“一族一谱”管理系统

系统优化和维护

小组：软件工程第六组

组员：王聪 曾德龙 蔡楷欣 曾欣

程钰涵 陈冠旭 郑湘萍

版次： v1.2

日期：2021年6月29日

文档修改历史记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 修改人 | 日期 | 修改内容 | 备注 |
| v1.0 | 王聪 | 2021.7.5 | 编写初稿 | 初稿 |
| v1.1 | 曾德龙 | 2021.7.6 | 添加数据形式内容 |  |
| v1.2 | 王聪 | 2021.7.7 | 检查版本图 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[1 系统优化 4](#_Toc76590553)

[1.1 兼容性问题 4](#_Toc76590554)

[1.1.1 兼容旧版本 4](#_Toc76590555)

[1.1.2 最低基础库版本 5](#_Toc76590556)

[1.1.3 兼容安卓版本问题 7](#_Toc76590557)

[1.2 降低冗余度 7](#_Toc76590558)

[1.2.1 添加组件 7](#_Toc76590559)

[1.2.2 函数重用 8](#_Toc76590560)

[1.3 数据形式 9](#_Toc76590561)

[2 数据库并发量测试 10](#_Toc76590562)

[2.1 申请修改数据库 10](#_Toc76590563)

[2.1 数据库访问 10](#_Toc76590564)

[3 系统日志 11](#_Toc76590565)

[4 维护设计 12](#_Toc76590566)

1 系统优化

1.1 兼容性问题

为了考虑微信小程序的版本兼容性问题，我们在调用微信小程序的最新API时，从以下两个方面进行考虑：

1.1.1 兼容旧版本

我们会考虑到兼容以前的微信小程序库版本，从而在一些API的设定上会考虑旧版本的API，例如微信小程序的2.10.4版本才能够支函数wx.getUserProfile，新用户将会自动获取用户的头像和昵称，而旧版本用户我们将提示使用按钮，才能获取微信头像和昵称。如下所示，我们现在Load函数中判断是否可以使用wx.getUserProfile函数，如果可以就可以调用该函数：

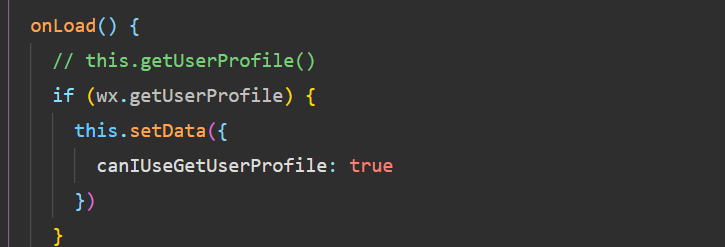


图 1-1 OnLoad中的判断

以下是getUserProfile函数的具体内容：

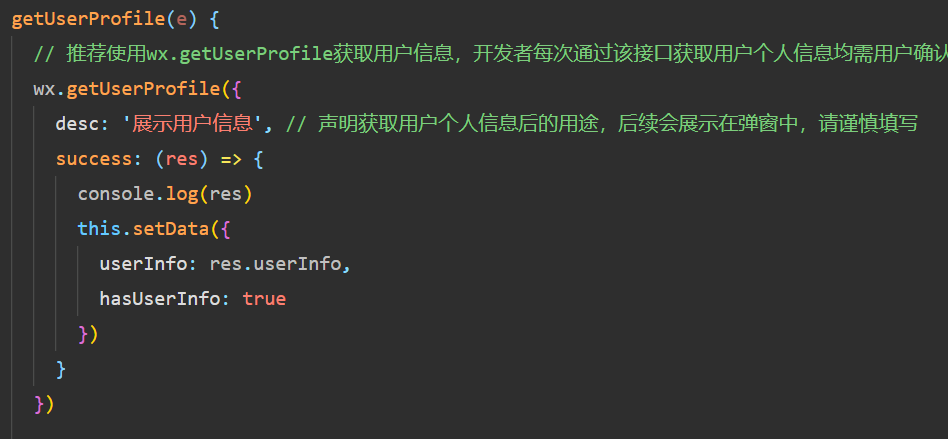


图 1-2 getUserProfile函数

如果用户的版本过低将会提示点击按钮之后获取头像等信息，调用的函数如下所示：

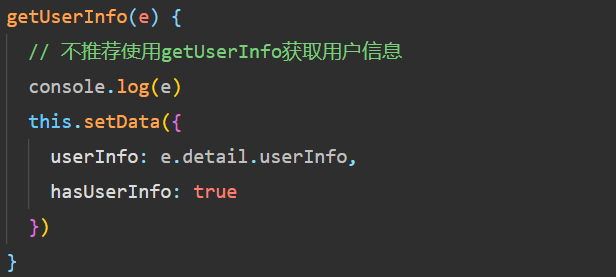


图 1-3 getUserInfo函数

1.1.2 最低基础库版本

为了解决低版本基础库无法兼容小程序的新功能的问题，我们设置了小程序最低基础库版本要求。通过查看近 30 天内访问当前小程序的用户所使用的基础库版本占比，了解当前用户使用的情况，从而设定微信小程序的最低基础库版本。如下图所示：



图 1-4 设置最低基础库版本

具体步骤为：登录小程序管理后台，进入「设置 - 基本设置 - 基础库最低版本设置」进行配置。

如果用户的版本号低于我们设定的版本号将会出现以下提示：



图 1-5 微信小程序的版本过低

1.1.3 兼容安卓版本问题

我们的微信小程序能够兼容2.10.4版本以上的任何手机，从下图中可以看出，已经包含了95%以上的手机使用者。更是兼容了安卓6.0以上的所有的手机的使用。



图1-6 兼容安卓手机示意图

上图来自于<https://developers.weixin.qq.com/miniprogram/dev/framework/client-lib/version.html>，基于微信小程序官方的统计数据，更新时间：2021 年 7 月 7 日。

1.2 降低冗余度

为了降低在编码过程中的冗余度，我们使用了以下几种方式，如下所示。

1.2.1 添加组件

在微信小程序中，我们使用组件的方式对搭建好的组件进行二次利用。例如，微信小程序中的弹出框，我们使用统一的组件，大大降低了代码的冗余度。其中弹出框的代码，我们统一放在dist文件夹下，如下所示为我们小组搭建的组件库。

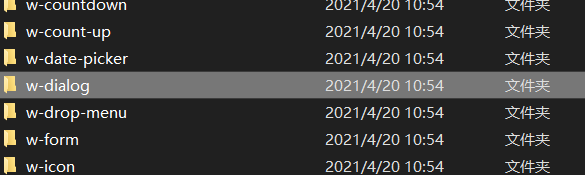


图1-9 搭建的组件库

如果我们想要再次调用该组件，只需要在对应界面的json文件中，进行声明即可。如下图所示，我们调用了如下组件：



图1-10 声明组件

1.2.2 函数重用

为了降低编码过程中的冗余度，我们还尝试了函数重用。将所有可以重用的函数放到utils/utils.js文件中，在使用的时候只需要在对应的js文件下import该文件就可以调用utils/utils.js文件中的函数。如下图所示，为utils.js中的函数。

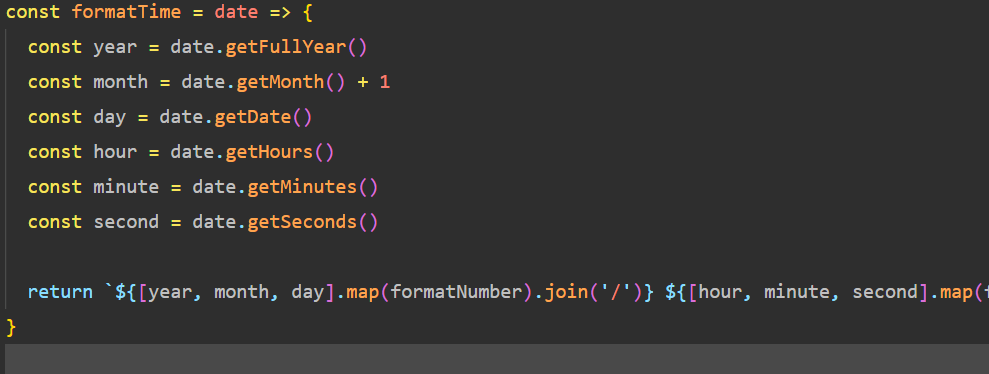


图1-11 utils.js函数

在对应的js文件中import utils就可以调用文件中的函数，从而实现函数重用，如下图所示。

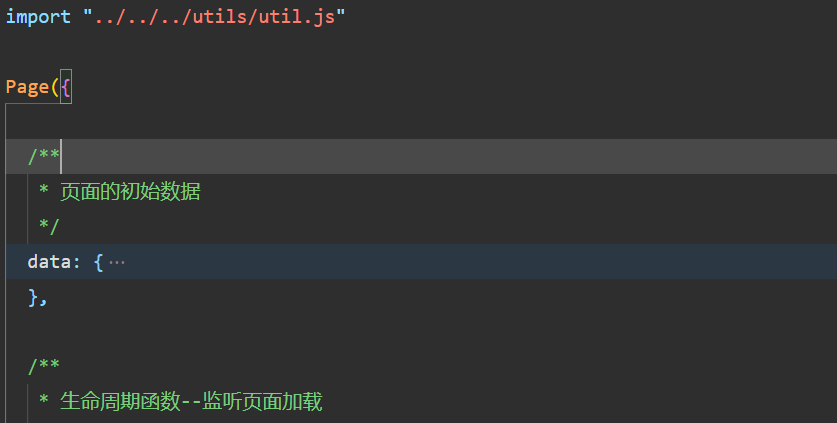


图1-12 调用util中的函数

1.3 数据形式

我们的数据形式采用并查集的形式，并查集的定义如下所示：

并查集，在一些有N个元素的集合应用问题中，我们通常是在开始时让每个元素构成一个单元素的集合，然后按一定顺序将属于同一组的元素所在的集合合并，其间要反复查找一个元素在哪个集合中。这一类问题近几年来反复出现在信息学的国际国内赛题中。其特点是看似并不复杂，但数据量极大，若用正常的数据结构来描述的话，往往在空间上过大，计算机无法承受；即使在空间上勉强通过，运行的时间复杂度也极高，根本就不可能在比赛规定的运行时间（1～3秒）内计算出试题需要的结果，只能用并查集来描述。并查集是一种树型的数据结构， 该种数据形式被用于构建族谱树中，将族人之间的关系数据使用并查集的方式进行构建。在搜索两个族人之间关系的时候，可以加快搜索的速度和减小系统占用的空间。从而大大提升了系统的效率。

2 数据库并发量测试

2.1 申请修改数据库

对于申请修改数据库，我们采用的是管理人员审核批准的形式，所以基本上不存在并发量的问题，因为同一时间只能有一个管理员对数据库进行修改操作。如下图所示。



图2-1 数据库修改申请

2.1 数据库访问

由于我们有多个数据库访问的用户，所以对数据库进行访问的测试是必须的。我们这里采用的是Python的request库文件，结合Python的多线性，在同一时间对数据库分别进行500、800、1000和3000次的访问，情况如下所示。

表1-1 数据库访问

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据库大小 | 500 | 800 | 1000 | 3000 |
| 访问失败次数 | 1 | 5 | 9 | 34 |
| 访问成功概率 | 99.8% | 99.375% | 99.1% | 98.86% |

可以看出在500次访问的时候，访问成功的概率非常的高。基本上可以忽略访问错误的情况。而且我们同一时间访问的人数不可能超过过500，实际上最多会有200人同时访问。可见我们的数据库对于并发量大的问题，有着比较好的应对。

接下来介绍一下，请求数据库中使用到的技术手段，如下所示，我们主要是使用到了Python中的request请求bmob数据库访问，再利用Python中的Thread进行多线程访问。由于篇幅限制，我们这里只给出多线程访问的代码，可以查看我们上交的最终文件中的具体代码。



图2-2 Python多线程访问

3 系统日志

系统日志策略可以在故障刚刚发生时就向你发送警告信息，系统日志帮助你在最短的时间内发现问题。 系统日志是一种非常关键的组件，因为系统日志可以充分了解当前系统的环境。这种系统日志信息对于决定故障的根本原因或者缩小系统攻击范围来说是非常关键的，因为系统日志可以我们了解故障或者袭击发生之前的所有事件。良好的系统日志可以防止从错误的角度分析问题，避免浪费宝贵的排错时间。另外一种原因是借助于系统日志，管理员很有可能会发现一些之前从未意识到的问题，在几乎所有刚刚部署系统日志的环境当中。 为此我们系统数据库的日志采用以下形式：

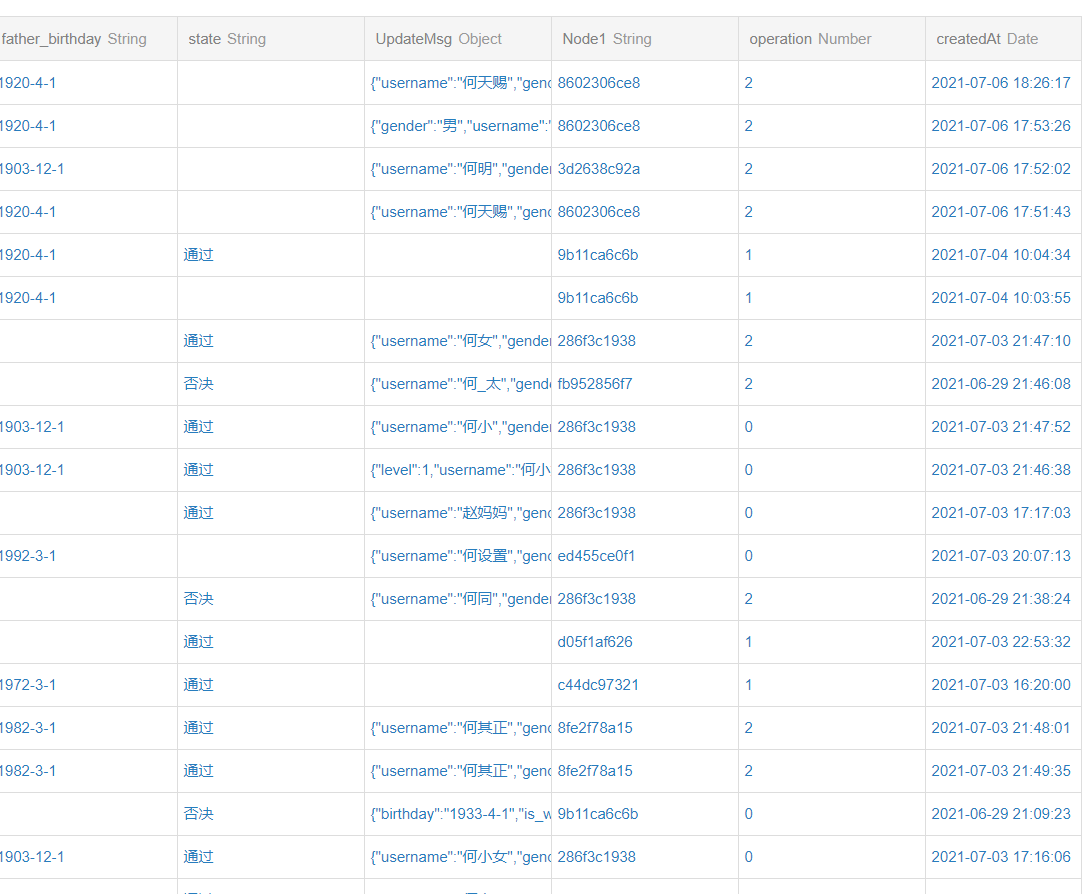


图2-3 系统日志

从上图中可以看得出来，无论何时对数据库进行修改，系统都将会记录操作的时间以及对应的详细信息。

4 维护设计

数据维护方面主要对服务器上的数据库数据进行维护，由于Bomb官方提供了客户端与服务器之间的一个长连接的维护，使得消息更新时由服务端可以主动推送消息，所以数据库内外数据的一致性可以得到足够的保证。

但是，由于Bomb官方没有提供乐观锁等功能，且在数据库层面只有Number类型数据在使用原子计算器时才可以实现用户输入时锁定表，输入完成后解锁表的功能，所以我们需要自行实现上锁操作。因此，我们引入了申请表，对于数据我们需要审批通过之后才可以更新，实际上相当于实现了锁的功能。而对于消息发布功能，由于不同消息的发布时不相关的，实际上只会对数据条目进行添加，删除等功能只有管理员才可进行操作，故而也无需在代码层级上锁。

最后，让专门的数据库管理人员进行定期的数据维护，进行错误的排查即可。