# 概要设计

## 一、总体设计

### 1.1 需求规定

功能性要求：构建一个完整的阅读小程序，实现阅读，推荐，背单词，查询单词，收藏文章等一系列功能。

输入项目为用户的一系列操作请求

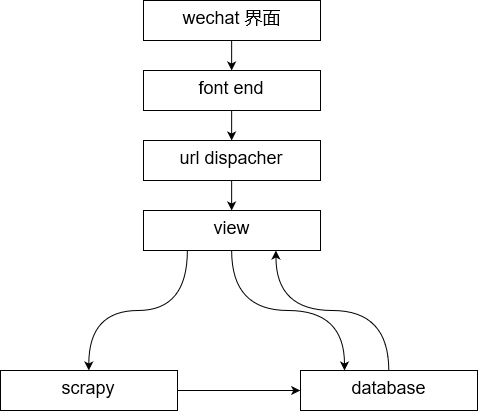
输出项目为针对不同请求的响应。

### 1.2 运行环境

* 安卓4.5及以上
* ios设备11.0及以上

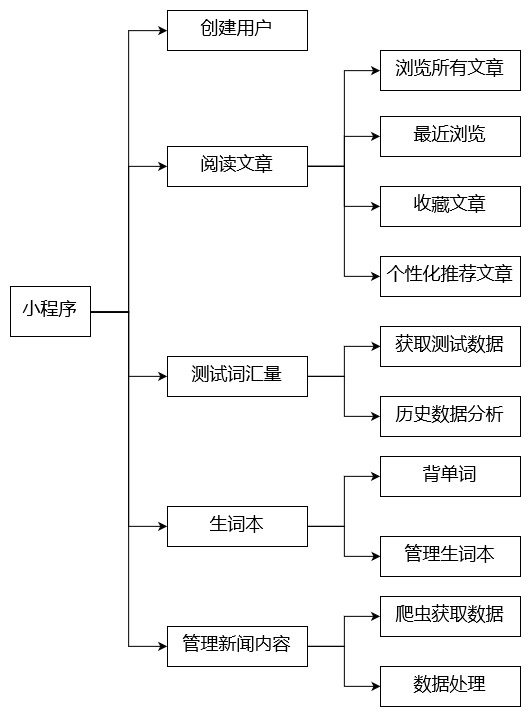
### 1.3 基本设计概念和处理流程

系统整体架构如下。用户在微信小程序界面进行操作，前端处理用户各种操作请求，针对需要涉及到数据交互以及复杂操作的命令，则交由后端处理。后端对于前端的http请求，首先从映射的url列表中一一匹配，找到合适的则进行处理并返回处理结果。



### 1.4 结构

小程序用户角色只有阅读者，无管理员等角色。功能主要分为四大模板，用户登陆小程序，创建身份，并获取唯一标识。阅读文章功能包含四个子系统，分别是浏览所有文章模板，查看最近浏览文章，收藏某文章以及获取个性化推荐的文章。测试词汇量下属两个子模板，分别是获取测试的词汇数据，以及查看历史测试结果。用户可以通过生词本背单词，并且对单词进行删除添加等操作。



### 1.5 人工处理过程

从新闻网站爬取数据以及将数据移植到服务器的数据库中需要手动操作。

## 二、接口设计

### 2.1 用户接口

用户接口主要功能为用户对文章，生词进行一系列相关处理，并且对个人的生词数据进行操作。

|  |  |
| --- | --- |
| 接口 | 功能 |
| 浏览推荐文章 | 根据用户词汇测试数据返回推荐的文章 |
| 登陆 | 获取用户登陆状态以及唯一标识 |
| 按分类浏览文章 | 获取某分类所有文章 |
| 管理生词本 | 删除/添加单词 |
| 背单词 | 获取生词本所有单词开始记忆 |
| 测试词汇量 | 获取待测试词汇，处理测试结果 |
| 收藏文章 | 收藏文章信息，可供再次查看 |
| 最近浏览 | 查看最近阅读的文章 |

### 2.2 外部接口

外部接口主要用于与数据库进行信息交互。

|  |  |  |
| --- | --- | --- |
| 接口类型 | 接口 | 功能 |
| 软件接口 | 与数据库接口 | 连接数据库并进行信息交互 |
| 服务器接口 | 微信服务器接口 | 从微信服务器解码code，获取用户openid |

### 2.3 内部接口

内部接口主要是系统内部功能，例如利用爬虫获取新闻资源，对文章内容进行一定的处理，不同模块之间的调用。

|  |  |
| --- | --- |
| 接口 | 功能 |
| 添加测试记录 | 获取用户测试词汇结果并存入数据库 |
| 查询测试记录 | 返回用户历史测试记录 |
| 添加单词 | 向服务器生词本添加单词 |
| 删除单词 | 删除生词本中指定单词 |
| 查询单词 | 返回所有生词 |
| 添加收藏记录 | 将收藏文章的信息录入数据库 |
| 生成测试词汇 | 随机生成用户测试词汇数据 |
| 录入文章信息 | 将服务器本地文章信息录入数据库 |
| 爬取新闻资源 | 调用scrapy框架获取新闻资源 |

## 三、运行设计

该部分主要描述针对系统施加不同外界运行控制时所引起的各种不同的运行模块组合。

### 3.1 运行模块组合

#### 3.1.1 阅读文章模块组合

* 用户浏览所有文章，点击查看文章，添加生词，添加收藏功能模块组合。
* 用户测试词汇，查看个性化推荐文章功能模块组合。
* 用户查看单词本，删除单词，背单词功能模块组合。
* 用户查看最近浏览文章，点击阅读，添加收藏功能模块组合
* 用户背单词，选择单词释义，查看测试结果功能模块组合
* 用户点击分类，阅读文章功能模块组合。

#### 3.1.2 处理新闻资源模块组合

* 后台运行爬虫程序，获取新闻资源，录入数据库功能模块组合
* 后台录入新闻资源，进行数据整理，查询数据功能模块组合
* 后台根据用户信息，获取数据库资源，进行数据比较，返回文章信息功能模块组合
* 后台根据文章图片地址，修改图片名，下载图片并修改数据库功能模块组合

#### 3.2 运行控制

#### 3.2.1 阅读文章模块组合

* 用户需要授权小程序获取用户id才能阅读文章
* 用户需要至少测试过一次词汇量才可以获取个性化推荐文章

### 3.3 运行时间

#### 3.3.1 阅读模块运行时间

系统主要的运行时间消耗在对数据库进行增删改查的操作以及前后端交互过程中大量数据传输的时间。当后台文章未及时清理，导致文章条目不断变多，会致使用户浏览全部文章时，后台一次性返回的数据量过多，加载缓慢。

#### 3.3.2 处理新闻资源运行时间

主要时间消耗在获取文章资源，由于外文网站服务器在国外，相对时延较大，并且在爬取的过程中，框架本身的特性也会造成一定的时间消耗。所以时间的开销不能忽略。

但是好在文章爬取一天仅做一次，并且并非实时获取。所以对于用户体验影响不大。

#### 3.3.3 耗时少的功能模块：

直接在前端实现的所有功能都耗时较少。例如，对用户测试词汇的数据进行分析，按照分类展示文章等等。

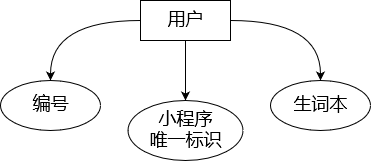
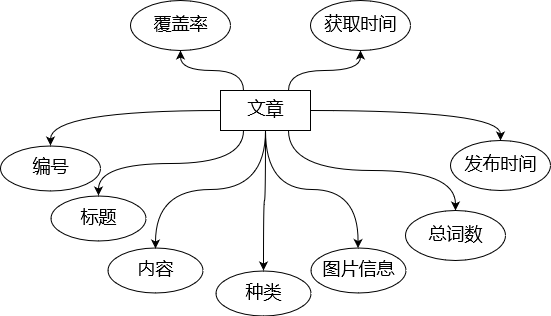
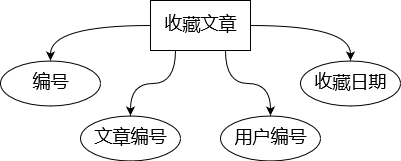
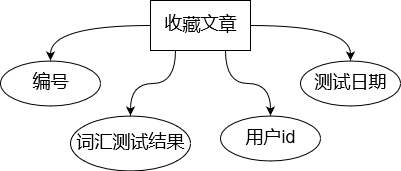
#### 3.3.4 耗时大的功能模块：

与后端进行大量数据传输的功能模块，以及与数据库进行交互的模块。

## 四、系统数据结构设计

### 4.1 逻辑结构设计要点

系统涉及到的实体主要有如下几种，文章，用户，词汇测试记录，收藏文章信息。

* 用户（编号，小程序唯一标识，生词本）
* 
* 文章（编号、标题、内容、作者、种类、图片信息、总词数、发布时间、覆盖率、获取时间）
* 
* 收藏文章（编号、文章编号、用户编号、收藏日期）
* 
* 词汇测试记录（编号、词汇测试结果、用户id、测试日期）
* 
* ER图
* 

### 4.2 物理结构设计要点

用户编号是用户唯一标识加密过后的结果，为了保证传输过程的安全性，在一般传输过程中都使用用户编号进行操作。生词本记录了用户生词本在服务器中的存储位置。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 主键/索引 | 约束 | 是否可为null |
| 用户编号 | varchar | 500 | 主键 |  | 否 |
| 用户唯一标识 | varchar | 500 |  |  | 否 |
| 生词本 | varchar | 500 |  |  | 否 |

文章通过文章编号唯一标识，该属性是自增的。考虑到文章内容有长有短，内容过长在数据库会占据较大空间，因此文章内容存储的是文章存储在服务器上的位置，是一个json文件。图片信息也是如此。只有一点特别的就是图片存储在服务器文件的指定文件夹下，可以通过url直接访问。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 主键/索引 | 约束 | 是否可为null |
| 文章编号 | integer |  | 主键 |  | 否 |
| 标题 | varchar | 500 |  |  | 否 |
| 文章内容 | varchar | 500 |  |  | 否 |
| 作者 | varchar | 500 |  |  | 否 |
| 类型 | varchar | 500 |  |  | 否 |
| 图片信息 | varchar | 500 |  |  | 否 |
| 总词数 | integer |  |  |  | 否 |
| 出版时间 | varchar | 500 |  |  | 否 |
| 覆盖率 | varchar | 500 |  |  | 否 |
| 获取时间 | date |  |  |  | 否 |

测试内容由编号唯一标识，该属性自增。外键为用户编号。一个用户可以有零到多条测试记录。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 主键/索引 | 约束 | 是否可为null |
| 测试编号 | integer |  | 主键 |  | 否 |
| 覆盖率 | varchar | 500 |  |  | 否 |
| 用户编号 | varchar | 500 |  |  | 否 |
| 日期 | date |  |  |  | 否 |

收藏条目由编号唯一标识，属性自增。外键为文章编号与用户编号。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 类型 | 宽度 | 主键/索引 | 约束 | 是否可为null |
| 收藏编号 | integer |  | 主键 |  | 否 |
| 文章编号 | integer |  |  |  | 否 |
| 用户编号 | varchar | 500 |  |  | 否 |
| 日期 | date |  |  |  | 否 |

### 4.3 数据结构与程序的关系

后端在获取用户测试结果，记录用户操作信息时需要对数据库中的表进行修改，即增删改查操作。

物理数据结构主要用于各模块之间函数的信息传递。接口传递的信息是数据结构封装了的数据，以参数或者返回值的形式在各模块之间传输。

## 六、系统出错处理设计

### 5.1 出错信息

|  |  |  |
| --- | --- | --- |
| 出错类型 | 出错形式 | 处理办法 |
| 404 | 从后端获取数据，显示内容不存在 | 检查发送请求的url地址以及http请求的格式，逐项排查。 |
| 500 | 从后端获取数据，显示服务器内部错误 | 检查服务器部分代码，重点查看负责该部分功能模块代码的业务逻辑以及返回数据的格式。 |
| 登陆 | 无法获取小程序用户的唯一标识 | 排查后端处理code部分关于小程序的开发者信息是否正确以及检查用户是否过长时间未登陆导致信息失效。 |
| 无法获取文章信息 | 用户查看收藏文章，无法正确显示 | 查看后端是否对数据库进行了误操作，删除了文章内容和记录。 |
| 无法爬取文章资源 | 启动爬虫程序，无法获取任何文章信息 | 可能是访问过于频繁被网站禁止访问了，可以通过更换ip池的ip来访问。 |

### 5.2 补救措施

通过对以上的错误信息进行分析，我们可以针对系统进行一些优化：

1、数据库数据交互错误：这类错误主要是因为开发人员对数据库进行了误操作或者是功能模块内部对数据进行了操作但是开发人员没有意识到，对此， 我们应该定期对数据库进行备份，对进行了数据库相关操作的模块进行严谨的测试。

2、请求错误：主要是前后端的开发人员之间对于接口的定义以及数据形式没有协商好，需要细化接口设计。

3、其他操作错误：针对用户因为不熟悉系统而造成的错误，我们可以通过撰写详尽的使用手册来规避。

4、其他不可知的程序错误：比如，原本网站突然不服务了，或者被禁掉了。这种不可预知的错误可以通过顶起维护后端来规避，或者在程序中设置提示，在发生异常的时候，设置自动消息提示。

### 5.3 系统维护设计

随着文章爬取数量的不断增加，数据库势必会越来越庞大，在爬取的过程中，可能会产生一些无效文章信息，以及脏数据。人工检查成本比较高，因此采取后端维护人员定期对数据进行维护备份的方案。

前端由开发人员不断根据技术更新进行升级完善。