满。

由于该功能是一个具有较大风险的功能，因此每次进行**功能禁用**时都**必须要求管理员重新输入密码才可执行。**

* **系统信箱：**



图表 6 系统信箱流程图

如图6是系统信箱功能流程图，当用户产生反馈时，会在系统内以“信件”的方式存在。当有管理员进入到系统信箱进行查看时，系统会自动检索所有未处理的“信件”，以时间排序进行展示，并显示其时间、反馈者、反馈内容等信息。

管理员可以选择某一条未处理的反馈信息，在阅读后可以点击“已处理”按钮，并**填入处理方式**，将该信件标记为已处理。

对于已处理的信件，管理员可以通过“查看历史反馈”功能进行查看，以时间顺序进行显示。历史反馈会显示该信件的处理者、处理方式和处理时间。

* **系统通知：**

拥有中等权限的管理员可以发布系统通知。

系统通知是一条具有时效性的、会在用户登录时自动推送的信息。当管理员需要发布一条系统通知时，需要按要求填写通知内容、有效时间。管理员可以进行如下动作：

a) **发布系统通知**。

b) **查看系统通知**。系统会优先显示正在生效的通知。

c) **撤回、修改通知**。需要注意的是，修改并不会改变其原有的生效时间。

* **管理日志：**

管理日志记载了所有管理员的管理动作，任何管理员无权直接进行任何形式的修改（最高权限的管理员可以通过设置自动维护进程来间接删除日志）。该日志是为了能及时检查出管理员的违规操作，并进行适当处理，以免造成不可挽回的后果。

* **概况查看：**

该功能旨在将大量自动化形成的数据通过 web 视图进行可视化，供管理员查看。由于数据分为实时数据和持久化数据，因此，查找的数据源也不尽相同。对于管理员而言，应当优先提供系统实时概况的显示，同时提供接口查看历史概况，如果技术成熟，可以提供数据分析图表。

**5.2.2 系统监控**

系统监控是系统控制子模块的核心功能。出于对安全、统计、规约、性能等诸多方面的考虑，系统需要有一个能以适当程度监控系统中用户和管理员的自动化子系统。为了提高用户的体验，该子系统应当以无侵入的形式加入到系统中，即用户和管理员在使用的过程中完全感受不到该系统的存在。

针对不同的用户，系统需要监控的方面不同，监控的程度也不同。

一、对于用户而言，系统需要监控其登录状态、访问状态，记录其访问数据。

a) **登录状态监控**：用户在登录时需要给该用户一个标志位，表示该用户已登录，当用户退出登录时需要撤回该状态位，以便系统管理员能查看到该用户的登录状态，并能够强制下线该用户。

b) **访问状态监控**：当用户发出一个请求时，需要根据用户的登录状态、权限状态和系统的维护状态综合决定是否允许用户访问。以下四种情况的任何一种出现，系统将会拒绝用户访问：

1. 用户没有登录。

2. 系统没有记录用户的登录状态位

3. 用户没有访问该请求的权限

4. 用户访问的系统功能已经被禁用

c) **记录访问数据**：当用户访问某个URL时，系统需要对不同功能模块分别统计其访问次数。访问数据包括了，系统日登录次数、各功能模块访问次数等等。

二、对于管理员而言，系统需要对每一个管理员的每一个动作进行记录，形成日志。

考虑到上述功能的无侵入性，系统应当采用拦截器（Interceptor）方式来实现该功能。当任何一个请求发往Web应用服务器时，在该请求得到处理之前，必须先经过多层拦截器链。如图7所示。

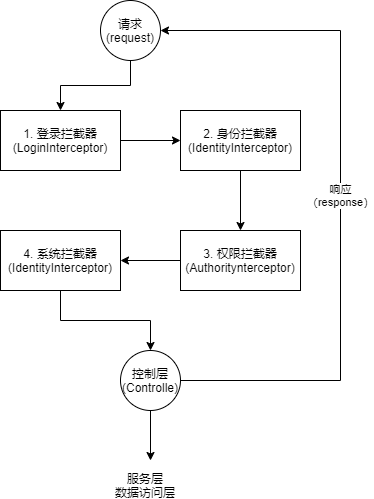


图 7 系统控制子模块拦截器链

拦截器的具体信息如表所示：

表 1 拦截器描述

|  |  |
| --- | --- |
| **拦截器名** | **功能** |
| 登录拦截器，LoginInterceptor | 检查用户登录令牌（token），判断是否有权访问 |
| 身份拦截器，IdentityInterceptor | 检查用户身份状态位，判断是否有权访问 |
| 权限拦截器，AuthorityInterceptor | 检查用户权限状态位，判断否有权访问 |
| 系统拦截器，SystemInterceptor | 检查系统功能状态位，判断是否有权访问 |

注：用户身份状态位是一个key-value对，存储在缓存数据库中，有两种状态位，如果是1，表示该用户以普通用户身份登录，如果是2，表示该用户以管理员身份登录。例如，当ID为1的用户登录时，就会在缓存数据库中添加u1:1，这一条键值对。

**5.2.3 自动化维护**

系统概况是衡量系统的重要数据，系统有必要定时记录下系统概况信息。该过程实际上是将缓存数据库中的数据持久化到关系型数据库。

此外，系统的信息有其时效性，例如：反馈信息、管理员行为日志等数量庞大但有效期较短的信息，需要定期进行清除。

由于这些数据都已经在关系型数据库中被持久化，无法像在缓存数据库中设置自动失效时间，因此，系统需要开启多个守护线程用于进行自动化维护。这些进程每隔一定时间被唤醒一次，用于进行系统维护，其他时间处于休眠状态，并不占用服务器性能。

需要注意的是，每一次进行自动化维护都意味着对大量数据进行修改、删除，这意味着每一次修改都需要封锁住数据库表内大部分数据，因此会导致系统陷入周期性卡顿。因此，自动化维护的时间应当与用户错峰执行.

具有最高权限的管理员可以设置自动化维护进程是否开启、维护时间（多少时间之后开始自动化维护）、清除周期（需要清除的数据是多少天前的）。

自动化维护进程条目如表2所示。

表 2 自动化维护进程表

|  |  |
| --- | --- |
| **进程名** | **进程功能** |
| AutoFeedbackCleaner | 自动清除“用户反馈”信息 |
| AutoAdminBehaviorCleaner | 自动清除“管理员行为日志”信息 |
| AutoLostAndFoundCleaner | 自动清除“失物招领”信息 |
| AutoGoTogetherCleaner | 自动清除“结伴同行”信息 |
| AutoQuestCleaner | 自动清除“委托”信息 |

**5.3 信息发布子模块功能设计**

信息发布子模块是系统的核心业务模块。该模块为用户提供多种多样的信息发布形式，因此也是最为复杂的模块。

为了更好的服务于用户，信息发布子模块提供了委托、“失物招领”、“文明吐槽”、“结伴同行”四种类型的信息发布方式。用户可以根据需要发布信息的不同性质自行选择。下文将详细讲述每个方式的具体流程。

需要注意，限于系统性能考虑，所有的信息发布均由次数限制，限制每日12点刷新。

**5.3.1 委托**

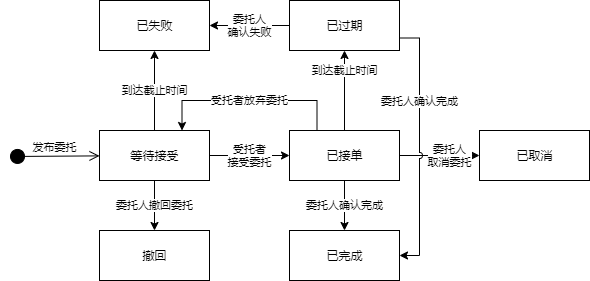


图 8 委托状态图

用户可以通过发送委托请求其它用户的帮助，并给予一定报酬。同时，用户可以接受其它用户的委托，并在规定时间内，在规定地点完成委托交接，从而成功完成委托。

图8为委托的状态图。下述为每个状态的解释。

a) **等待接受：**委托人成功发布委托后，委托即进入“等待接受”状态，处于该状态下的委托会展示在公共专栏下，供所有用户查看。

b) **已接单：**受托者接受委托后，委托状态转为“已接单”，此时该委托将不会显示在公共专栏上。如果受托人接单后主动放弃委托，那么状态又会自动转化为“等待接受”状态。受托人一旦主动放弃委托，系统将会扣除其委托信誉。

c) **已过期：**如果委托已经被接收，那么到达委托的截止时间后，委托将自动转化为“已过期”状态。到达该状态并不意味着受托人没有完成委托，也有可能是已经完成了委托，但是委托人忘记确认了。因此如果委托人一直没有确认，那么委托的状态将一直保持在“已过期”状态，处于该状态的委托无法被删除、撤回、取消。

d) **已完成：**在委托状态变为“已接单”后，委托人可以随时确认，将其状态变为“已完成”，委托交接成功，系统将会自动进行下一步数据处理。处在“已完成”状态下的委托，可以被委托人删除。

e) **已取消**：当委托状态变为“已接单”后，委托人依旧可以取消委托，此时，委托状态变为“已取消”。处于该状态的委托会自动被系统删除，同时，委托人的委托信誉会相应进行扣除。

f) **已失败：**委托转变为“已失败”状态有两种可能，一种是委托已经被接受，超过截止时间后委托人确认其失败，这种情况下，会扣除受托人的委托信誉。另一种是委托没有被接受，但是也到达了截止时间，那么系统会自动判定其“已失败”。处在该状态的委托可以被委托人主动删除。

g) **撤回：**委托处在“等待接受”时，委托人可以随时对其进行撤回，此时，并不会对委托人的信誉造成影响，但是还是会扣除每日委托发布上限。

由于委托有其时效性，为了实现这种动态的状态变化，必须对状态码进行区分。状态码主要根据其特性和存储区域进行区分，具体实现如表2所示：

表 3 状态描述表

|  |  |  |  |
| --- | --- | --- | --- |
| **状态** | **状态码** | **特性** | **存储区域** |
| 等待接受 | 001 | 限时状态 | 缓存 |
| 已接单 | 010 | 限时状态 | 缓存 |
| 已过期 | 011 | 不限时状态 | 磁盘 |
| 已完成 | 100 | 不限时状态 | 磁盘 |
| 已失败 | 101 | 不限时状态 | 磁盘 |
| 已取消 | 无 | 临时状态 | 即刻销毁 |
| 撤回 | 无 | 临时状态 | 即刻销毁 |

开发人员需要根据状态描述表中的描述相应的进行功能实现。

**5.3.2 失物招领**

“失物招领”是另一种类型的信息。考虑到其独特性，将其从委托类型的信息中分离出来。与委托不同，“失物招领”状态有如下几个特点：

a. **可匿名**。用户发布失物招领时可以选择匿名发布。

b. **高限制**。用户发布的失物招领对信息格式上有严格限制，例如：必须填写失物名称且不得超过10个字、失物领取地点不得超过15个字。一来可以便于其他用户查看；二来可以有效限制用户通过匿名在“失物招领”专区发表不当言论。

c. **自动失效**。由于“失物招领”同样具有时效性，因此，如果一条“失物招领”信息一直存在以上，那么系统将自动判定其已“失效”，随后自动删除。该功能由系统控制子模块中自动化维护功能实现。

**5.3.3 文明吐槽**

文明吐槽允许用户自由的发表言论。

基于网络言论安全，本系统会对敏感词汇、不文明词汇进行屏蔽。此外，用户必须对自己的言论负责，因此不提供匿名功能。

为了防止用户在评论区发生争吵，本系统没有提供对文明吐槽信息的追加评论功能，二使用点赞/点踩功能来让用户表明自己对该条“文明吐槽”的赞同度。

系统会以时间和点赞数综合进行优先展示。

**5.3.4 结伴同行**

类似于失物招领，不同是必填词条变为了：结伴事件、集合地点、联系方式、截止时间。可匿名。到达截止时间后，系统会自动清理这条信息。

**5.3.5 截止时间计时的实现**

信息发布子模块中有很多具有截止时间的信息，在到达截止时间后，系统需要对这些信息执行相应操作。根据现有的技术选型，系统采用如图9的方式进行实现

由图9可见，一个计时系统需要通过另一个独立的守护线程来动态的维护缓存数据库中的有效集。

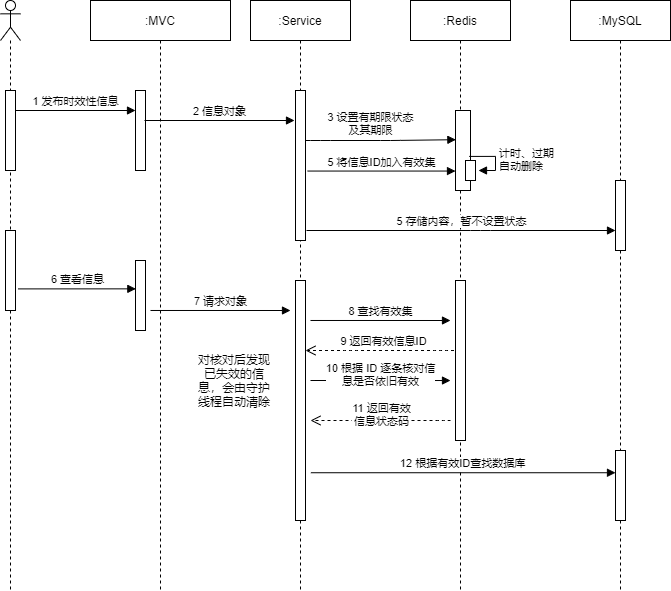


图 9 计时系统时序图

**5.4 用户信息子模块功能设计**

用户信息子模块负责管理用户的个人信息以及用户登录。

**5.4.1 个人信息管理**

对于用户个人信息而言，用户可以对其进行增删改查，由于该部分实现仅仅和关系型数据库挂钩，因此不再进行赘述。

**5.4.2 系统登录**

用户登录的具体流程如图10所示。

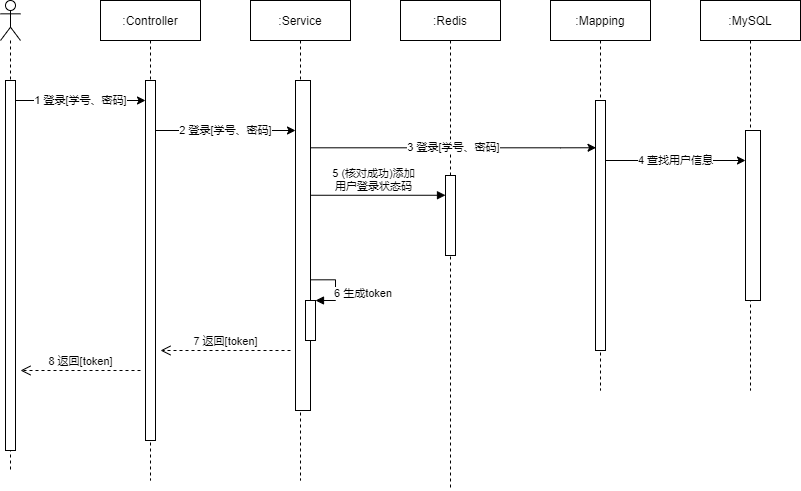


图 10 用户登录时序图

可见，对于用户的登录功能，其实现较为复杂，用户的登录包括了令牌生成、登录状态生成等过程。由于前后端分离的开发模式，使得传统的 Session 会话无法使用，因此本系统只得采用JWT（JsonWithToken）技术实现http的状态保存。

该技术指出，当客户端与服务器通过json传输数据时，需要携带一个令牌（token）该令牌中存放了用户的相关数据。该令牌虽然通过Base64编码为了不可读形式，但并没有改变其明文本质，因此，需要将其进行人为的加密。

其中，由于生成的 token 如果不进行加密，那么将会导致用户信息泄露，从而给黑客冒名顶替甚至窃取账号的风险，因此，生成token时需要考虑到用户信息的保护。系统生成token的方式如图11所示。

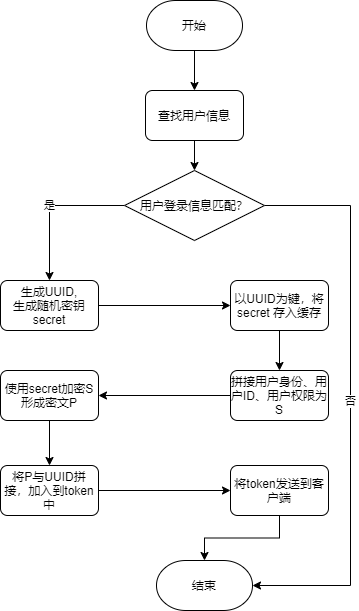


图 11 token加密流程图

当服务器接收到token时，按照下述步骤获取到用户的权限等重要个人信息：

1. 使用Base64逆编码token的有效荷载部分

2. 根据固定格式，获取到以明文形式出现的UUID和以密文形式出现的账户重要信息S

3. 使用用户ID从缓存数据库中获取到解密密钥secret，使用密钥解密S，获取到用户身份、用户ID、用户权限等重要信息。

4. 利用这些重要信息进行进一步操作。

使用这种方式可以杜绝入侵者通过随意UUID从而获取到较高权限的可能，因为权限部分的密文是无法被伪造的。但是却无法防治用户token泄露，黑客利用泄露的token进行重放攻击。

为此，用户的token需要设置有效期，例如两个小时，从而降低用户遭受到重放攻击的可能。

**5.4.3 token解析器**

考虑到系统控制子模块的拦截器需要使用到用户身份、用户权限、用户ID等用户信息，而这些信息都存放在token中，无法直接使用，因此必须**在系统控制拦截器链之前**加入token解析拦截器，以保证其他拦截器可以使用到这些存放在token中的数据。

**5.5 用户管理子模块功能设计**

用户管理子模块负责提供对所有用户进行管理的功能，包括了管理员和普通用户。

最高管理员在整个系统中**有且只有一个**，由系统所属机构持有。最高管理员可以添加、修改、删除任何用户或管理员的信息，限制、授予任何用户或管理员的权限。

**5.5.1 增删改系统管理员**

最高管理员可以**添加系统管理员**。由于该操作对系统风险性较大，因此，每进行一次都必须重新输入管理口令。在口令正确后，管理员通过填写真实姓名、登录令牌、登录密码、权限等级即可添加一个新的管理员。需要注意的是，为了保证系统安全，新添加的管理员不会立即生效，而会在一天（默认）后才生效，在这期间，最高管理员可以随时撤销该管理员的登记。通过这种方式登记的管理员，其权限等级不得等于最高管理员。最高管理员可以在后续修改这些管理员的权限，或直接删除该管理员。

**5.5.2 管理员权限等级**

系统的管理员权限等级如表4所示。

表 4 系统权限等级

|  |  |  |  |
| --- | --- | --- | --- |
|  | 最高权限 | 高权限 | 低权限 |
| 状态码 | 100 | 010 | 001 |
| 增删改管理员 | √ | × | × |
| 禁用系统功能 | √ | × | × |
| 修改系统维护参数 | √ | × | × |
| 增删改用户 | √ | √ | × |
| 发送系统通知 | √ | √ | × |
| 修改业务信息 | √ | √ | √ |
| 处理用户反馈 | √ | √ | √ |
| 查看各类管理信息 | √ | √ | √ |

**5.5.2 增删改普通用户**

持有高权限等级以上的管理员可以登记普通用户。该过程只需要填写用户学号和初始密码即可完成。持有高权限等级以上的管理员可以修改普通用户的权限等级、准入状态、委托信誉等信息。除此之外，还可以直接删除用户。

用户登记日志作为重要信息会从管理员行为日志中分离出来，作为单独一栏供所有管理员查看。

**5.5.4 普通用户权限等级**

表 5 普通用户权限等级

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 未受限 | 低受限 | 中受限 | 高受限 |
| 状态码 | 0001 | 0010 | 0100 | 1000 |
| 发表评论 | √ | × | × | × |
| 发表文明吐槽 | √ | × | × | × |
| 发表委托 | √ | √ | × | × |
| 发表博客 | √ | √ | × | × |
| 发表结伴同行 | √ | √ | √ | × |
| 发表失物招领 | √ | √ | √ | × |

**5.6 业务管理子模块功能设计**

业务管理子模块为管理员提供各类业务的信息管理。这包括了：博客管理、信息管理。

**5.7 用户交互子模块功能设计**

用户交互模块提供给用户相互之间进行交互的功能，如：关注、点赞等。由于点赞功能的特殊性，下面将详细讲述点赞功能的实现。

本系统中一个用户可以给同一个用户或文章点多次赞，为了防止不良用户恶意点赞，导致系统遭受洪泛攻击，每一个用户每天的点赞次数都有上限。因此，当用户点赞时，会在缓存数据库中加入一个点赞次数标志位。用户每点一次赞，就将这个点赞标志位扣减一次，如果标志位扣减为零，则用户将无法继续点赞。

为了保护数据库，当用户点赞后，系统并不会立刻将点赞数更新到数据库中，而是先暂存在缓存数据库中。每隔1小时，系统的守护线程会自动更新这些点赞数据到数据库中。具体的点赞流程如图12所示。

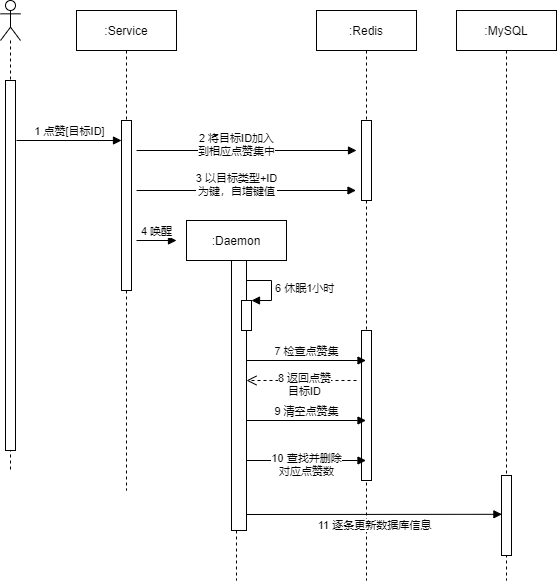


图 12 点赞时序图

显然，这种设计会导致用户实际操作与数据库数据不一致的情况。因此，在用户查看点赞数时，会以从数据库中查找的数据作为基准值，随后，在缓存数据库中查找到尚未更新到数据库中的值，累加后再返回给用户。

**6. 视图设计**

Vue 开发框架的宗旨是将原本繁琐而重复度高的前端视图设计过程通过模块化的思想进行简化。组件化便是其实现模块化的一个具体手段。该模式指出：对于同一个网站中的诸多页面之间，有很多可以复用的部分，可以将这些可以复用的部分抽离出来，作为一个组件，而一个页面实际上就是多个组件组合的结果。

既然一个页面实际上是由多个组件构成，那么前端路由就可以转化为单纯的组件替换，而不用重新渲染整个页面。这在很大程度上改进了Web视图的加载性能，提高了用户的使用体验。

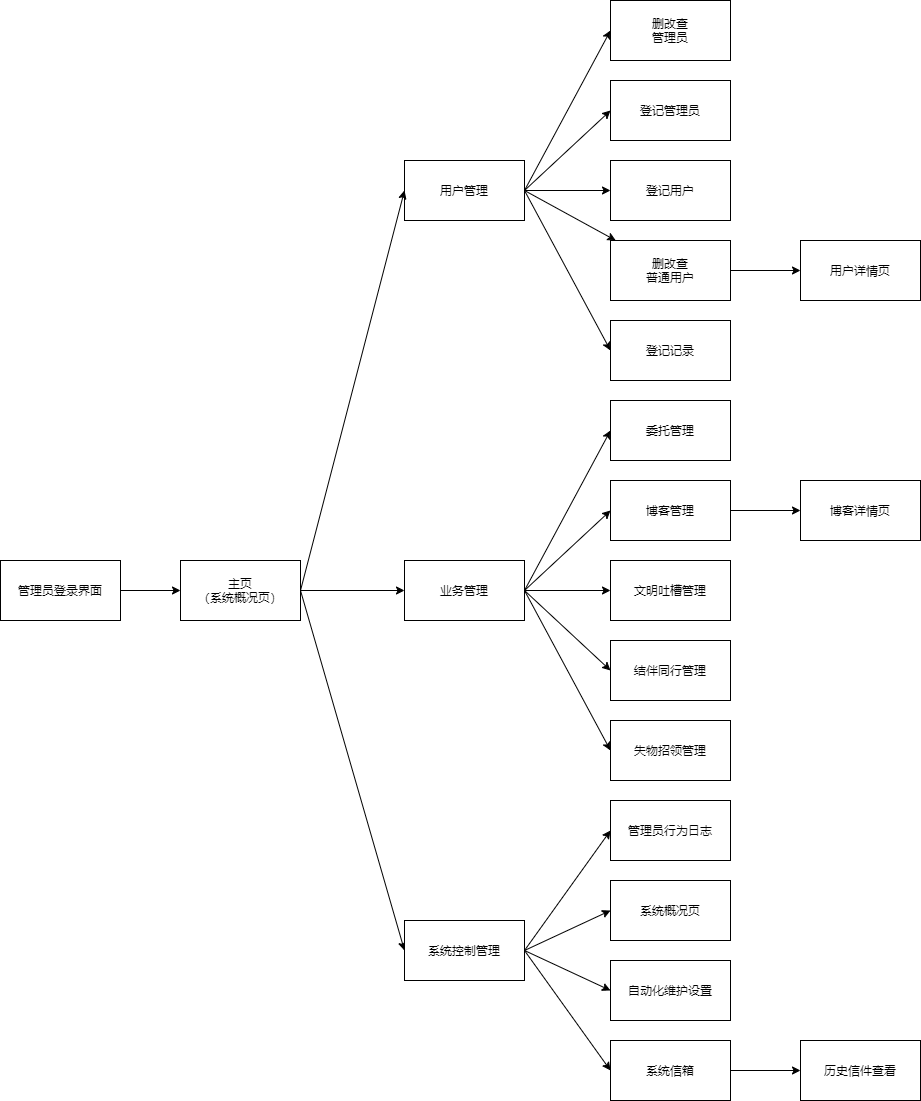


图 13 管理系统页面逻辑

**7. 数据库设计**

**7.1 概念结构设计**

概念结构设计主要反映的是项目中各个实体之间的基本关系，如图13所示。本文中所使用的概念结构设计使用的是 ER 表示法。考虑到 ER 图中可以显示出各个实体的属性，但是由于这样会导致图形结构变得复杂，而失去原本希望展示出来的基本实体关系，本图中没有给出每个表的属性。

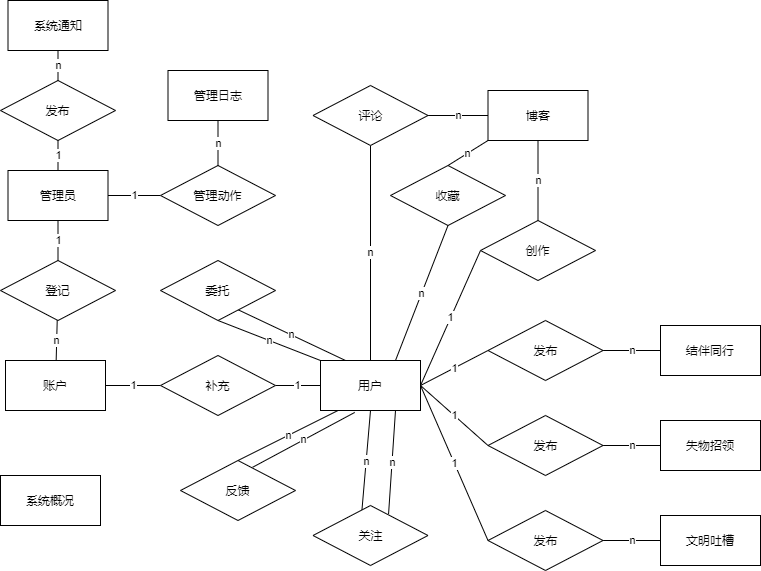


图 14 概念结构设计

**8.2 逻辑结构设计**

根据数据库概念设计，即 ER 图，可以在其中抽象出具体的表结构，在这里，使用箭头表示各个表之间的依赖关系。

由于该系统并没有存在过分追求性能的需求，以至于要进行反范式化设计，因此，在此处采用的是遵循基本的第三范式进行表设计，由 ER 图转化为满足第三范式的表结构具有固定的方法论，因此所得结果如图14所示。

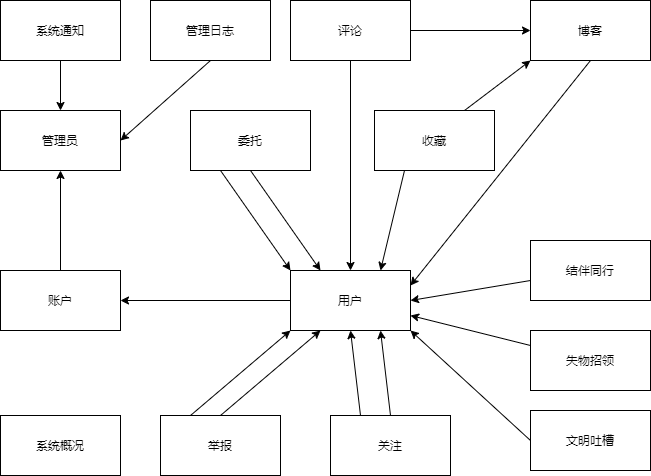


图 15 数据库逻辑设计

**7.3 物理结构设计**

根据数据库逻辑结构设计，对各表进行属性、属性类型选择、索引结构的设计是物理结构设计希望达到的目的。

物理结构设计往往需要结合具体的数据库产品进行，尤其是对属性类型的选择会直接影响到后续的开发过程，因此需要特别谨慎。得到的物理结构设计如图15所示。

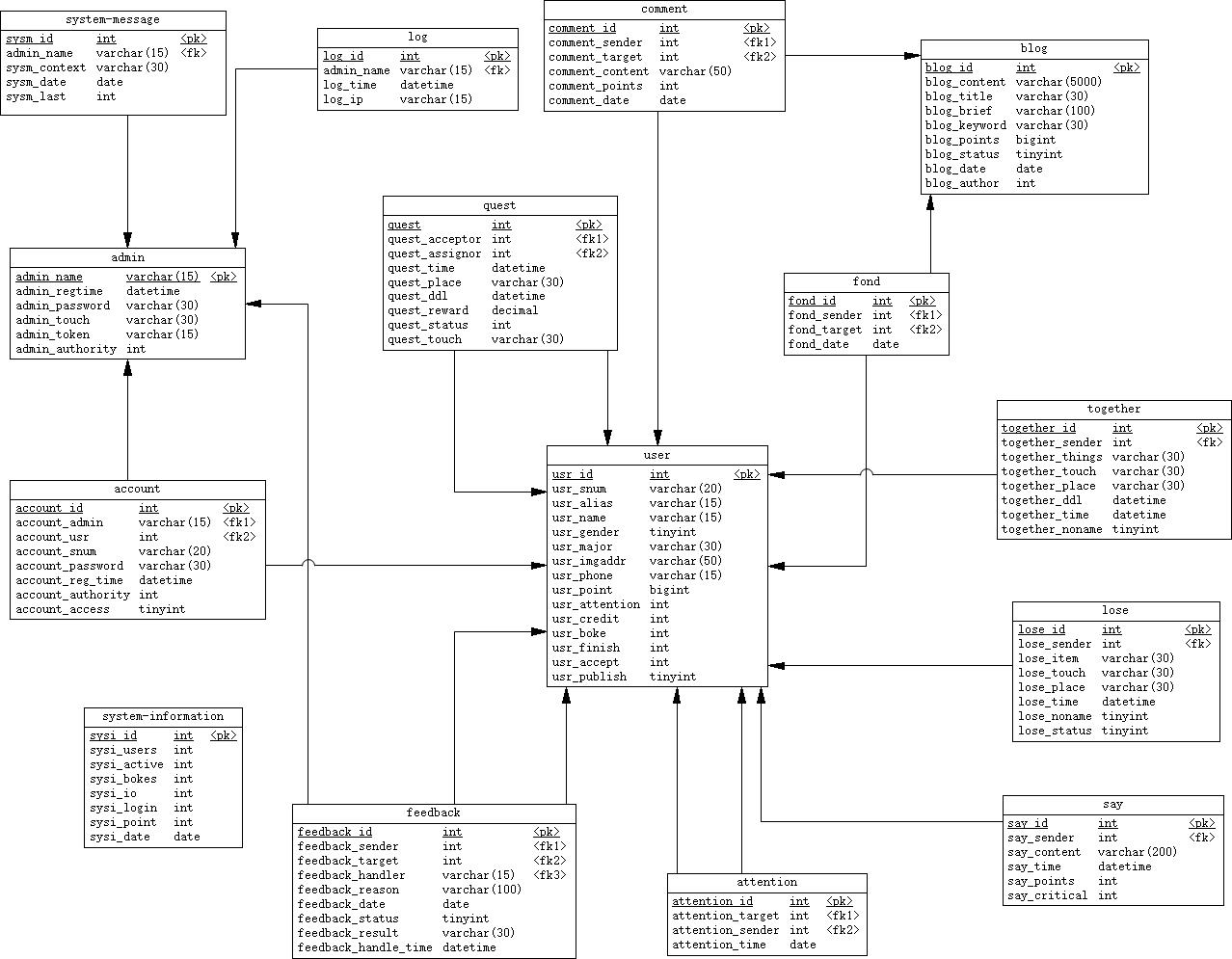


图 16 物理结构设计

其次是索引设计，设置恰当的索引可以有效的提高数据库查询、排序的速度。但是对索引的设计应当先预估系统所能够产生的数据量，只有在确定数据最终会达到中等规模数据量及其以上后才应当设计索引。同时，一定要意识到索引的加入会在一定程度上降低数据插入与删除的性能，只有进行了综合考量后才能决定是否添加索引。否则，索引所带来的查询效率的提升并不能弥补其所消耗的空间所带来的负面效果。

**7.4 数据项**

数据项的设定参照数据库物理结构设计和先前的需求分析工程，在需求分析的结构化分析中已经给出完善的数据流条目和数据项条目描述。此处，将依据数据库选型进行落实。本说明书中的数据项以表为单位进行描述，包括了：数据项名、说明、数据库类型、索引。

**7.4.1 系统通知表（system-message）**

表 6 系统通知表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| sysm\_id | 自增主键 | INT | 主键索引 |
| sysm\_context | 通知内容 | VARCHAR(300) | 无 |
| sysm\_date | 通知时间 | DATE | 辅助索引 |
| sysm\_sender | 通知发布者姓名 | VARCHAR(15) | 无 |
| sysm\_last | 持续时间（天） | INT | 无 |

**7.4.2 系统概况表（system-information）**

表 7 系统概况表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| sysi\_id | 自增主键 | INT | 主键索引 |
| sysi\_users | 系统总用户量 | INT | 无 |
| sysi\_active | 系统日活量 | INT | 无 |
| sysi\_bokes | 系统总博客量 | INT | 无 |
| sysi\_io | 系统信息吞吐量 | INT | 无 |
| sysi\_login | 当日登录量 | INT | 无 |
| sysi\_point | 当日浏览量 | BIGINT | 无 |
| sysi\_date | 日期 | DATE | 辅助索引 |

**7.4.3 反馈表（feedback）**

表 8 反馈表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| feedback\_id | 自增主键 | INT | 主键索引 |
| feedback\_sender | 反馈发出者ID | INT | 无 |
| feedback\_target | 举报目标ID | INT | 无 |
| feedback\_reason | 反馈内容 | VARCHAR(100) | 无 |
| feedback\_date | 反馈日期 | DATE | 辅助索引 |
| feedback\_status | 状态 | TINYINT | 无 |
| feedback\_handler | 处理者姓名 | VARCHAR(15) | 无 |
| feedback\_result | 处理结果 | VARCHAR(30) | 无 |
| feedback\_handle\_time | 处理时间 | DATETIME | 无 |

**7.4.4 账户表（account）**

表 9 账户表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| account\_id | 自增主键 | INT | 主键索引 |
| account\_snum | 学号 | VARCHAR(20) | 复合索引-A-1 |
| account\_password | 密码 | VARCHAR(30) | 复合索引-A-2 |
| account\_usr | 对应用户 | INT | 无 |
| account\_reg\_time | 登记时间 | DATETIME | 无 |
| account\_admin | 登记者姓名 | VARCHAR(15) | 无 |
| account\_authority | 权限 | INT | 无 |
| account\_access | 准入 | TINYINT | 无 |

**7.4.5 用户信息表（usr）**

表 10 用户信息表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| usr\_id | 自增主键 | INT | 主键索引 |
| usr\_snum | 学号 | VARCHAR(20) | 唯一索引 |
| usr\_alias | 昵称 | VARCHAR(15) | 无 |
| usr\_name | 真实姓名 | VARCHAR(15) | 无 |
| usr\_gender | 性别 | TINYINT | 无 |
| usr\_major | 专业 | VARCHAR(30) | 无 |
| usr\_imgaddr | 头像地址 | VARCHAR(50) | 无 |
| usr\_phone | 手机 | VARCHAR(15) | 无 |
| usr\_point | 点赞数 | BIGINT | 辅助索引 |
| usr\_attention | 被关注数 | INT | 辅助索引 |
| usr\_credit | 委托信誉 | INT | 复合索引-A-1 |
| usr\_boke | 博客数 | INT | 辅助索引 |
| usr\_finish | 已有效完成委托数 | INT | 辅助索引 |
| usr\_accept | 接收的委托数 | INT | 复合索引-A-2 |
| usr\_publish | 公布状态 | TINYINT | 无 |

**7.4.6 委托表（quest）**

表 11 委托表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| quest\_id | 自增主键 | INT | 主键索引 |
| quest\_time | 委托时间 | DATETIME | 辅助索引 |
| quest\_place | 委托地点 | VARCHAR(30) | 无 |
| quest\_ddl | 截止时间 | DATETIME | 辅助索引 |
| quest\_reward | 报酬 | DECIMAL | 无 |
| quest\_status | 状态 | INT | 无 |
| quest\_acceptor | 接收者 | INT | 无 |
| quest\_assignor | 委托人 | INT | 无 |
| quest\_touch | 联系方式 | VARCHAR(30) | 无 |

**7.4.7 博客表（blog）**

表 12 博客表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| blog\_id | 自增主键 | INT | 主键索引 |
| blog\_content | 博文内容 | VARCHAR(5000) | 无 |
| blog\_title | 博文标题 | VARCHAR(30) | 复合索引-A-1 |
| blog\_brief | 博文简介 | VARCHAR(100) | 无 |
| blog\_keyword | 关键字 | VARCHAR(30) | 复合索引-A-2 |
| blog\_points | 点赞数 | BIGINT | 辅助索引 |
| blog\_status | 状态 | TINYINT | 无 |
| blog\_date | 创建日期 | DATE | 辅助索引 |
| blog\_author | 作者 | INT | 无 |

**7.4.8 评论表（comment）**

表 13 评论表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| comment\_id | 自增主键 | INT | 主键索引 |
| comment\_content | 内容 | VARCHAR(50) | 无 |
| comment\_points | 点赞数 | INT | 复合索引-A-1 |
| comment\_date | 评论日期 | DATE | 复合索引-A-2 |
| comment\_target | 评论博客 | INT | 无 |
| comment\_sender | 评论人 | INT | 无 |

**7.4.9 结伴同行表（together）**

表 14 结伴同行表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| together\_id | 自增主键 | INT | 主键索引 |
| together\_things | 事件 | VARCHAR(30) | 无 |
| together\_touch | 联系方式 | VARCHAR(30) | 无 |
| together\_place | 集合地点 | VARCHAR(30) | 无 |
| together\_ddl | 截止时间 | DATETIME | 无 |
| together\_time | 发布时间 | DATETIME | 辅助索引 |
| together\_sender | 发布者 | INT | 无 |
| together\_noname | 匿名状态 | TINYINT | 无 |

**7.4.10 失物招领表（lose）**

表 15 失物招领表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| lose\_id | 自增主键 | INT | 主键索引 |
| lose\_item | 物品 | VARCHAR(10) | 无 |
| lose\_touch | 联系方式 | VARCHAR(20) | 无 |
| lose\_place | 领取地点 | VARCHAR(15) | 无 |
| lose\_time | 发布时间 | DATETIME | 辅助索引 |
| lose\_sender | 发布者 | INT | 无 |
| lose\_noname | 匿名状态 | TINYINT | 无 |
| lose\_status | 状态 | TINYINT | 无 |

**7.4.11 文明吐槽表（say）**

表 16 文明吐槽表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| say\_id | 自增主键 | INT | 主键索引 |
| say\_content | 内容 | VARCHAR(200) | 无 |
| say\_sender | 吐槽人 | INT | 无 |
| say\_time | 发布时间 | DATETIME | 辅助索引 |
| say\_points | 点赞数 | INT | 辅助索引 |
| say\_critical | 踩数 | INT | 无 |

**7.4.12 管理员表（admin）**

表 17 管理员表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| admin\_name | 管理员姓名 | VARCHAR(15) | 无 |
| admin\_regtime | 登记时间 | DATETIME | 无 |
| admin\_password | 登录密码 | VARCHAR(30) | 无 |
| admin\_touch | 联系方式 | VARCHAR(30) | 无 |
| admin\_token | 管理员令牌 | VARCHAR(15) | 无 |
| admin\_authority | 权限 | INT | 无 |

**7.4.13 收藏表（fond）**

表 18 收藏表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| fond\_id | 自增主键 | INT | 主键索引 |
| fond\_target | 收藏博文 | INT | 无 |
| fond\_sender | 收藏着 | INT | 无 |
| fond\_date | 收藏日期 | DATE | 无 |

**7.4.14 关注表（attention）**

表 19 关注表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| attention\_id | 自增主键 | INT | 主键索引 |
| attention\_target | 关注用户 | INT | 无 |
| attention\_sender | 关注者 | INT | 无 |
| attention\_time | 关注时间 | DATE | 无 |

**7.4.15 管理员动作日志（log）**

表 20 管理员动作日志

|  |  |  |  |
| --- | --- | --- | --- |
| **数据项名** | **说明** | **数据库类型** | **索引** |
| log\_id | 自增主键 | INT | 主键索引 |
| log\_admin | 管理员姓名 | VARCHAR(15) | 无 |
| log\_time | 动作发生时间 | DATETIME | 辅助索引 |
| log\_ip | 管理员主机IP | VARCHAR(15) | 无 |
| log\_desc | 管理员动作描述 | VARCHAR(30) | 无 |

**8. 系统限制**

系统的硬件环境是有限的，而用户的操作不会受到限制，系统设计人员必须凭借有限的硬件设施来应对无限的业务需求增长。系统限制是其中最为关键的手段，通过在数量、时间、大小三个维度的限制，使得系统可以依靠有限的设备，进行扩张。

系统限制对于系统的稳定性、有序性和性能都有着巨大的影响。

**8.1 数量限制**

表 21 数量限制

|  |  |  |
| --- | --- | --- |
| **项目名** | **单位** | **上限数量** |
| 用户博文数 | 篇/人 | 30 |
| 用户单日点赞数 | 次/（人·天） | 10 |
| 用户发布委托数 | 个/人 | 3 |
| 用户关注数 | 个/人 | 50 |
| 用户单日吐槽数 | 条/（人·天） | 3 |
| 用户单日发布信息数 | 条/（人·天） | 5 |
| 用户单日博客发布数 | 篇/（人·天） | 3 |
| 用户发布失物招领数 | 条/人 | 5 |
| 用户发布结伴同行数 | 条/人 | 5 |
| 系统有效通知条数 | 条 | 5 |

**8.2 时间限制**

表 22 时间限制

|  |  |  |
| --- | --- | --- |
| **项目名** | **单位** | **数量** |
| 默认单条失物招领维持时间 | 月 | 1 |
| 默认单条结伴同行维持时间 | 天 | 15 |
| 默认单个委托维持时间 | 天 | 10 |
| 默认管理员行为日志持续时间 | 年 | 1 |
| 默认用户反馈信息持续时间 | 月 | 2 |
| 用户令牌有效时间 | 小时 | 10 |
| 默认 |  |  |

**8.3 大小限制**

表 23 大小限制

|  |  |  |  |
| --- | --- | --- | --- |
| **项目名** | **单位** | **最大数量** | **最小数量** |
| 用户密码长度 | 字符 | 30 | 6 |
| 用户昵称长度 | 字符 | 15 | 1 |
| 真实姓名长度 | 字符 | 15 | 1 |
| 博文长度 | 字符 | 5000 | 500 |
| 博文简介长度 | 字符 | 100 | 0 |
| 博文标题长度 | 字符 | 30 | 1 |
| 博文关键字长度 | 字符 | 30 | 0 |
| 吐槽内容字数 | 字符 | 200 | 10 |
| 同行事件描述 | 字符 | 30 | 2 |
| 招领物品描述 | 字符 | 30 | 2 |
| 地址 | 字符 | 30 | 0 |
| 举报原因 | 字符 | 100 | 10 |
| 联系方式 | 字符 | 30 | 0 |
| 吐槽内容字数 | 字符 | 200 | 10 |
| 评论字数 | 字符 | 50 | 5 |
| 图片大小 | M（兆） | 10 | 0 |
| 系统通知长度 | 字符 | 300 | 0 |
| 系统通知有效时间 | 天 | 30 | 1 |
| 结伴同行截止时间 | 天 | 15 | 1 |
| 委托截止时间 | 天 | 15 | 1 |