

**详细设计文档**

关于科技资讯整合的

 微信小程序Collect

G20小组·Grateful dead

浙江大学城市学院

计算机与科学技术学院软件工程专业

Release date : 2019年5月4日

Version : V0.2

# 文档说明

文档信息

文档信息表

|  |  |
| --- | --- |
| 文档原作者 | 周磊 |
| 创建日期 | 2019年4月25日 |
| 当前版本 | V0.2 |
| 上次版本 | V0.1 |

**变更记录**

变更记录表

|  |  |  |  |
| --- | --- | --- | --- |
| **变更日期** | **变更人** | **版本** | **备注** |
| 2019.4.25 | 周磊 | V0.1 | 结合总体设计，进行初步详细设计 |
| 2019.5.2 | 周磊 | V0.2 | 更新材料 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[文档说明 2](#_Toc7880814)

[1 引言 6](#_Toc7880815)

[1.1 标识 6](#_Toc7880816)

[1.2 系统概述 6](#_Toc7880817)

[1.2.1 软件的用途 6](#_Toc7880818)

[1.2.2 该软件的一般特性 6](#_Toc7880819)

[1.3 文档概述 6](#_Toc7880820)

[1.4 基线 6](#_Toc7880821)

[2 参考文件 7](#_Toc7880822)

[3 系统级设计决策 7](#_Toc7880823)

[4 系统体系结构设计 7](#_Toc7880824)

[4.1 系统总体设计 7](#_Toc7880825)

[4.1.1 概述 7](#_Toc7880826)

[4.1.2 设计思想 7](#_Toc7880827)

[4.1.3 基本处理流程 11](#_Toc7880828)

[4.1.4 系统体系结构 12](#_Toc7880829)

[4.1.5 功能需求与系统配置项的关系 14](#_Toc7880830)

[4.1.6 人工处理过程 14](#_Toc7880831)

[4.2 系统部件 14](#_Toc7880832)

[4.3 执行概念 15](#_Toc7880833)

[4.4 接口设计 15](#_Toc7880834)

[4.4.1 接口标识和图表 15](#_Toc7880835)

[4.4.2 数据字典 15](#_Toc7880836)

[5 运行设计 16](#_Toc7880837)

[5.1 系统初始化 16](#_Toc7880838)

[5.1.1 用户界面 16](#_Toc7880839)

[5.2 运行控制 22](#_Toc7880840)

[5.2.1 用户点击标栏的板块 涉及到资讯展示模块 23](#_Toc7880841)

[5.2.2 用户点击一条具体的新闻 涉及到具体的资讯展示模块 23](#_Toc7880842)

[5.2.3 用户用户点击评论或是收藏、分享按钮 涉及到资讯的收藏以及资讯的“我喜欢”模块、分享模块 此时需要在服务器端更新数据库 以及调用微信的朋友圈/朋友分享接口 23](#_Toc7880843)

[5.2.4 用户点击我的历史和我的收藏 涉及到时光机模块和资讯的收藏模块 将向云端服务器发送请求 23](#_Toc7880844)

[5.2.5 用户点击个人信息 涉及到用户个人信息的修改模块 23](#_Toc7880845)

[5.3 运行结束 23](#_Toc7880846)

[6 系统出错处理设计 23](#_Toc7880847)

[6.1 出错信息 23](#_Toc7880848)

[6.2 补救措施 24](#_Toc7880849)

[6.3 故障处理技术 24](#_Toc7880850)

[7 系统维护设计 24](#_Toc7880851)

[7.1 检测点的设计 24](#_Toc7880852)

[7.2 检测专用模块的设计 24](#_Toc7880853)

[8 尚待解决的问题 25](#_Toc7880854)

[9 需求的可追踪性 25](#_Toc7880855)

[10 注解 25](#_Toc7880856)

[11 附录 25](#_Toc7880857)

[11.1 接口设计说明 25](#_Toc7880858)

[11.2 数据库(顶层)设计说明(DBDD) 25](#_Toc7880859)

[11.3 用户使用手册 25](#_Toc7880860)

[11.4 测试文档（自动化测试和回归测试） 25](#_Toc7880861)

[11.5 小程序前端伪代码 25](#_Toc7880862)

[12 DBDD（数据库(顶层)设计说明） 25](#_Toc7880863)

[12.1 引言 25](#_Toc7880864)

[12.1.1 标识 25](#_Toc7880865)

[12.1.2 数据库概述 25](#_Toc7880866)

[12.1.3 文档概述 26](#_Toc7880867)

[12.2 参考文件 26](#_Toc7880868)

[12.3 数据库级设计决策 26](#_Toc7880869)

[12.4 数据库详细设计 27](#_Toc7880870)

[12.4.1 暂无 27](#_Toc7880871)

[12.5 用于数据库访问或操纵的软件配置项的详细设计 27](#_Toc7880872)

[12.5.1 暂未开发 27](#_Toc7880873)

[12.6 需求的可追踪性 27](#_Toc7880874)

[12.7 注解 27](#_Toc7880875)

[12.8 数据库系统设计附录 28](#_Toc7880876)

[13 接口设计 28](#_Toc7880877)

[13.1 Jaskson图示例 28](#_Toc7880878)

[13.2 接口简介 30](#_Toc7880879)

[13.3 接口列表 30](#_Toc7880880)

[13.3.1 管理 30](#_Toc7880881)

[13.3.2 首页 30](#_Toc7880882)

[13.3.3 微信 31](#_Toc7880883)

[13.3.4 用户 31](#_Toc7880884)

[13.3.5 通用 31](#_Toc7880885)

[14 软件用户手册 32](#_Toc7880886)

[15 前端伪代码 32](#_Toc7880887)

[15.1 微信小程序登录 32](#_Toc7880888)

[15.1.1 检测用户是否已经注册： 32](#_Toc7880889)

[15.1.2 注册页面： 32](#_Toc7880890)

[15.1.3 登陆页面： 33](#_Toc7880891)

[15.2 用户主界面 33](#_Toc7880892)

[15.2.1 下拉刷新页面： 33](#_Toc7880893)

[15.2.2 上拉刷新页面： 33](#_Toc7880894)

[15.2.3 跳转资讯内容页面： 34](#_Toc7880895)

[15.2.4 切换分类： 34](#_Toc7880896)

[15.3 我的页面 34](#_Toc7880897)

[15.3.1 跳转收藏页面 34](#_Toc7880898)

[15.3.2 取消收藏 35](#_Toc7880899)

[15.3.3 跳转历史页面 35](#_Toc7880900)

[15.3.4 跳转标签页面 36](#_Toc7880901)

[15.4 资讯页面 36](#_Toc7880902)

[15.4.1 点赞 36](#_Toc7880903)

# 引言

## 标识

标题：SE2019春——G20小组详细设计文档

版本号：v0.2

该版本作者：周磊

## 系统概述

### 软件的用途

该软件是为了服务那些喜好阅读科技、IT行业、科研最新的资讯的人群，是其轻松阅读他们喜欢的计算的相关资讯。

### 该软件的一般特性

开发、运行和维护的历史：该系统从2019/3/19开始正式开发，暂未正式投入使用和维护  
投资方：该项目的由G20小组自行负担开发经费  
需求方：在校学生、相关从业人员、对科技有兴趣的人群。  
开发方：G20小组  
支持机构：浙江大学城市学院

## 文档概述

在经过总体工作后，我们大致了解了系统的主要任务和功能，以及模块的划分。而这份文档的主要目的便是确定系统该怎么去实现这些功能，提供相关伪代码，将实现败在用户面前，这份文档会将详细介绍各个模块的相互的联系，和系统之间的调用。这是在编码之前的对整体系统的白皮书

## 基线

本项目的详细设计说明书

# 参考文件

计算机软件文档编制规范 GB/T 8567-2006

《软件工程导论（第六版）》 张海藩 牟永敏 编著 ISBN 978-7-302-33098-1

<https://github.com/FZliweiliang/wechat-app-mall-server>

**微信小程序商城服务端**

# 系统级设计决策

因为我们的主要的功能是资讯的展示提供，所以主要的运行都是在云服务器上部署的，对于云服务器数据的易丢性和可能存在的不稳定性，以及小可能的安全性问题，我们觉得每天都会将新闻和用户的信息进行备份，拷贝到开发人员的电脑本地，防止存在的风险。

# 系统体系结构设计

## 系统总体设计

### 概述

#### 功能描述

1、用户可通过该软件进行登陆注册

2、用户拥有对新闻进行收藏的功能。

3、用户收到的信息都是及时的最新内容。

4、用户可以分享资讯。

5、用户可以在时光机上根据不同时间来查看自己曾经喜欢和收藏的资讯

#### 运行环境

手机端微信小程序

### 设计思想

#### 系统构思

我们的项目由三大部分组成。第一大部分是运行在服务器的爬虫程序，爬虫定期到各大资讯网站上去爬取制定分类的文章。提取出标题、作者、来源和日期信息和标签。然后通过调用数据库的连接接口，将一个资讯作为一条字段加入到数据库中。

在服务器上搭建的后端，需要有相应程序调用的请求的接口【包括相应爬虫程序的插入还有小程序前端调用的请求】，要有当数据出错的警告或者安全措施

在小程序前端设计上，以微信开发者文档为指导进行开发，要有用户注册的模块和界面布局设计（包括与服务器的数据库的调用）。相关分类的资讯的展示模块和界面布局设计（包括与服务器的数据库的调用） 。具体资讯的展示的模块和界面布局设计（括与服务器的数据库的调用） . 具体资讯的喜欢、分享、收藏的功能模块（包括与服务器的数据库的调用和微信接口的调用）

用户信息的修改（包括与服务器的数据库的调用和微信接口的调用 。时光机功能（展示全部收藏信息、指定时间段）

#### 关键技术与算法

#### 爬虫相关技术

我们将采用scarpy+multiprocessing的技术多线程爬取数据，编程语言采用python来实现。

数据的存储方面，我们将使用MongoDB和scrapy\_redis来搭建分布式爬虫。

#### 新闻分类相关技术

利用python语言构建朴素贝叶斯分类器，通过网络上已经公开的词库和人工的审核添加，进行模型的训练达到分类新闻的效果。

#### 服务器后端技术和算法

采用的是Nginx + Node.js + Express的框架，让开发更高效和容易上手。重要的是和服务器调用（Require(数据库表)），GET和POST（相关HTTP知识）

#### 数据库技术

数据库的技术：采用MongoDB，具体详见数据库设计文档，存入的数据为了方便查找应该建立索引

#### 前端技术

因为微信小程序主要基于JavaScript的语言，CSS+HTML5，所以我们界面设计我们主要是参考已有的界面美观的小程序的源码，进行符合我们的界面要求的修改

#### 关键数据结构

// 引入 api 路由

const routes = require('./routes/index')

// api 路由

app.use('/api', routes)

//配置服务端口

const server = app.listen(8082, function () {

let host = server.address().address;

let port = server.address().port;

console.log(host, port);

})

exports.user = (req, res, next)=>{ //验证用户是否登录

const openid = req.query.openid||req.body.openid

User.find((err,data)=>{

console.log(data)

})

if(!openid){

return res.json({

code: -400,

message: '请登录',

})

}

User.findOneAsync({ //查询数据库是否有该用户

openid:openid,

})

.then(result => {

if (result&&openid) {

next()

} else {

return res.json({

code: -400,

message: '不存在该用户',

})

}

})

.catch(err => {

res.json({

code: -200,

message: err.toString(),

})

})

}

exports.uploadBanner = (req,res) => {

//生成multiparty对象，并配置上传目标路径

const form = new multiparty.Form({uploadDir: './file/banner'});

form.parse(req,(err, fields, files)=>{

if(err){

console.log(err);

}else{

var inputFile = files.inputFile[0];

let data = {

img\_url:inputFile.path, //图片地址

href:fields.href[0], //跳转地址

name:fields.name[0], //名称

is\_show:fields.is\_show[0], //是否显示

effective:fields.effective[0].split(','), //有效期

}

return Banner.createAsync(data).then(result => {/\*根据Model创建了一个实例 \*/

res.json({

code: 200,

message: '添加成功',

data: result

})

}).catch(err => {

res.json({

code: -200,

message: err.toString()

})

})

}

})

}

路由：

// ------- 微信 -------

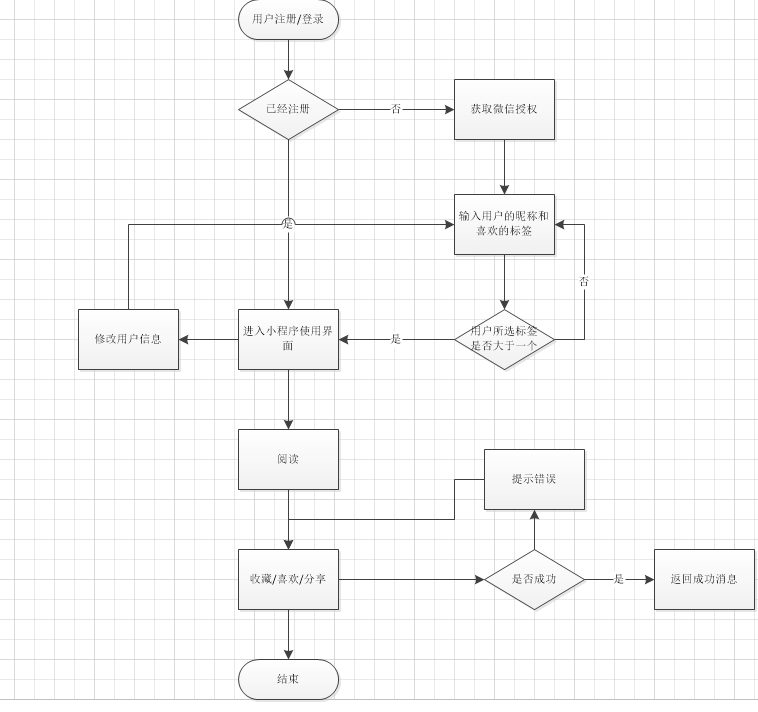
router.get('/v1/wx/getUser',userApi.getWxUser) //获取微信用户信息

// ------- 用户 -------

router.post('/v1/user/bindMobile',intercept.user,userApi.bindMobile) //绑定手机号

### 基本处理流程

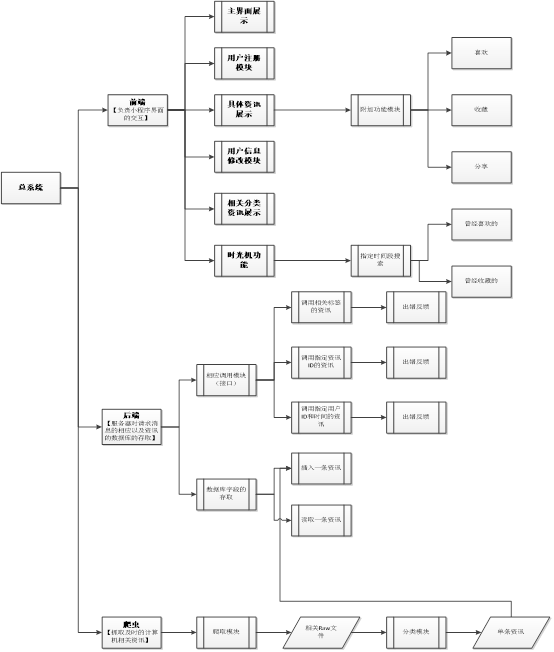
#### 系统流程图



#### 数据流程图

### 系统体系结构

#### 系统配置项（见下页）



#### 系统层次结构

#### 

#### 系统配置项设计

页面展示模块、消息请求、喜欢、收藏、分享、时光机、爬虫、分类、数据库连接、数据库查询、数据库插入模块

### 功能需求与系统配置项的关系

### 人工处理过程

暂无，除了需要人工定期查看数据库的信息是否有异常

## 系统部件

硬件支持：阿里云服务器（学生机——轻量应用服务器，2核2G）

软件支持：微信小程序开发平台、Nginx、Express、NodeJS、MongoDB

## 执行概念

详见上下图

## 接口设计

### 接口标识和图表

和唐敏敏一起分析

### 数据字典

|  |  |
| --- | --- |
| 数据结构卡片编号：1  数据结构名称：网页元素  别名：原新闻 | |
| 说明：爬虫程序从网页直接爬取到的数据。 | 有关数据流/数据储存：  从网页上爬取下来  信息容量：10000条 |

|  |  |
| --- | --- |
| 数据结构卡片编号：2  数据结构名称：新闻分类表  别名：不同的新闻 | |
| 说明：每条新闻都带有相关的分类信息。分类标签 = 标签ID+标签名字+其他相关标签。 | 有关数据流/数据储存：  存储到Mongo DB中的新闻表中  信息容量：10000条 |

|  |  |
| --- | --- |
| 数据结构卡片编号：3  数据结构名称：喜欢的新闻  别名：我的喜欢表 | |
| 说明：这是用户点击“喜欢”按钮时产生的数据。  喜欢信息 = 喜欢的时间 + 但用户id + 新闻的id + 新闻的标签 | 有关数据流/数据储存：  从用户接口到「我的喜欢表」  信息容量：5000条 |

|  |  |
| --- | --- |
| 数据结构卡片编号：4  数据结构名称：历史记录表  别名： | |
| 说明：这是用户每次浏览新闻时产生的数据  历史记录 = 用户id + 阅读时间 + 新闻id + 新闻标签 | 有关数据流/数据储存：  从用户接口到「我的阅读历史表」  信息容量：20000条 |

# 运行设计

## 系统初始化

### 用户界面

#### 微信外部



#### 资讯首页及“不感兴趣”此新闻的提醒



#### 资讯详细内容



#### “我的”板块



#### 时光机功能



该系统的初始化主要是通过重构数据库的数据来完成的

#### 相关操作警告



#### 原型链接

<https://org.modao.cc/app/f2be5b741d4668d75bfd5821c2b9b92ad07d1b60>

## 运行控制

用户点击进入小程序app  
涉及到主界面展示模块和资讯展示模块、读取模块  
此时需要调用云端服务器

### 用户点击标栏的板块 涉及到资讯展示模块

### 用户点击一条具体的新闻 涉及到具体的资讯展示模块

### 用户用户点击评论或是收藏、分享按钮 涉及到资讯的收藏以及资讯的“我喜欢”模块、分享模块 此时需要在服务器端更新数据库 以及调用微信的朋友圈/朋友分享接口

### 用户点击我的历史和我的收藏 涉及到时光机模块和资讯的收藏模块 将向云端服务器发送请求

### 用户点击个人信息 涉及到用户个人信息的修改模块

## 运行结束

用户点击可以通过两种方式退出该系统

1.点击“退出小程序”按钮，即可退出

2.直接退出微信也可以退出该系统

# 系统出错处理设计

## 出错信息

|  |  |  |  |
| --- | --- | --- | --- |
| 微信小程序 | 错误类型 | 出错模块 | 处理方式 |
| 网络错误 | 新闻读取模块 | 提示错误，停止页面读取 |
| 操作失败 | 获取个人收藏  获取个人标签信息  获取历史记录  分享到微信  评论 | 告知用户 建议重试 |

## 补救措施

1.通过弹出窗口告知用户进行重试

2.回滚数据库

3.在清晨或是午夜不繁忙的时候，自动备份数据库和运行日志

4.发生错误时的时候，G20小组按照工作分工，对出错的模块进行分析，尽快使系统重新上线

## 故障处理技术

**故障处理技术：  
1、事务故障的恢复：撤消事务（UNDO）  
2、系统崩溃故障的恢复：撤消事务（UNDO）+重做事务（REDO）  
3、磁盘（介质）故障的恢复  
4、灾难故障的恢复：远程备份**

# 系统维护设计

## 检测点的设计

采用日志模块记录系统运行过程状态以及出现的问题。

系统外部接口调用前后都要进行日志的详细记录，方便接口调试；系统中重要的状态信息的变化都要通过日志记录下来，方便查问题时还原现场，推断程序运行过程；系统入口与出口记录输入与输出，方便定位问题；任何业务异常都进行日志详细记录；日志记录要求清晰准确，能够通过日志准确定位系统问题

## 检测专用模块的设计

待补充

# 尚待解决的问题

接口能不能统一设计？

分类算法的实现难度可能过大？

# 需求的可追踪性

我们会在每个里程碑前和长期用户确认需求是否达到他们的要求，同时对照着需求文档进行系统的修改和实现。

# 注解

暂无

# 附录

## 接口设计说明

## 数据库(顶层)设计说明(DBDD)

## 用户使用手册

## 测试文档（自动化测试和回归测试）

## 小程序前端伪代码

# DBDD（数据库(顶层)设计说明）

## 引言

### 标识

DBDD，Collect小程序数据库顶层说明，作者周磊，版本号v0.1，发行号v0.1

### 数据库概述

本数据库主要用途是为了给爬虫程序提供不同资讯的存放的一个容器。同时允许用户在使用小程序的时候通过接口来调用数据库中的资讯，以在界面展示。

本数据库采用的是非关系型数据库，采用此类型的数据库的原因是考虑到我们具体的需求。因为网页的爬取的信息并不是有规律的，不同的网页，图片和文字的布局都不一样。所以不能按照同一模板来存放不同字段的信息。所以我们采用了非关系型数据库，先把整个网页的文档以JSON格式存下来。在通过分类的程序，把关键字和页面的布局存下来。这样比Mysql等关系型数据库的存放更为灵活。

维护方法我们的开发人员会每隔两天查看检查数据库是否有异常（字段的内容是否合理，有没有怪异的参数信息）还有服务器的运行的压力（暂时未讨论出明确的解决方案）。

数据库搭建在Nginx + Node.js + Express的坏境下。选择的是非关系型数据库的热门数据库MongoDB。

数据库部署在阿里云主机上，24小时不间断运行，同时我们会定期的创建快照，保证资料的安全和稳定。

### 文档概述

本文档用于向开发人员和客户介绍本项目采用的数据库的数据结构、定义的接口和交互的逻辑，并尽量简洁明了的介绍数据库所起到的作用和数据的流向以及处理过程，相信这在对后期的测试和数据库的开发是一个指南。

## 参考文件

* **GB/T 8567-2006 《中华人民共和国国家标准-计算机软件文档编制规范》 之** *7.14数据库(顶层)设计说明(DBDD)*
* **《软件工程导论（第六版）》 张海藩 牟永敏 编著 ISBN 978-7-302-33098-1**
* **《PMBOK》项目管理计划书**

## 数据库级设计决策

* 关于该数据库应接受的查询或其他输入和它应产生的输出(显示、报告、消息、响应等)的设计决策————通过Node.js结合HTTP的GET和POST，调用接口
* 有关响应每次输入或查询的数据库行为的设计决策，包括动作、响应时间和其他性能特性、所选择的方程式/算法/规则、配置和对不允许的输入的处理。——————每次输入，【1】如果是请求搜索相关ID和标签的资讯的集合，将通过数据库语言在数据库中搜索（提前会建立好索引并定期更新），搜索并返回对应的信息。【2】如果是爬虫请求插入信息，先检测字段有无异常，无异常则插入，并返回成功结果。响应时间每条不应超过5秒
* 有关数据库/数据文件如何呈现给用户的设计决策————通过前端的设计以一个整洁的布局将资讯展示给用户
* 有关数据库要提供的可用性、保密性、私密性和运行连续性的层次与类型的设计决策——————数据库的登录信息只有组内开发人员知道，并且使用前需在小组内申明理由并获得全员同意。同时给服务器增加一些安全的保障（购买额外的服务），同时每天给服务器的数据库记录进行备份，选择的阿里云的主机运行较为稳定且广受好评
* 有关备份与恢复的设计决策，包括数据与处理分布策略、备份与恢复期间所允许的动作、对例如音像等新技术或非标准技术的特殊考虑——————暂无思路

## 数据库详细设计

### 暂无

## 用于数据库访问或操纵的软件配置项的详细设计

### 暂未开发

## 需求的可追踪性

暂无，我们需要和固定用户不断地联系，并且以可以运行的原型展示，测试用户指定的需求，若满足，则数据库设计达到要求。（压力测试？）

## 注解

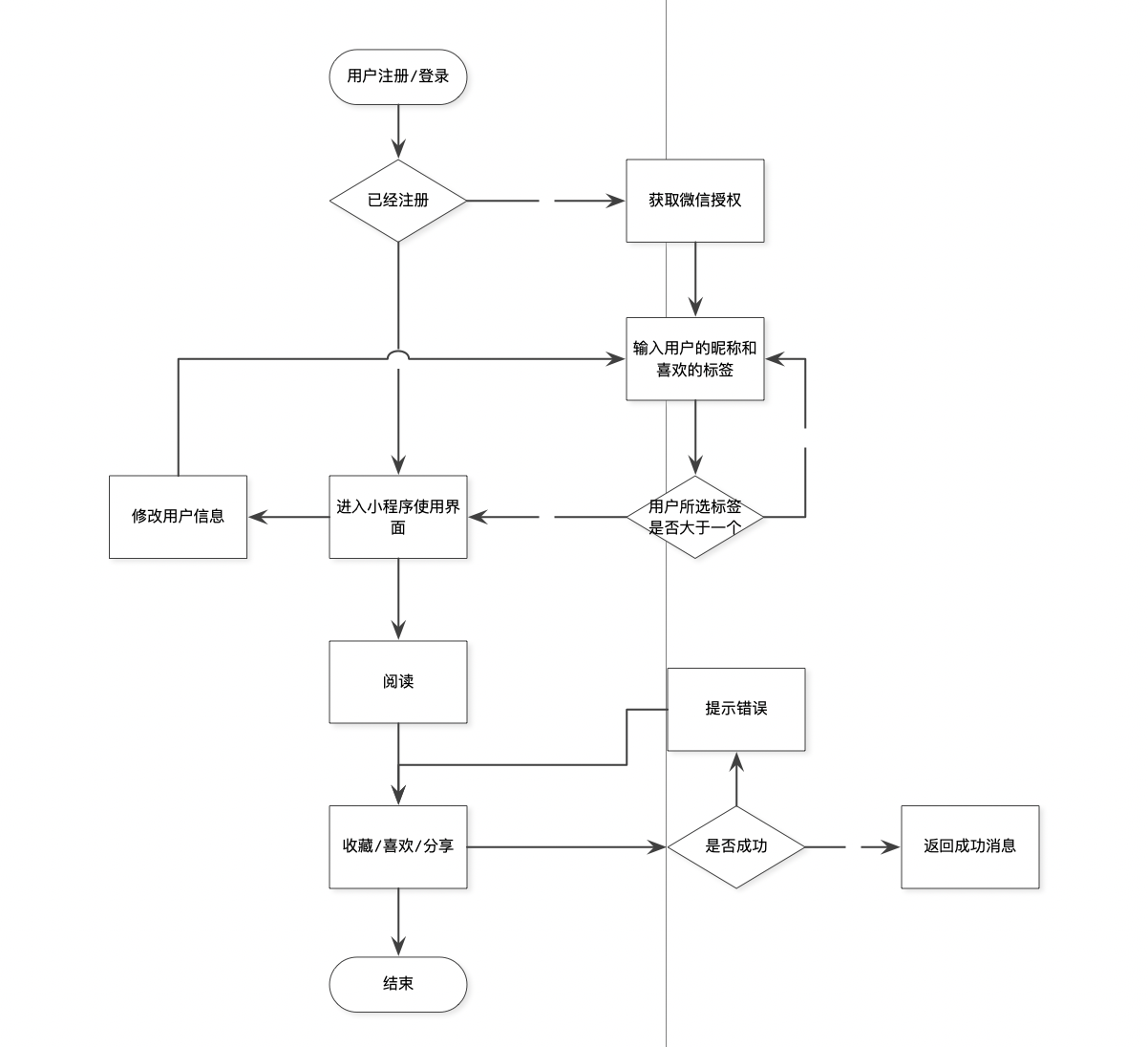
暂无

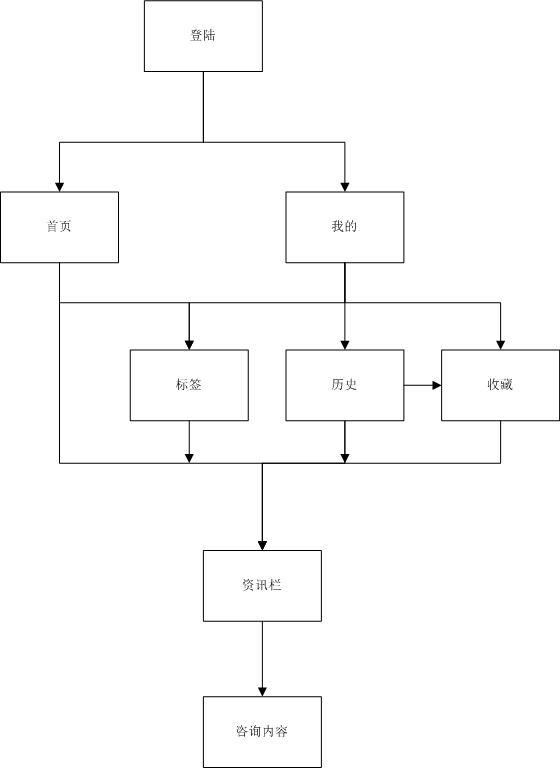
## 数据库系统设计附录

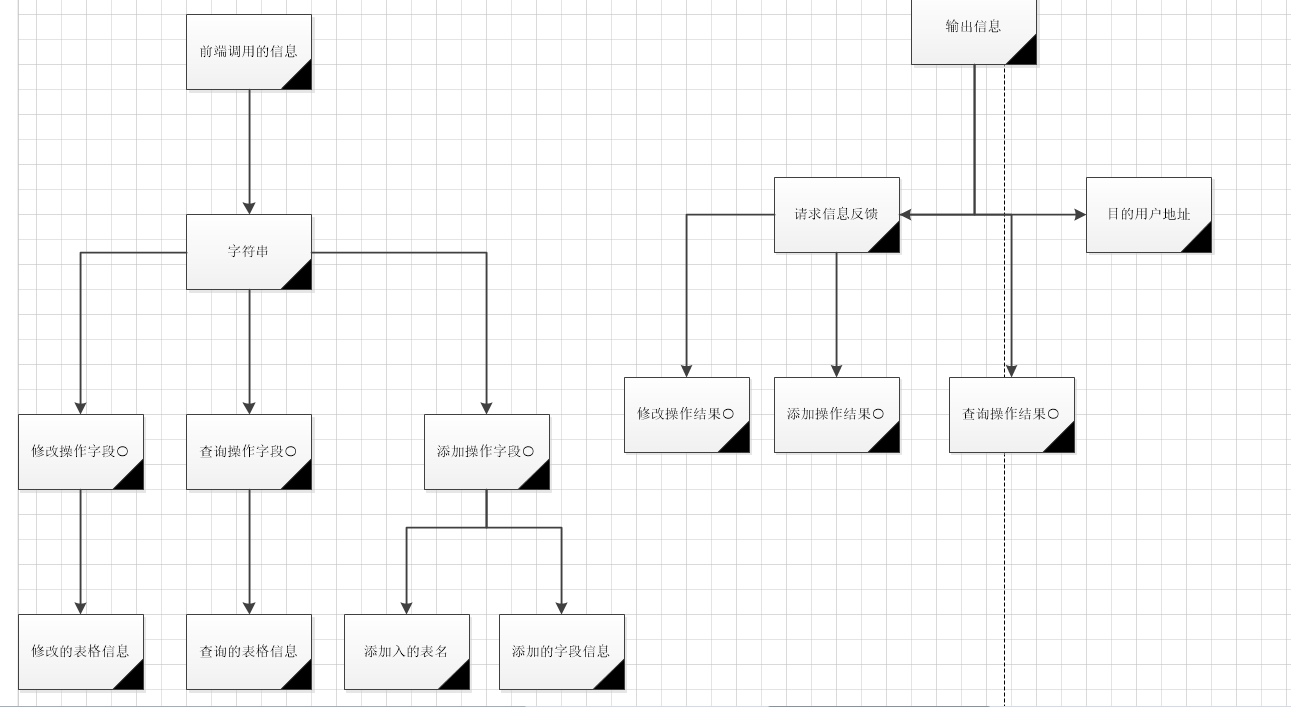
暂无

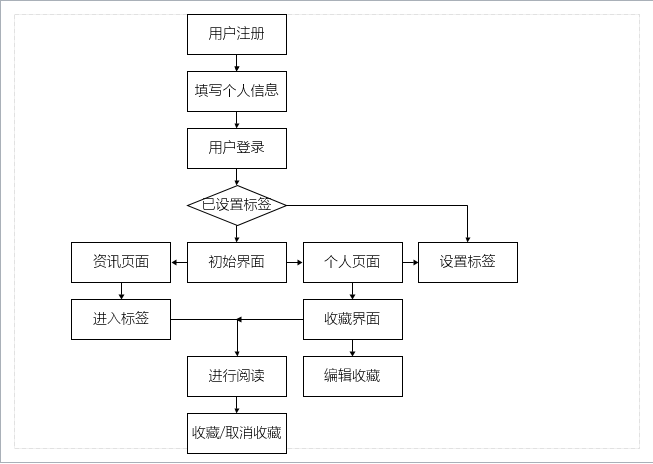
# 接口设计

## Jaskson图示例









## 接口简介

小程序需要后台的接口来对消息的调用进行响应，其目的是为了使对应的请求能获得正确且高效的响应和反馈。

## 接口列表

### 管理

| **Name** | **Method** | **Default** | **Description** |
| --- | --- | --- | --- |
| /v1/admin/delUser | get | auto | 删除用户 |
| /v1/admin/addNews | post | auto | 添加新闻 |
| /v1/admin/delNews | get | auto | 删除新闻 |
| /v1/admin/delClass | get | auto | 删除分类 |
| /v1/admin/addClass | post | auto | 添加分类 |

### 首页

| **Name** | **Method** | **Default** | **Description** |
| --- | --- | --- | --- |
| /v1/home/bannerList | get | auto | 获取banner |
| /v1/home/getHotList | get | auto | 获取推荐列表 |
| /v1/home/getList | get | auto | 获取列表 |
| /v1/home/getItem | get | auto | 获取详情 |

### 微信

| **Name** | **Method** | **Default** | **Description** |
| --- | --- | --- | --- |
| /v1/wx/getUser | get | auto | 获取微信用户信息 |

### 用户

| **Name** | **Method** | **Default** | **Description** |
| --- | --- | --- | --- |
| /v1/user/bindIde | post | auto | 绑定身份信息 |
| /v1/user/modifyNickname | post | auto | 添加/修改昵称 |
| /v1/user/editCategory | post | auto | 设置喜爱分类 |

### 通用

| **Name** | **Method** | **Default** | **Description** |
| --- | --- | --- | --- |
| /v1/public/getClassList | get | auto | 获取分类列表 |

# 软件用户手册

详见附件[（ SE2019春-G20-软件用户手册(SUM)）](SE2019春-G20-软件用户手册(SUM).doc)

# 前端伪代码

## 微信小程序登录

### 检测用户是否已经注册：

procedure 用户登录界面

if 没有注册过 then

跳转至注册界面

else

跳转至用户首界面

end if

end 用户登录界面

### 注册页面：

procedure 用户注册界面

从分类栏选择推送标签

读入标签id

选择一个标签，则 标签数++

if <标签数=0> then

弹出对话框：请选择一个以上的标签

else if<标签数>0>then

弹出对话框：操作成功

end if

end if

end 用户注册界面

### 登陆页面：

Procedure 用户登陆页面

If <小程序已登录> then

Procedure 首页

Else if <小程序未登录> then

Procedure 登陆弹窗

If click <确定> then

Procedure 主页

Else if click <取消> then

Procedure 登陆弹窗

输出：请重新登录

end if

## 用户主界面

### 下拉刷新页面：

Procedure 主页

onPullDownRefresh()

出现小点

and打印：玩命加载中

与服务器段连接wx.request{}

wx.hideLoading()

wx.stopPullDownRefresh()

if <加载成功> then

输出：刷新成功

输出新载入的资讯栏内容

else if<加载失败> then

输出：刷新失败，请重试

end if

### 上拉刷新页面：

Procedure 主页

获取当前页和共几页

onReachBottom()

打印：玩命加载中

加载成功，页数++

与服务器端连接wx.request

wx.hideLoading()

if <加载成功> then

输出：加载成功

and输出新载入的资讯栏内容

else if<加载失败> then

输出：加载失败，请重试

end if

### 跳转资讯内容页面：

Procedure 主页

if <获取内容成功> then

跳转该条资讯页面

输出资讯内容

else if <获取内容失败> then

弹出：获取资讯失败，请重试

end if

### 切换分类：

Procedure 主页

If click 某一分类 then

输出该分类的资讯

End if

If click 标签图标 then

Procedure 标签页面

End if

## 我的页面

### 跳转收藏页面

Procedure 我的

If click 收藏 then

Procedure 收藏页面

if c <收藏数=0>then

打印：暂无收藏资讯

else if c <收藏数>0>then

Procedure 一天内

If c <收藏时间<1天>then

输出资讯栏内容

Procedure 一周内

Else if c <收藏时间<1周>then

输出资讯栏内容

Procedure 十五天内

Else if c <收藏时间<15天>then

输出资讯栏内容

End if

End if

### 取消收藏

Procedure 收藏页面

If click 删除 then

选择，选择数++

if<选择数=0> then

弹出对话框：你还没有选择任何资讯

Else

弹出确定框

If click 确定 then

将这些收藏的资讯从数据库中删除

弹出对话框：删除成功

Else if click 取消 then

End if

end if

end if

### 跳转历史页面

Procedure 我的

If click 历史 then

Procedure 历史页面

if<历史数=0>then

打印：暂无收藏资讯

else if<历史数>0>then

Procedure 一天内

If c <历史时间<1天>then

输出资讯栏内容

If click 一周内 then

Procedure 一周内

Else if c <历史时间<1周>then

输出资讯栏内容

End id

If click 一周内 then

Procedure 十五天内

Else if c <历史时间<15天>then

输出资讯栏内容

End if

End if

End if

### 跳转标签页面

Procedure 我的

If click 标签then

Procedure 标签页面

If click 添加 then

弹出添加符号

If click 加号 then

订阅标签数++

发送标签id至后端

End if

End if

If click 删除 then

弹出删除符号

If click 减号 then

订阅标签数--

发送删除请求至后端

End if

If <标签数<0> then

弹出对话框：请留下一个标签

End if

End id

## 资讯页面

### 点赞

Procedure 资讯内容

If click 点赞 then

点赞数++

End if

If click 点赞 then

点赞数--

End if