

一、项目介绍 [徐超信,潘淼森]

h1

背景

h2

本项目旨在帮助用户记忆和熟悉大学阶段的英语词汇学习,解决日常的查词功能

- 当下的词典软件功能很丰富,但是高频使用的功能为数不多,
- 而且充斥着各种广告和产品推销
- 软件体积较大,运行开销不小,想要流畅的使用软件,需要用户有较好的设备
- 千方百计诱导用户充值vip,添加各种限制和不必要的信息,分散用户的精力

我们设计的这套英语学习系统,希望能够帮助用户更加轻松愉快的学习英语,包括:

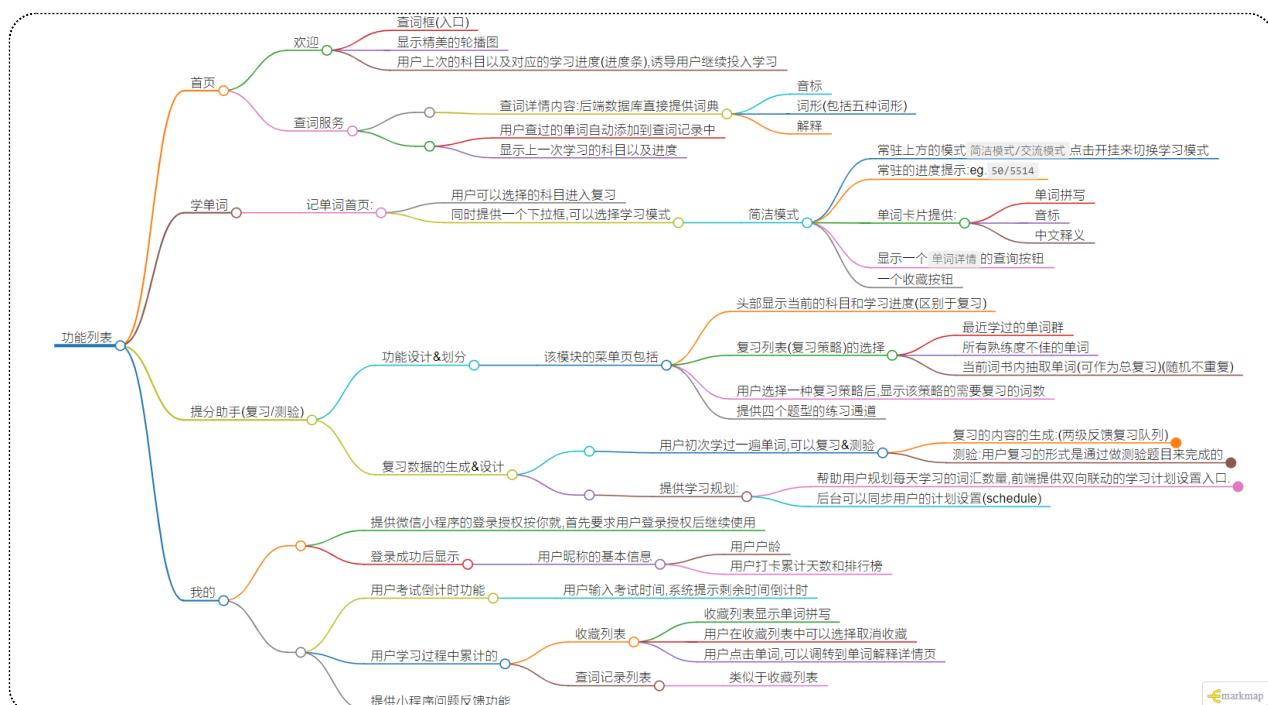
- **cet4/cet6/考研英语** 的词汇记忆学习/复习系统
- 用户可以分享自己对于词汇的记忆方法和技巧,互助学习
- 提供常规的词汇查询功能,用户可以通过输入英文单词来查询单词的基本信息(音标/词形/释义)
- 同时提供模糊查词的功能,帮助用户减轻查词过程中 **摩擦感**
- 同时提供基本的形近词推荐,帮助用户集中记忆相似单词
- 记录用户的使用痕迹和习惯,帮助用户定制学习计划,检验学习效果,集中学习专注度,拒绝分心

项目需求分析

h2

思维导图

h3





首页 h3

欢迎 h4

- 查词框(入口)
- 显示精美的轮播图
- 用户上次的科目以及对应的学习进度(进度条),诱导用户继续投入学习

查词服务 h4

- 查词详情内容:后端数据库直接提供词典
 - 音标
 - 词形(包括五种词形)
 - 解释
- 用户查过的单词自动添加到查词记录中
- 显示上一次学习的科目以及进度

学单词 h3

- 记单词首页:
 - 用户可以选择科目进入学习(刷单词卡片)
 - 同时提供一个下拉框,可以选择学习模式
 - 简洁模式
 - 常驻上方的模式 **简洁模式/交流模式** 点击开挂来切换学习模式
 - 常驻的进度提示:eg. **50/5514**
 - 单词卡片提供:
 - 单词拼写
 - 音标
 - 中文释义
 - 显示一个 **单词详情** 的查询按钮
 - 一个收藏按钮

- 交流模式(引入其他用户的一些统计数据)
 - 基本和简洁模式一致,但包括:
 - 提供批注的发送和查看功能
 - 显示所有用户对该单词的平均掌握程度

提分助手(复习/测验) h3

- 该模块的菜单页包括
 - 头部显示当前的科目和学习进度(区别于复习)
 - 复习列表(复习策略)的选择
 - 最近学过的单词群
 - 所有熟练度不佳的单词
 - 当前词书内抽取单词(可作为总复习)(随机不重复)
 - 用户选择一种复习策略后,显示该策略的需要复习的词数
 - 提供四个题型的练习通道

复习数据的生成&设计 h4

- 用户初次学过一遍单词,可以复习&测验
 - 复习的内容的生成:(两级反馈复习队列)
 - 最近学过的内容(譬如指定时间24小时)
 - 如何判断时间:过去24小时见过的单词(刷卡片学习单词时会刷新相关属性,后端会完成响应操作)
 - 所有熟练度小于特定值的单词(该标准参考测验的答题情况,来量化熟练度)
 - 测验:用户复习的形式是通过做测验题目来完成的
 - 题目的交互形式主要交由前端来落实,而数据反馈会同步到后端
 - 系统的答题模块提供了若干种题型
 - 根据中文意思选择单词
 - 根据音标以及意思提示拼写整个单词
 - 根据提示,为单词字母填空
 - 如果用错答,那么系统会将该用户对于错单单词的熟练度值-1
 - 如果用户正确答题,那么响应的将熟练度+1
- 提供学习规划:
 - 帮助用户规划每天学习的词汇数量,前端提供双向联动的学习计划设置入口.
 - 根据用户设定的每日任务计划数,计算出总耗时(天数)

- 比如用户希望在多少天内完成,那么每天任务量是多少词,
- 后台可以同步用户的计划设置(schedule)

我的 h3

- 提供微信小程序的登录授权按你就,首先要求用户登录授权后继续使用
- 登录成功后显示
 - 用户昵称的基本信息
 - 用户户龄
 - 用户打卡累计天数和排行榜
- 用户考试倒计时功能
 - 用户输入考试时间,系统提示剩余时间倒计时
- 用户学习过程中累计的
 - 收藏列表
 - 收藏列表显示单词拼写
 - 用户在收藏列表中可以选择取消收藏
 - 用户点击单词,可以调转到单词解释详情页
 - 查词记录列表
 - 类似于收藏列表
- 提供小程序问题反馈功能

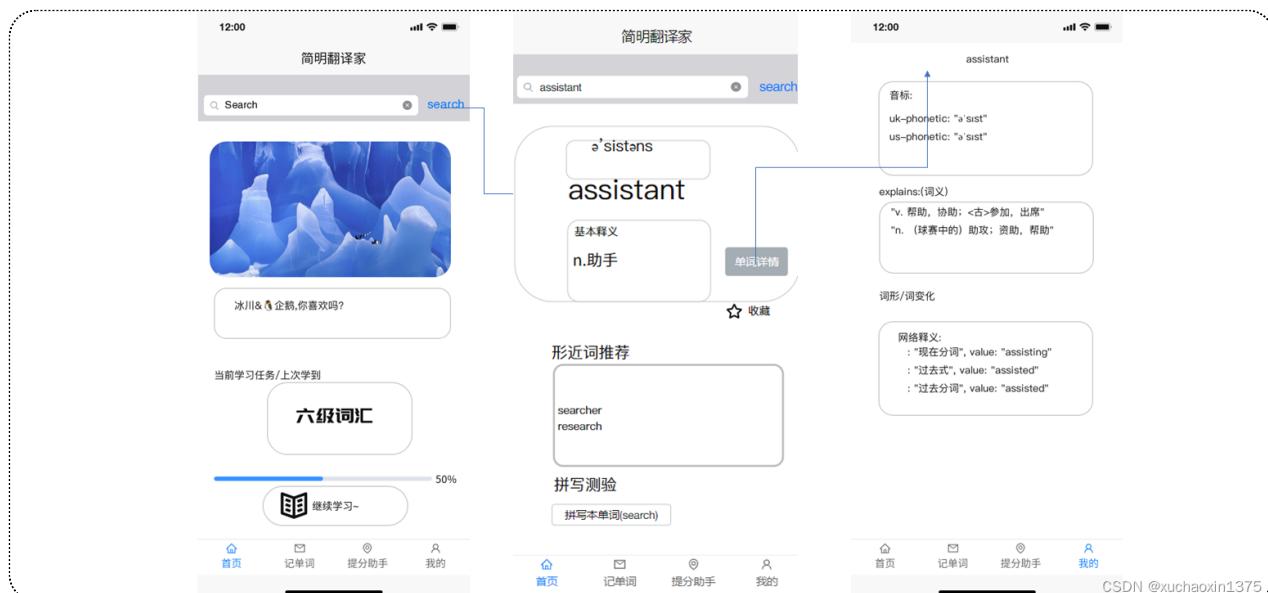
计划和分工 h2

介绍大致的开发计划以及每个人的分工。

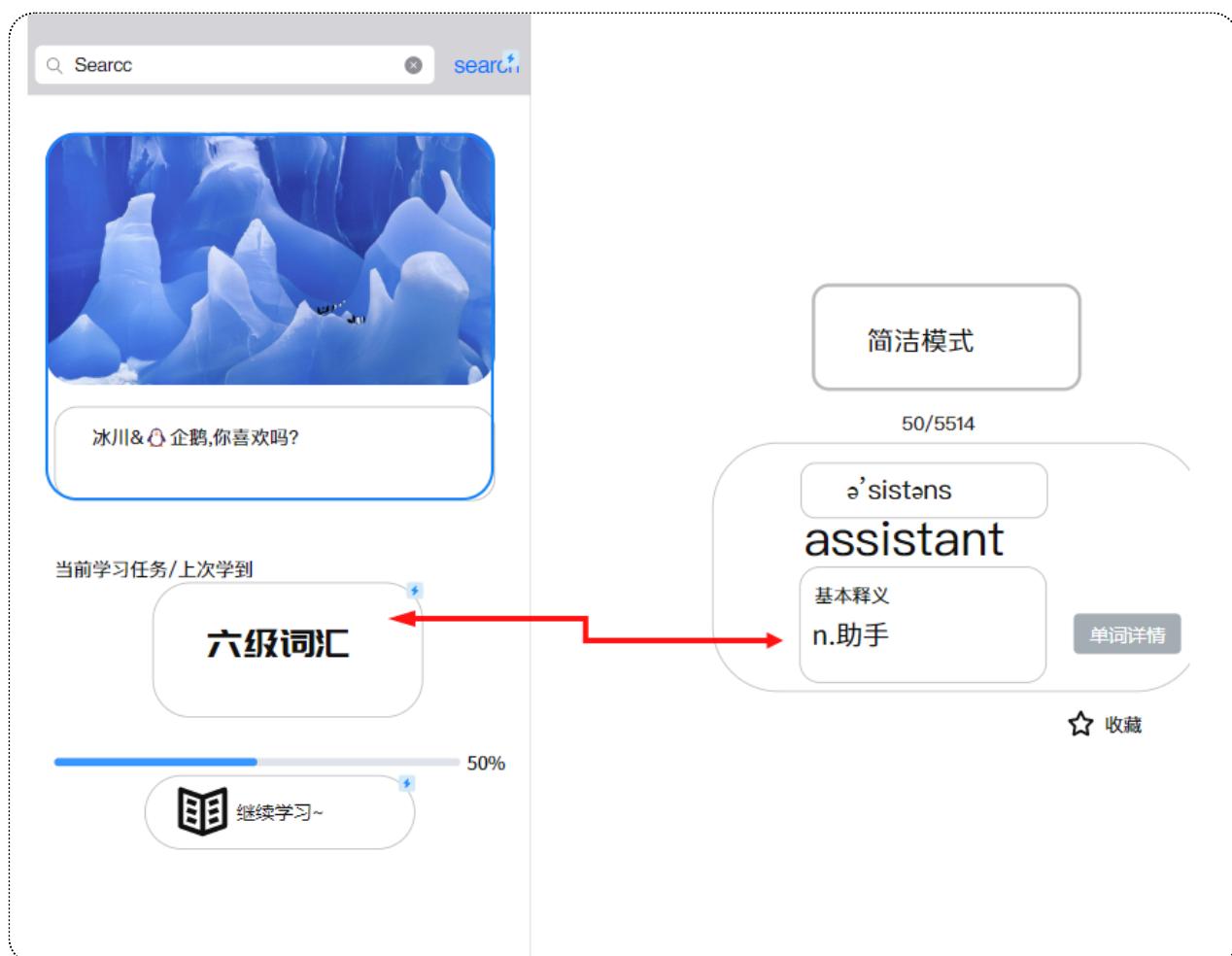
- 徐超信:
 - 原型设计和功能设计
 - 数据库设计
 - 后端开发接口开发与测试
 - 后端服务部署
 - 文档编写(主体)
- 潘森森:
 - 前端小程序的开发与测试
 - 文档编写(前端)

二、界面原型设计 [徐超信] h1

结合上述功能设计,我们将原型设计为对应的四个模块,采用 墨刀工具进行设计



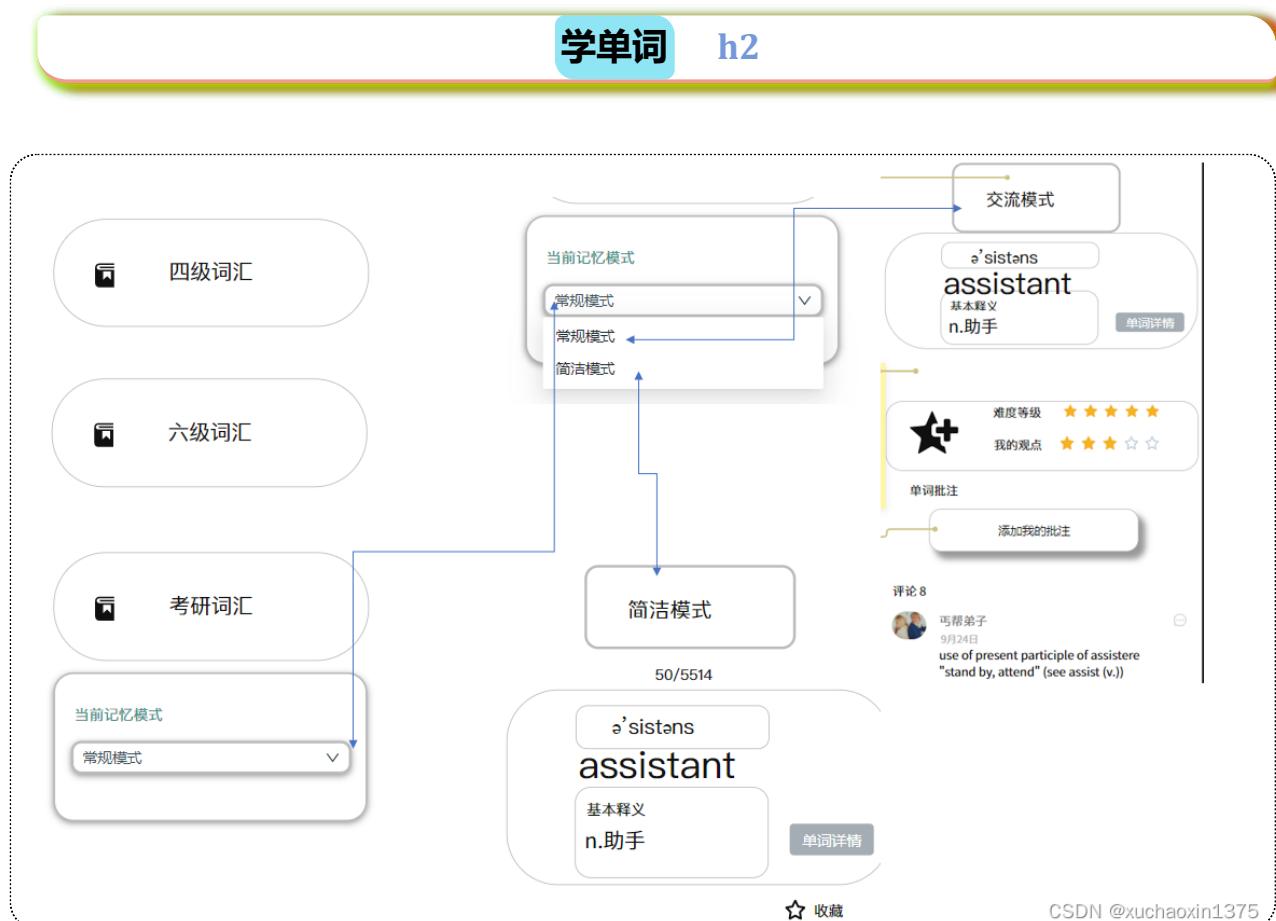
- 用户可以在首页提供的查词框中输入单词进行查词
- 查词框下面是一个轮播图,获取通过bing的图片接口,获取精美图片,为学习带来一点视觉上的享受
- 我们将单词的解释分为两层,第一层仅仅提供单词的音标和简单的解释,这一般能够满足主要的需求;此外,我们在第一层中配置了一个进一步查询单词的词形变化的按钮,点击该按钮,会跳转到第二层,这一层提供了更详细的解释,包括单词的词性等等
- 此外,还提供了收藏该的单词的按钮,点击该按钮,会将该单词添加到收藏列表中



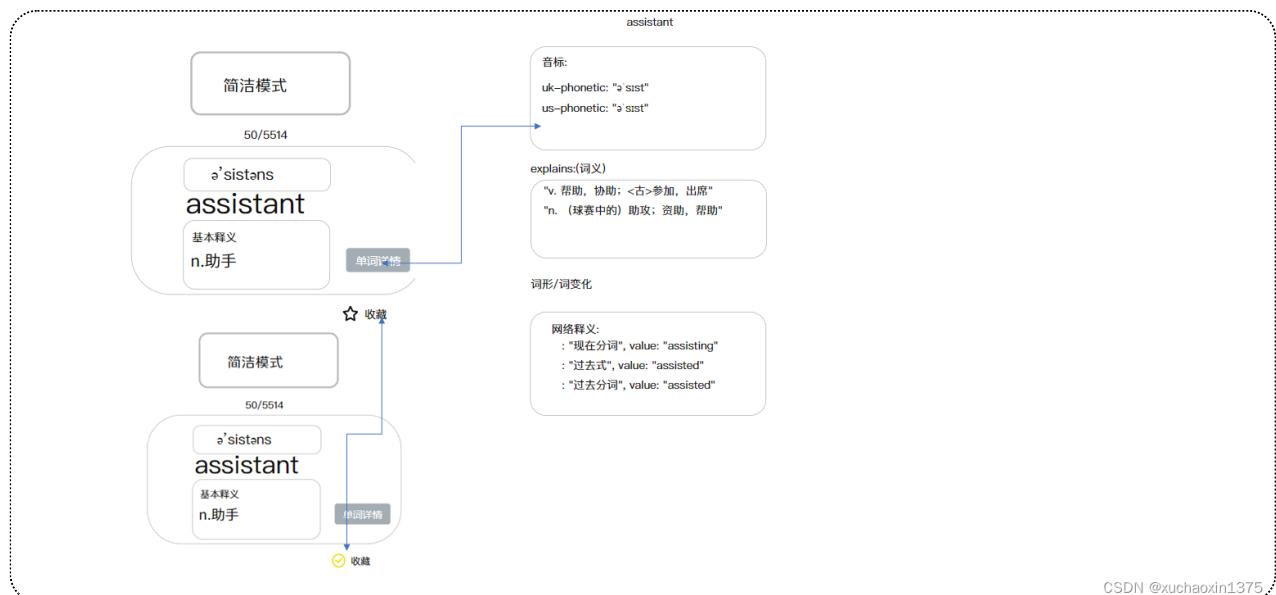
- 轮播图下面安排了用户当前的学习科目和学习进度,用户点击继续学习,便可以进入学习模式



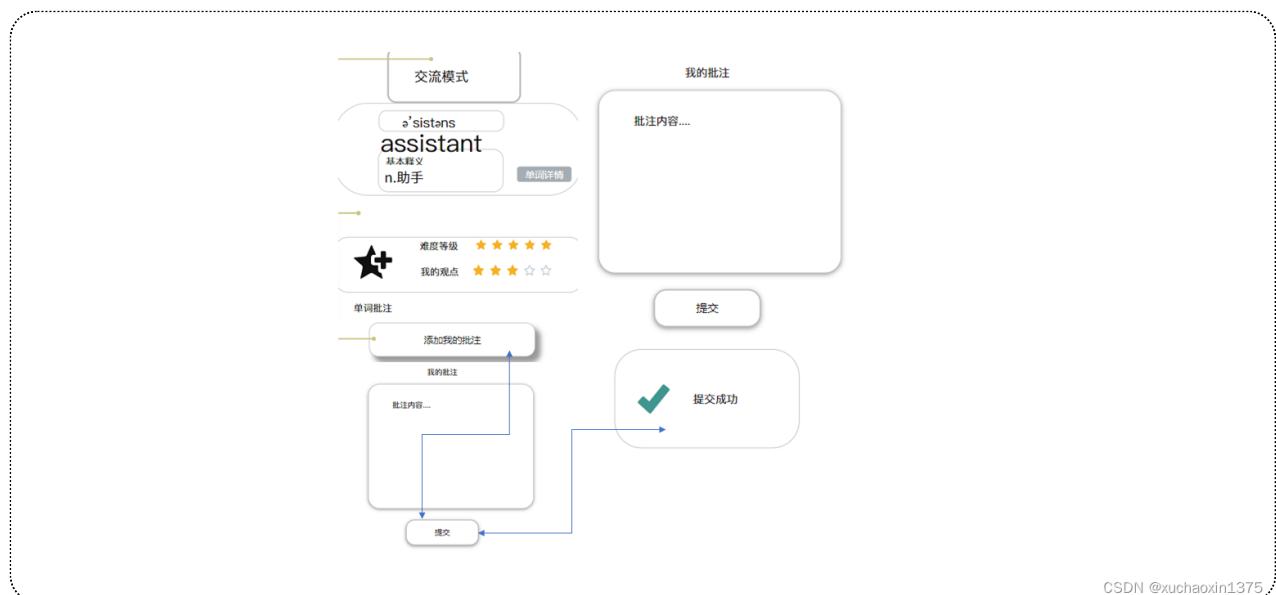
- 对于拼写错误的单词,后台会尝试通过匹配算法推荐一些形近词
 - 对于记忆不清的单词来说,这会很有用,用户也可以利用该接口查找形近词
 - 事实上,后端可以提供正则匹配/通配符等高级功能(尽管已经很少用到了)



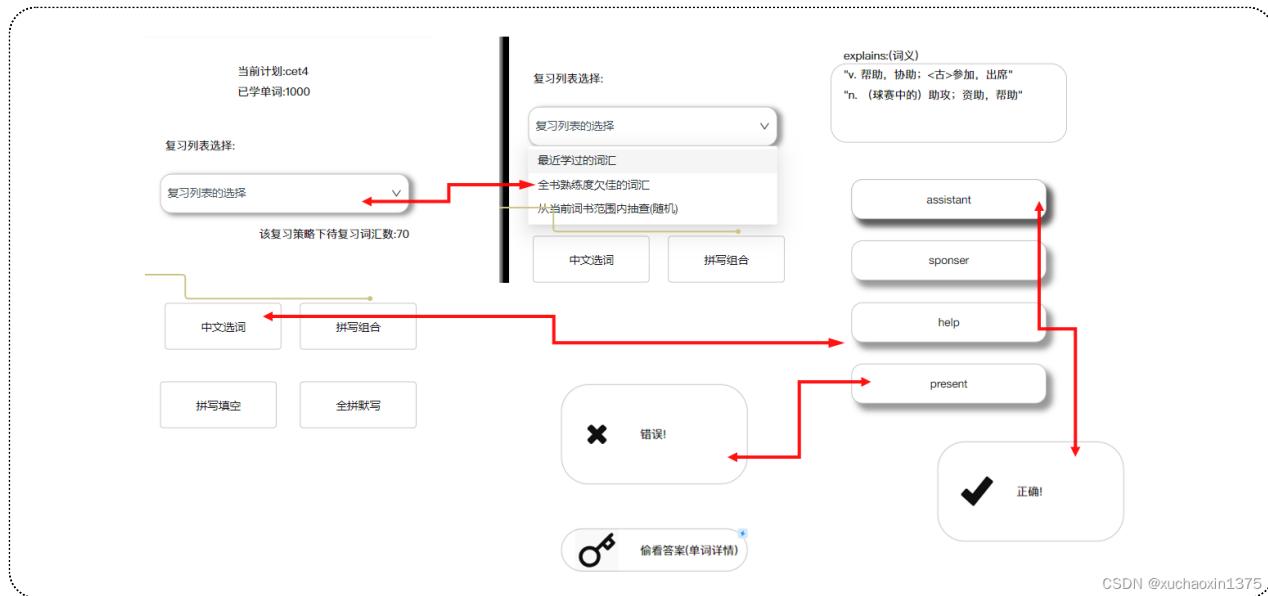
- 学单词模块,也就是本应用的核心模块
- 用户可以在该模块的主菜单页选择记忆模式(包括简洁模式和交流模式(也叫常规模式))
 - 默认的,记忆模式是常规模式
- 然后选择自己的考试类型(对应的词书)
- 其中,简洁模式包含内容较少,只有音标和基本的解释,已经一个查询单词详情解释的按钮和收藏按钮
- 而另一个模式(交流模式中),除了包含简洁模式中的相关功能,还提供了基于后台数据分析的 所有用户平均熟练度指标 (也被称为 难度等级)
- **我的观点** 则是反映本人的当前对于该词汇的熟练度



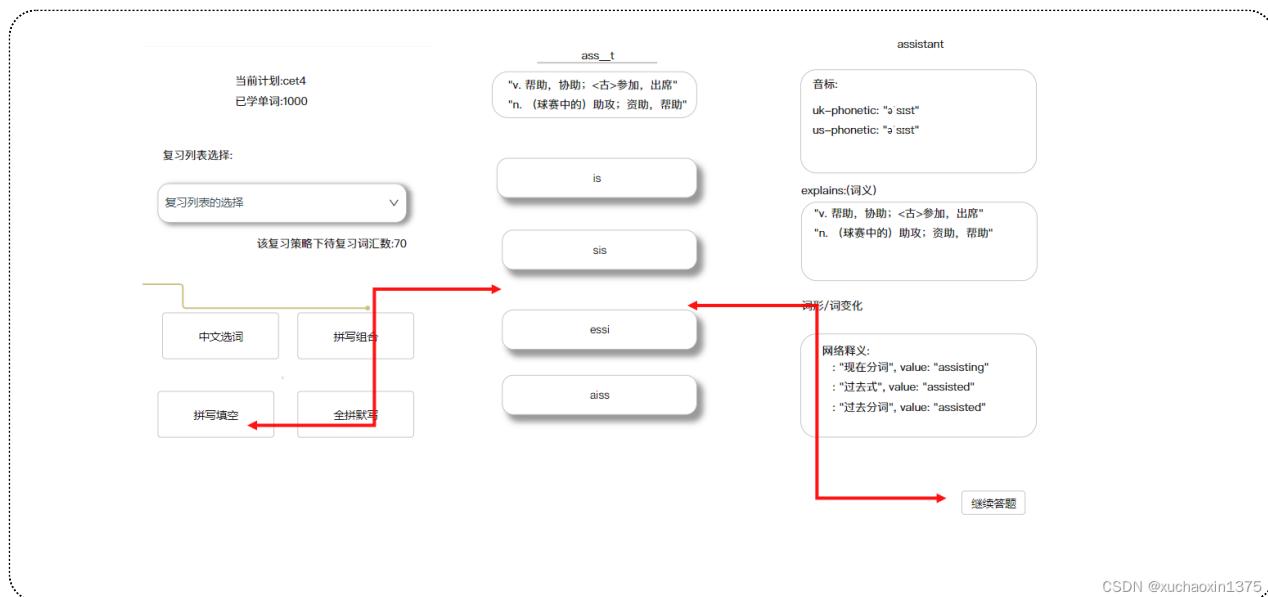
- 这是用户点击单词详情和收藏后分别的出现的响应结果



- 这是用户在交流模式下,提交自己的记忆技巧(简称为 批注)
- 提交成功则反馈一个 提交成功 的标识给用户



- 复习&测验也是本应用的主要功能,能够帮助用户检查自己的记忆效果(掌握程度),帮助用户对自己的学习成果有更加客观的把握
- 我们提供了多样化的复习策略和题型,包括中文选词,拼写组合,拼写填空和全拼默写
- 对于答错的题目,页面会切换到正确答案的解释页面
- 对于答对的题目,页面会切换到下一题
- 注:问题提交答案的方式:
 - 用户选中一个选项后,(被自动提交),后台自动判断正确性,根据正确性切换对应的页面
 - 上述流程表示的是中文选词的答题过程

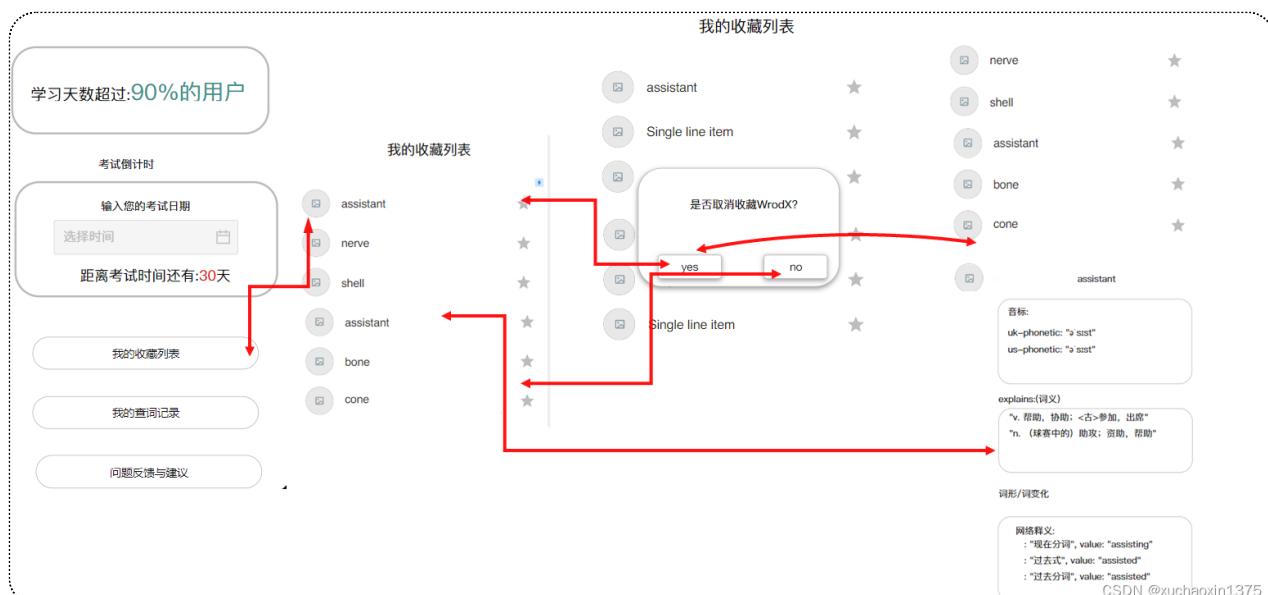


- 这是拼写填空题型下的答题过程

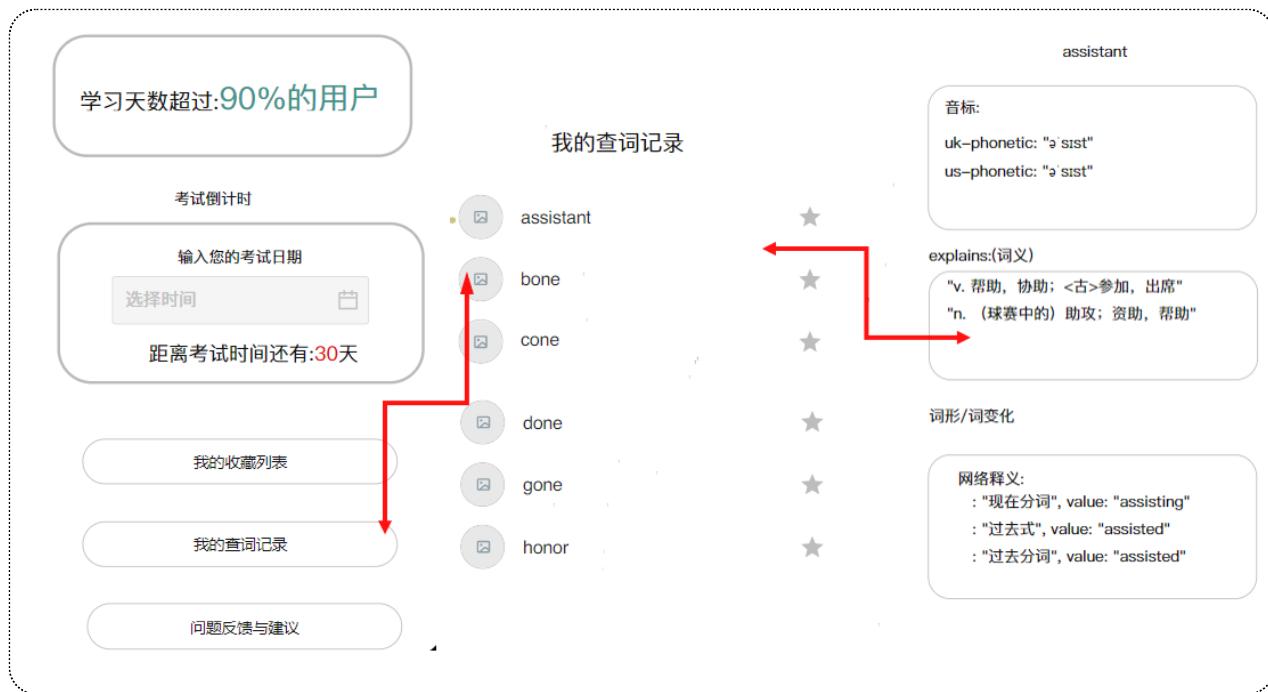
我的 h2



- 这是 **我的(用户中心)** 模块, 用于借助于微信平台的登录授权功能方便的注册登录到本系统
 - 后台通过获取微信提供的信息头像和昵称的信息创建一个用户记录
 - 该记录也作为登录状态保持(session)的value
 - 后台将会凭借session来判断和区别用户
- 登录成功后, 小程序拉取必要的同步数据, 并且做一定的数据计算和转换, 得到签到天数, 户龄
- 还包括考试倒计时/单词收藏列表和查词记录/反馈与建议的提交入口



- 用户点击我的->收藏列表, 可以看到之前做过的单词收藏,(这些单词可能是用户自认为容易混淆意思/难以拼写/品读正确的单词)
- 用户点击某个条目后, 可以跳转到响应的词典解释页面
- 用户点击星号 **star**, 可以取消掉对某个单词的收藏, 程序会向用户发送一个确认询问, 当用户确认取消, 才真正将对应的单词从收藏列表中移除, 否则, 操作被取消



- 这是进入 [单词搜索记录](#) 的流程, UI和操作逻辑基本和 [收藏列表](#) 一致

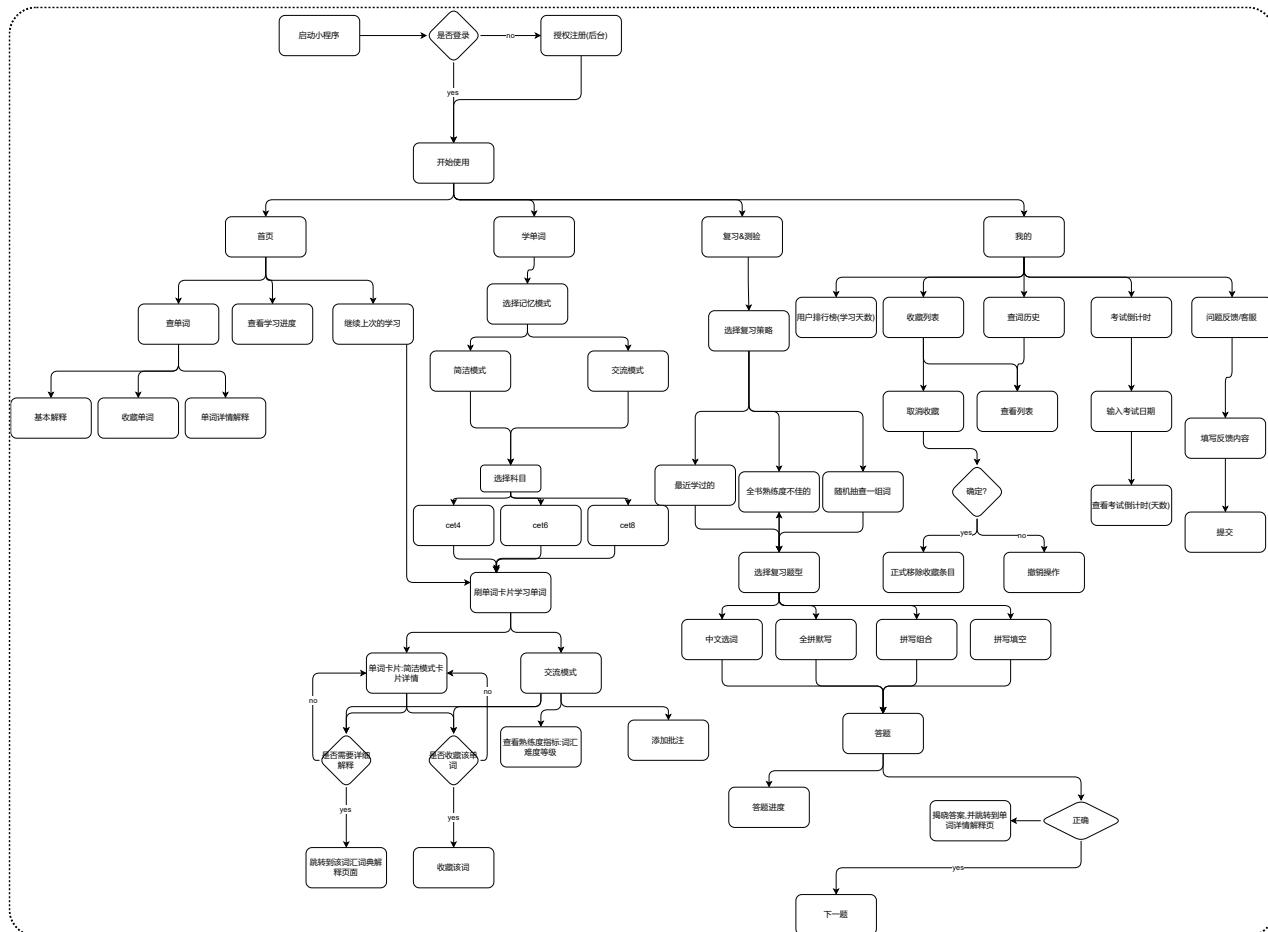
墨刀在线预览(可交互)

- <https://modao.cc/app/Zq2TY5o8rd1a7cROgqSHwe> 《EnglishLearningAsistant》
(页面附带多种状态)
- <https://modao.cc/app/xg43top2raoiorN4gHVgF> 《ELA_morePages》

三、系统架构设计 [徐超信]

前端 h2

- 前端我们分为四个功能模块
- 第一个模块是工具性模块,提供查词功能和形近词推荐功能
- 第二个模块是学单词模块(核心),并且具有两种模式可供选择,可以满足用户不同的学习风格
- 第三个模块是承接第二个模块,也是核心模块,用户可以复习和检测自己的学习效果;并提供了多样的复习策略和复习题型,也能更加全面的检测对单词的掌握情况
- 第四个模块是用户中心,属于不太常用但又不可或缺的模块,用户可以通过本模块获取自己的总体的学习情况和学习痕迹(打卡天数/收藏列表/查词记录列表/...),用户也是在该模块中反馈问题给程序后台
- 程序操作逻辑如下



后端 h2

- 后端对应前端,创建了4个功能模块,每个功能模块中在进一步细分

- 为了实现灵活性,独立性,使用前后端分离的方式渐渐称为主流,本项目中,我们采用前后端分离的开发模式,并且借助于apifox做接口设计和对接,前后端可以有自己实际开发进度
- 我们奉行 **api first** 的开发方式,促进前后端的进一步分离
- 后端采用Python/Django技术实现数据管理,用户登录与信息同步等功能

数据库 h2

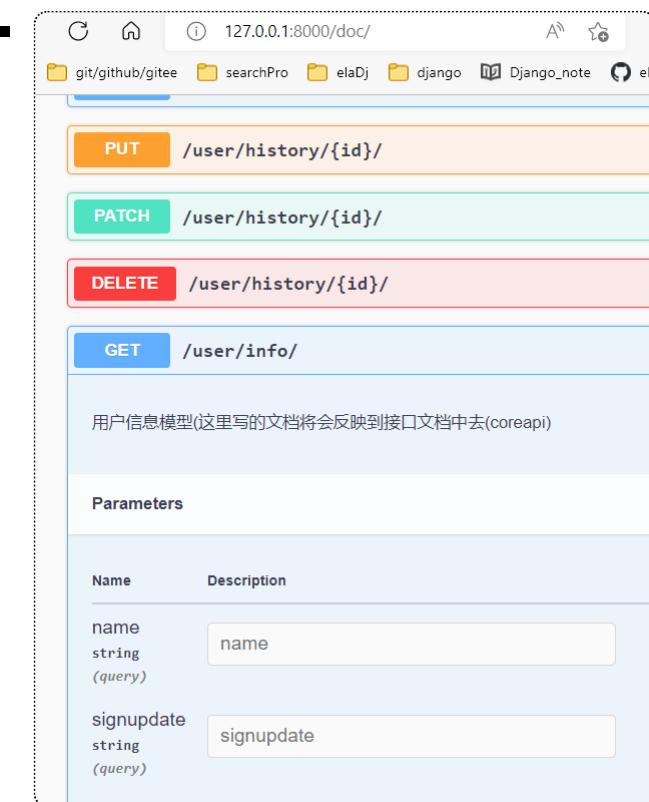
- 数据库采用免费的关系型数据库mysql,该数据库足够流行(意味着它经过了足够多的考验),完全可以胜任我们的本次项目
- 除了数据库软件本身的功能足够,我们本身已有的数据库知识也主要是关系型数据库的理论,因此最终采用mysql来提供数据管理服务

api h2

api设计是项目功能的重点,良好的接口设计有利于提高开发效率,节约沟通成本,提供可维护性

本项目的所有api都统一在apifox上设计,包括指定参数和响应,编写mock来实现前后端开发,借助于mock,前后端都有所参照,可以更加灵活的开发,项目的api总体符合restful的设计理念,具有简洁明了的特点

- 此外,后端还提供了基于swagger的文档,前端即使不查看后端代码,也可以对后端提供的接口有所了解
- 123.56.72.67:8000/doc/



四、API设计 [徐超信] h1

- 这部分主要是API的设计,分模块进行介绍,并通过Apifox介绍API的设计理念,使用、测试方法等。
- 用列表和文档对所有的API进行详细的列举和描述。

api风格与设计理念 h2

我们采用流行的RESTful api 设计风格,改善我们的api开发效率和规范性

- RESTful API是目前比较成熟的一套互联网应用程序的API设计理论。

- 访问一个网站,就代表了客户端和服务器的一个互动过程。在这个过程中,势必涉及到数据和状态的变化。

互联网通信协议HTTP协议，是一个无状态协议。这意味着，所有的状态都保存在服务器端。因此，如果客户端想要操作服务器，必须通过某种手段，让服务器端发生“状态转化”（State Transfer）。而这种转化是建立在表现层之上的，所以就是“表现层状态转化”。

客户端用到的手段，只能是HTTP协议。具体来说，就是HTTP协议里面，四个表示操作方式的动词：GET、POST、PUT、DELETE。它们分别对应四种基本操作：GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用来更新资源，DELETE用来删除资源。

- 我们在开发项目的api的过程中，尽可能地采用RESTful理念，充分利用了http协议中的四个常用动词来设计api，客户端通过四个HTTP动词，对服务器端资源进行操作，实现“表现层状态转化”
- RESTful API最好做到Hypermedia，即返回结果中提供链接，连向其他API方法，使得用户不查文档，也知道下一步应该做什么。我们的后端的四个模块的基础路由提供了类似的功能，帮助api的使用者更快了解后端的api功能组织

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "info": "http://127.0.0.1:8000/user/info/",
    "register": "http://127.0.0.1:8000/user/register/",
    "history": "http://127.0.0.1:8000/user/history/",
    "star": "http://127.0.0.1:8000/user/star/"
}
```

- 尽管RESTful是一个很好的理念，但是在开发过程中，发现有少量的api较难通过四个动词来贴切地描述api的实际用意，因此，我们结合实际需求，对少数api做了折衷处理

详细的api文档参看附件

附件中的api文档是通过apifox导出

五、数据库设计 [徐超信]

持久化数据

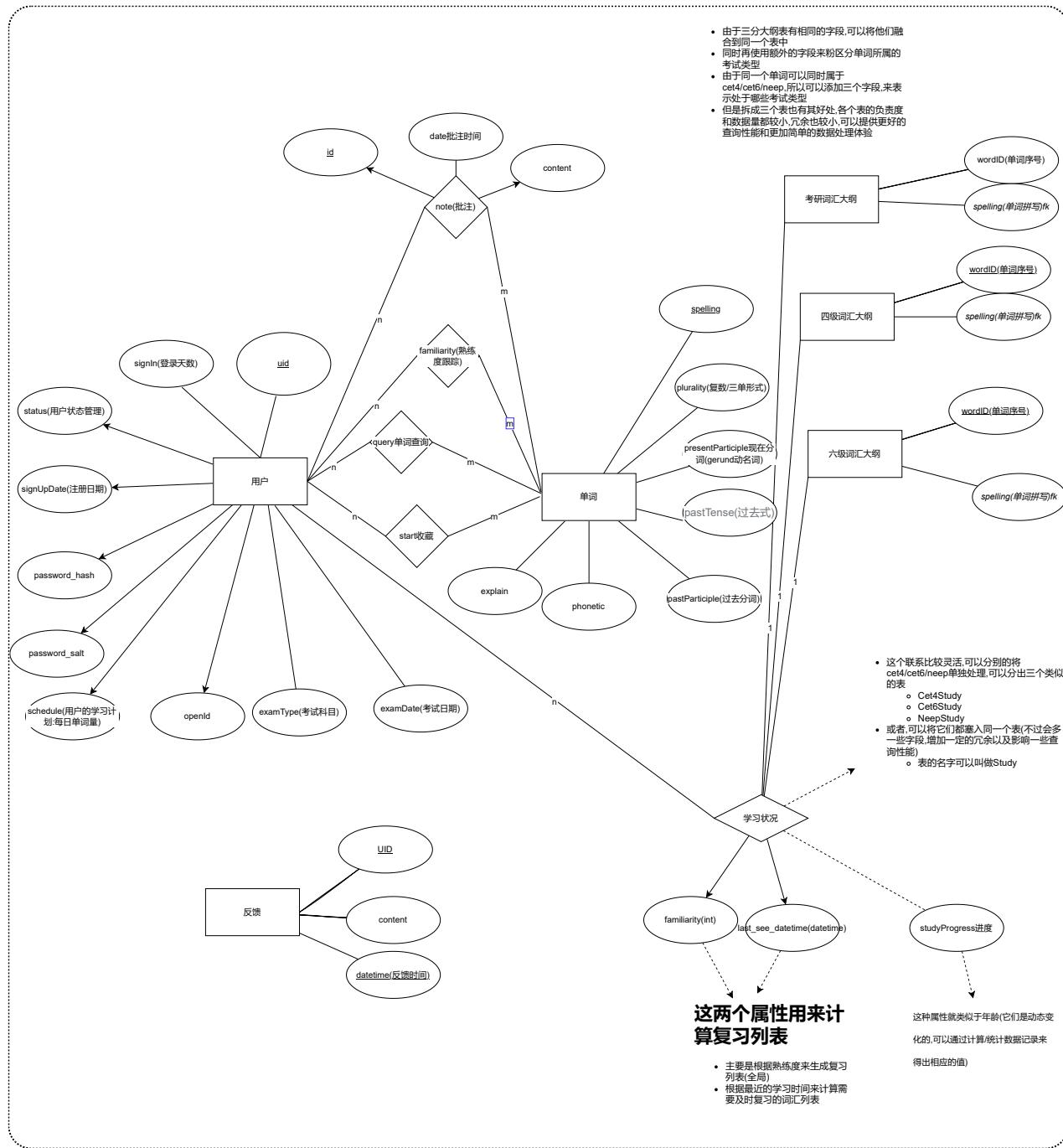
- 本项目中，需要持久化的数据包括
 - 词典（单词的各个属性）
 - 三种考试科目考纲词汇列表
 - 用户信息

- 用户的学习记录和学习情况
- 用户的问题反馈记录
- 用户的收藏
- 用户的查询记录
- 用户的评论记录(批注)
- 等其他附加信息

数据库的选择 h2

- 数据库方面,我们希望采用具有如下特点的数据库软件:
 - 当下流行的
 - 易用的
 - 参考资料丰富的
 - 免费的
 - 跨平台的
- 由于我们有关系型数据库的理论基础,对此会更加熟悉一些,因此我们考虑在关系型数据库中选择一款数据库软件
- 在具体选择数据库的时候,我们考虑过
 - postgres数据库(先进,功能完备,django首推的生产环境数据库)
 - mysql数据库(小巧,速度快)
- 两者都是可以免费使用,但基于资料的丰富程度,和现有的教程,我们选择了mysql

ER关系图 h2



创建数据库时的考虑 h2

- 将不同考试类型的词汇大纲怎么处理
 - 放在不同表
 - 字段最少(结构最简单)
 - 但是需要管理的表数量变多,而且结构重复,编写接口的时候有一定重复
 - 放在一张表
 - 为每个单词增加字段
 - 一个字段
 - 多个值:一个单词可能属于多个考试考纲中的词汇)(4,6,8,46,68,...)

- 字段单个值,拆分为多条记录:在一条记录的同一个字段有个取值时,将其拆分为多条记录加以存放
- 多个字段:
 - 分别设置cet4/cet6/neep字段,如果某个单词属于该字段,那么为其取bool值true(1)
- 主属性采用原主键+多值的那个字段构成多字段主键
- 收藏表和搜索记录表是否分开放
 - 两个表结构相似
- 但是考虑到各自的可独立扩展性,本项目将其拆分

基于如下考虑,我决定不拆分word表(将词形(诸如past tense/presentParticiple/pastParticiple))从而提高查询效率

- 复杂表的优缺点

- 优点:包含全部字段的表,在查询时避免了连表查询,程序处理起来很方便,有时候某些表会加进一些冗余字段,也就是为了避免连表查询。查询的效率方面有优势。
- 缺点:如果字段里面有大字段(text,blob)类型的,而且这些字段的访问并不多,这时候放在一起就变成缺点了。

- 简单表(字段简单的表)

- 更可能符合更高的范式
- MYSQL数据库的记录存储是按行存储的,数据块大小又是固定的(16K),**每条记录越小**,存储块内存储的记录就越多。此时应该把大字段拆走,这样**应付大部分小字段的查询时,就能提高效率**。
- 当需要查询大字段时,此时的关联查询是不可避免的,但也是值得的。
- 拆分开后,对字段的UPDATE就要UPDATE多个表了
- 用于查询和展示的不建议分表,
- 若只是存储数据,可以考虑分表。

各个模块下的表设计 h2

User用户模块 h2

用户表

h3

![image-20220604160924400](<https://s2.loli.net/2022/06/04/EjDpu7TSYJqMiF8.png>)

```

1 +-----+-----+-----+-----+-----+
2 | Field      | Type       | Null | Key | Default | Extra        |
3 +-----+-----+-----+-----+-----+
4 | uid        | int(11)    | NO   | PRI | NULL    | auto_increment |
5 | name       | varchar(250) | NO   |     | NULL    |               |
6 | signIn     | int(11)    | NO   |     | NULL    |               |
7 | examType   | varchar(1)  | NO   |     | NULL    |               |
8 | examDate   | date       | NO   |     | NULL    |               |
9 | signUpDate | date       | NO   |     | NULL    |               |
10 | openid     | varchar(150) | YES  | UNI | NULL    |               |
11 | password_hash | varchar(250) | NO   |     | NULL    |               |
12 | password_salt | varchar(250) | NO   |     | NULL    |               |
13 | status     | int(11)    | NO   |     | NULL    |               |
14 | schedule   | int(11)    | NO   |     | NULL    |               |
15 +-----+-----+-----+-----+-----+
16 11 rows in set

```

查词记录

h3

```

1 +-----+-----+-----+-----+-----+
2 | Field      | Type       | Null | Key | Default | Extra        |
3 +-----+-----+-----+-----+-----+
4 | id         | bigint(20) | NO   | PRI | NULL    | auto_increment |
5 | spelling   | varchar(25) | NO   |     | NULL    |               |
6 | user_id   | int(11)    | NO   | MUL | NULL    |               |
7 +-----+-----+-----+-----+-----+
8 3 rows in set

```

单词收藏表

h3

```

1 +-----+-----+-----+-----+-----+
2 | Field      | Type       | Null | Key | Default | Extra        |
3 +-----+-----+-----+-----+-----+
4 | id         | bigint(20) | NO   | PRI | NULL    | auto_increment |
5 | spelling   | varchar(25) | NO   |     | NULL    |               |
6 | user_id   | int(11)    | NO   | MUL | NULL    |               |
7 +-----+-----+-----+-----+-----+
8 3 rows in set

```

反馈表

h3

- 由于不是所有用户都会做反馈,为了减少冗余,我们将反馈表单独拆分出来

```
1 | root@localhost [Sat Jun  4 20:56:41 2022 23 el4] > desc feed_back;
```

```

2 +-----+-----+-----+-----+
3 | Field   | Type      | Null | Key  | Default | Extra       |
4 +-----+-----+-----+-----+
5 | id      | bigint(20) | NO   | PRI   | NULL    | auto_increment |
6 | content | varchar(255) | NO   |       | NULL    |               |
7 | date    | datetime(6)  | NO   |       | NULL    |               |
8 | user_id | int(11)    | NO   | MUL   | NULL    |               |
9 +-----+-----+-----+-----+
10 4 rows in set

```

words(词典/记单词模块)模块

h2

单词表

h3

- 词典表

-

```

1 +-----+-----+-----+-----+
2 | Field   | Type      | Null | Key  | Default | Extra       |
3 +-----+-----+-----+-----+
4 | wid      | int(11)    | NO   | PRI   | NULL    | auto_increment |
5 | spelling | varchar(255) | NO   |       | NULL    |               |
6 | phonetic | varchar(255) | YES  |       | NULL    |               |
7 | plurality | varchar(255) | YES  |       | NULL    |               |
8 | thirdpp  | varchar(255) | YES  |       | NULL    |               |
9 | present_participle | varchar(255) | YES  |       | NULL    |               |
10 | past_tense | varchar(255) | YES  |       | NULL    |               |
11 | past_participle | varchar(255) | YES  |       | NULL    |               |
12 | explains   | longtext   | YES  |       | NULL    |               |
13 +-----+-----+-----+-----+
14 9 rows in set (0.32 sec)

```

- 字段描述:其中

- wid作为主键,表示词汇序号

- thirdpp为单词第三人称单数
- past_tense为动词过去式
- past_participle为动词过去分词
- explains作为单词中文解释

词典拼写分析(word_matcher) h3

```

1 |-----+-----+-----+-----+-----+
2 | Field | Type | Null | Key | Default | Extra |
3 |-----+-----+-----+-----+-----+
4 | id | bigint(20) | NO | PRI | NULL | auto_increment |
5 | spelling | varchar(255) | NO | | NULL | |
6 | char_set_str | varchar(26) | NO | | NULL | |
7 |-----+-----+-----+-----+-----+
8 3 rows

```

- 该表辅助模糊匹配算法的实现
- char_set_str是构成单词的字符集合,类型为字符串

单词批注 h3

```

1 |-----+-----+-----+-----+-----+
2 | Field | Type | Null | Key | Default | Extra |
3 |-----+-----+-----+-----+-----+
4 | id | bigint(20) | NO | PRI | NULL | auto_increment |
5 | content | varchar(255) | YES | | NULL | |
6 | spelling | varchar(255) | YES | | NULL | |
7 | UID | int(11) | YES | | NULL | |
8 |-----+-----+-----+-----+-----+
9 5 rows in set

```

- 原本想要使用一个 **difficalty_rate** 难度投票字段,来丰富用户的交互,但是后来发现这样不实用
- 遂采用后台统计的平均熟练度来表征应该更为准确.

- 此表用户记录用户在单词下的留言批注
- content是留言内容
- spelling是被留言的单词

词汇大纲表 h3

- 在单词(释义)表(词典)之外,我们另外建立了考试大纲词汇的索引数据库(大名单词索引)
- 有三张表使用了相同的结构

```

1 |-----+-----+-----+-----+-----+
2 | Field | Type | Null | Key | Default | Extra |

```

```

3 +-----+-----+-----+-----+
4 | wordOrder | int(11) | NO | PRI | NULL | auto_increment |
5 | spelling | varchar(255) | NO | | NULL |
6 +-----+-----+-----+-----+
7 2 rows in set

```

- 字段wordOrder用来充当该表的主键
- 原本打算使用spelling直接作为主键,然而,mysql默认不区分大小写,会导致某些词汇发生冲突;
- 虽然可以配置mysql强制区分大小写,但是Django文档指出,这可能会引发意料之外的问题,故采用了需要来作为主键

scoreImprover(复习&测验)模块 h2

学习记录表 h3

本表将所有用户的所有考试类型的学习记录都整合起来,方便管理和分析数据

```

1 +-----+-----+-----+-----+
2 | Field | Type | Null | Key | Default | Extra |
3 +-----+-----+-----+-----+
4 | id | int(11) | NO | PRI | NULL | auto_increment |
5 | last_see_datetime | datetime(6) | YES | | NULL |
6 | familiarity | int(11) | NO | | NULL |
7 | examType | varchar(1) | NO | | NULL |
8 | user_id | int(11) | NO | MUL | NULL |
9 | wid_id | int(11) | NO | MUL | NULL |
10 +-----+-----+-----+-----+
11 6 rows in set (0.17 sec)

```

- 根据ER图表示上看,学习记录是一个多对多的关系,对于每个科目下都可以产生一个学习记录表
- 但是表的结构上看,如果每个科目一个表,那么就会有三张结构一致的表,这显得不那么有利于后台开发
- 所为了方便管理,我们将不同的科目不同用户的学习记录聚合到一张表上并且用一个字段examType来区分每条记录时属于哪个科目下的学习记录

数据相关技术 h2

- 我们采用django框架提供的ORM来操作数据,这使得后端代码的更加具有通用性,相对于编写原生的SQL语句,使用更加通用和专业的编程语言会更加开发上的有效率优势和维护优势

- 此外,django提供了强大的数据库迁移功能,当数据模型发生变化时,数据库迁移 [migrations&migrate](#) 操作可以将模型的修改同步反映到数据库表的修改
- 数据库迁移的另一大好处是,可以和代码版本控制相互配合,实现整整的整个项目上的版本控制
- 当代码回滚到早期的版本时,数据库结构也需要回滚到相兼容的版本,否则项目可能直接无法运行起来
- 另一方面,django自带的ORM操作还比较初级,我们选用了基于Django的子框架DRF来提高数据库操作相关的编码效率和数据安全性检查,
- 特别是对于大量数据查询操作返回的结果的分页功能的实现上,使用DRF会比原生的 django分页更加合适前后端分离的项目

六、微信小程序端的实现 [潘淼森] h1

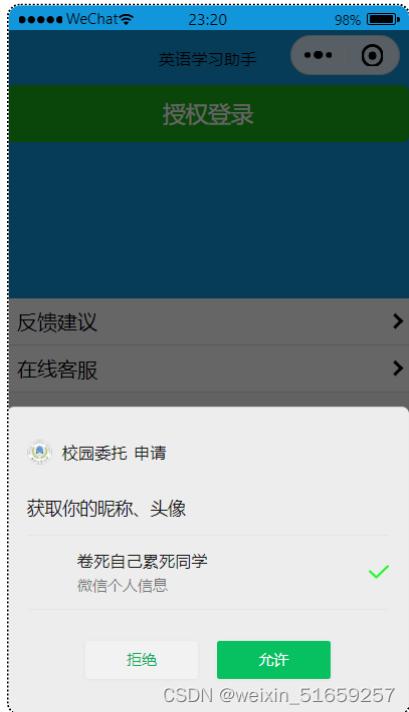
这部分分模块来描述前端的具体实现,比如:

- vant app 组件+微信小程序

6.1 登录登出功能的实现 h2

这部分是用户管理模块,如登录、注册、修改等功能的具体实现。这里应该重点将实现时考虑的因素,使用的算法以及这样做的优缺点,最后可以通过界面的截图来展示实现效果。





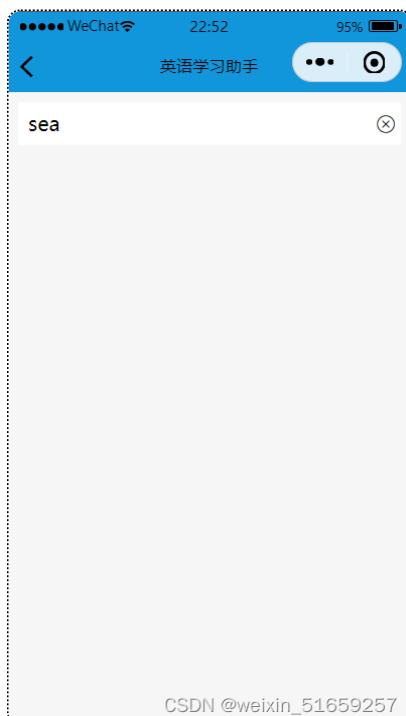
6.2 查词功能的实现 h2

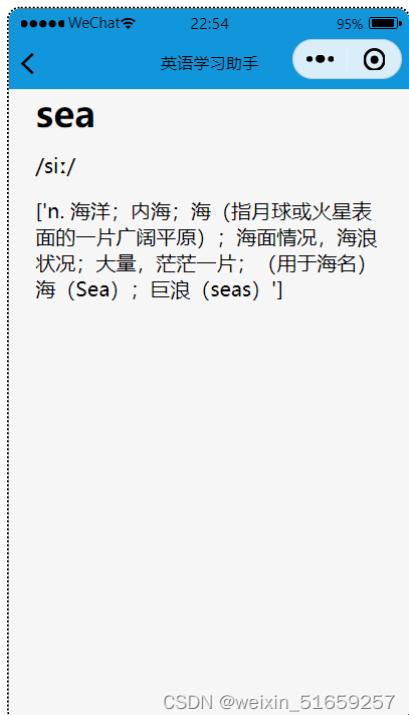
这部分是用户管理模块，如登录、注册、修改等功能的具体实现。这里应该重点将实现时考虑的因素，使用的算法以及这样做的优缺点，最后可以通过界面的截图来展示实现效果。

1.如果数据库中未找到则显示未找到相关内容



2.如果输入正确

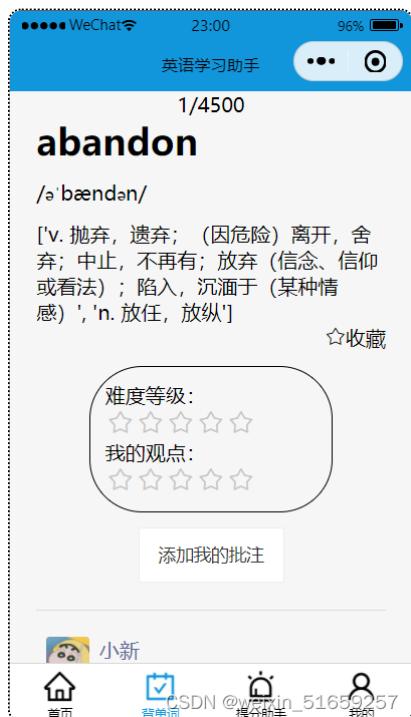






6.3 学单词功能的实现 h2

左滑右滑实现翻页，翻到第一页再往前翻会提示

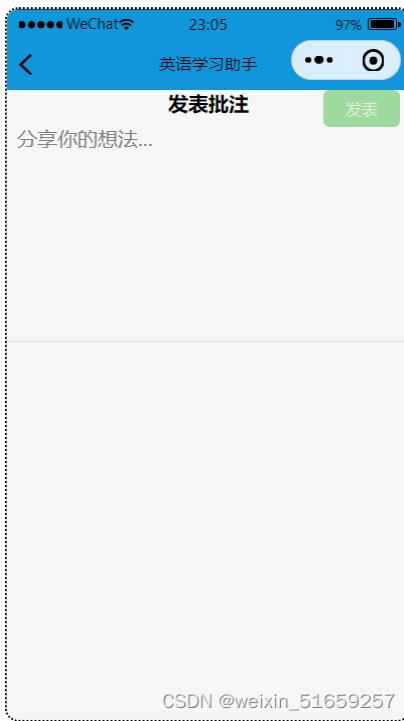


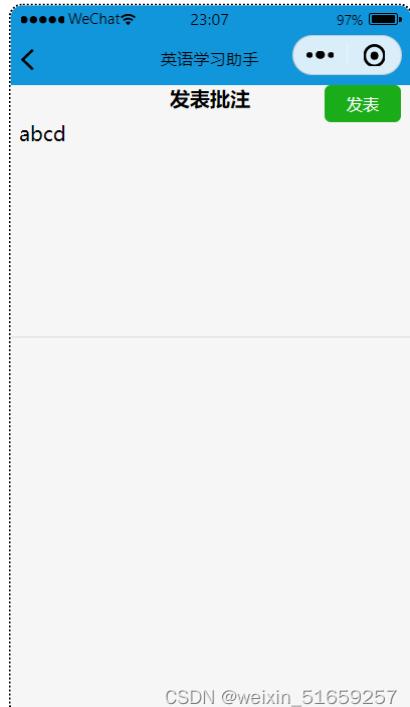


难度评价打分

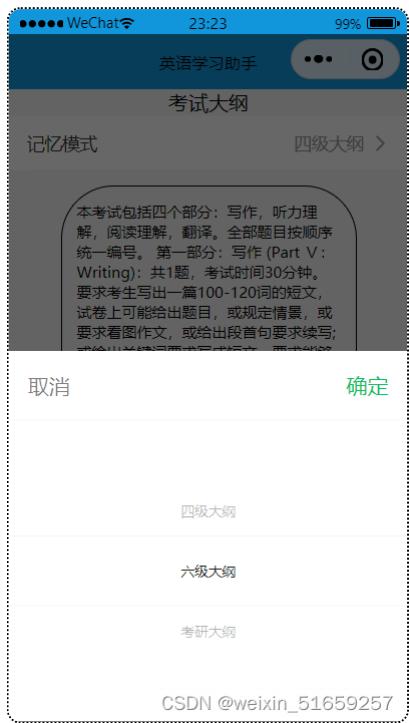
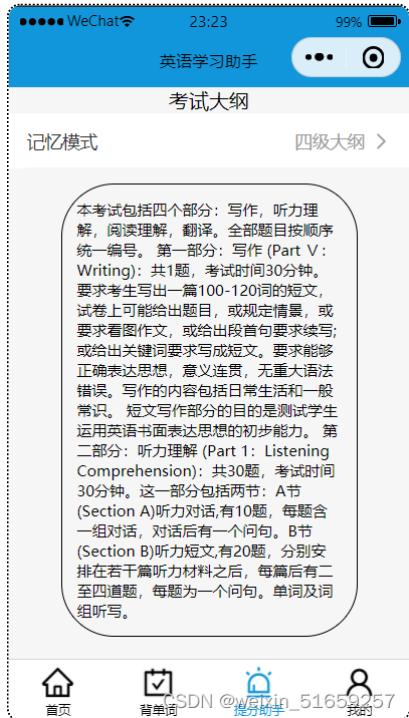


添加批注





6.4 查考纲功能的实现 h2





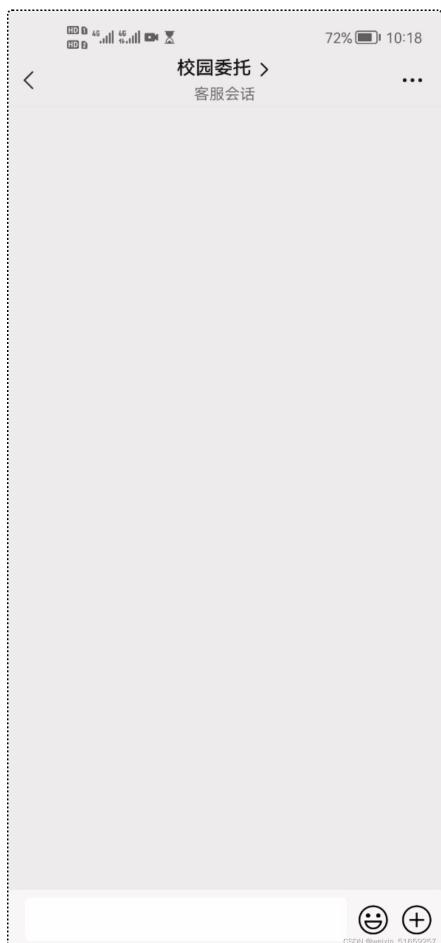
6.5 计算ddl功能的实现 h2





6.6 反馈建议&在线客服功能的实现

h2



七、英语助手后端的实现[徐超信] h1

这部分也是分模块来展示后端的实现方案。具体参见前面第六部分。

这部分分模块来描述后端的具体实现，比如：

7.1 查词功能的实现 h2

这部分是用户管理模块，如登录、注册、修改等功能的具体实现。这里应该重点将实现时考虑的因素，使用的算法以及这样做的优缺点，最后可以通过界面的截图来展示实现效果。

- 查词功能是本程序的首页模块的功能,类似于词典
- 我们通过python+pandas从有道词典的接口爬取了13k单词,每个单词都具有
 - 中文释义
 - 音标
 - 5中词形(当然,名词没有过去式,动词没有复数,这种情况下都用NULL来填充字段)
 - 他们被存储在数据中,并通过ORM编写响应的查词接口,实现查词功能

- 

7.2 学单词功能的实现 h2

h2

h2

GET

http://127.0.0.1:8000/word/{examty...}

发送

保存

删除

获取考纲词汇 (成功cet4)

Params 1

Body Headers ... </>

Path 参数

	参数名	参数值
<input checked="" type="checkbox"/>	examtype	cet4

Body

Cookie 1

Header 10

控件

...

Pretty

utf8



...

```

1  {
2    "count": 4500,
3    "next": "http://127.0.0.
4    1:8000/word/cet4/?page=2",
5    "previous": null,
6    "results": [
7      {
8        "wordorder": 1,
9        "spelling": "abandon"
10       },
11       {
12        "wordorder": 2,
13        "spelling": "abatement"
14       },
15       {
16        "wordorder": 3,
17        "spelling": "abdomen"
18       },
19     ]
20   }

```

CSDN @xuchaoxin1375

- 学单词功能的实现,首先要获取对应科目的考纲词汇列表,由于列表很长,所以提供了分页功能
- 值得一提的是,后端几乎为所有返会长列表(大数据量)的操作提供了分页参数,因此不在每个接口地反复说明
- 实现本功能用到地接口原理也比较简单,获取考纲列表地接口通过读取对应的科目地词汇数据库表,将他们分页返回
- 在搭配一个刷单词时更新最近刷过(见过面的)卡片的时间(last_see_datetime),用以提供后面计算复习单词列表的基础数据

单词平均难度指数的实现

h3

GET http://127.0.0.1:8000/word/avg-familiarity/{spelling}

发送 保存 删除

获取单词的用户平均熟练度 (有效记录:spelling=abandon)

Params 1	Body	Headers	Cookies	...	</>
Path 参数					
<input checked="" type="checkbox"/>	参数名	参数值			
<input checked="" type="checkbox"/>	spelling	abandon			

Body	Cookie	Header 10	控制台	实际请求•
Pretty	Raw	Preview	utf8	...
<pre> 1 { 2 "spelling": "abandon", 3 "avg_familiarity": 3, 4 "validity": true, 5 "msg": "查询成功" 6 }</pre>				
CSDN @xuchaoxin1375				

- 该指标是通过计算数据库中所有用户对该单词的熟练度，并计算平均值，以此来计算单词的平均难度指数
- 可以帮助用户参考准确记忆该单词的难度

7.3 复习功能的实现

GET http://127.0.0.1:8000/user/info/{pk}/review/global/{examtype}

发送 保存 删除

用户的全局复习列表(推荐复习) _separate (成功pk=4&examtype=6)

Params 2	Body	Headers	Cookies	...	</>
Path 参数					
<input checked="" type="checkbox"/>	参数名	参数值			
<input checked="" type="checkbox"/>	pk	4			
<input checked="" type="checkbox"/>	examtype	6			

Body	Cookie 1	Header 10	控制台	实际请求•
Pretty	Raw	Preview	utf8	...
<pre> 1 [2 { 3 "id": 6, 4 "last_see_datetime": "2022-06-09T04:37:37.593018Z", 5 "familiarity": 0, 6 "user": 4, 7 "user_name": "testScriptUser", 8 "wid": 2, 9 "spelling": "abide" 10 }, 11 { 12 "id": 9, 13 "last_see_datetime": "2022-05-15T14:17:59.604789Z", 14 "familiarity": 4, 15 "user": 4, 16 "user_name": "testScriptUser", 17 "wid": 3, 18 "spelling": "abnormal" 19 }, 20 { 21 "id": 12,</pre>				
CSDN @xuchaoxin1375				

- 用户通过指定科目，获取系统推荐的全书范围内的熟练度不佳的单词列表

GET http://127.0.0.1:8000/user/info/review/recently/

获取最近学过的单词列表_aggregate (成功(examtype=4unit=days&value=8))

Params 3 Body Headers Cookies ... </>

Body Cookie 1 Header 10 控制台 实际请求 •

Query 参数

参数名	参数值
unit	days
value	7
examtype	4

Pretty Raw Preview utf8 ▾

```
1 [ {  
2   "id": 6,  
3   "last_see_datetime": "2022-06-06T05:57:44.213902Z",  
4   "examtype": "4",  
5   "familiarity": 3,  
6   "user": 112,  
7   "user_name": "cxxu",  
8   "wid": 65,  
9   "spelling": "actress"  
10 },  
11 {  
12   "id": 7,  
13   "last_see_datetime": "2022-06-06T05:58:40.866844Z",  
14   "examtype": "4",  
15   "familiarity": 3,  
16   "user": 112,  
17   "user_name": "cxxu",  
18   "wid": 645,  
19   "spelling": "actress"  
20 }
```

CSDN @xuchaoxin1375

- 这是另一种复习模式:根据用户近期学习的词汇,提供复习列表
- 这里使用的是UTC时间,但是不影算法效果
- 客户端可以自行指定 **最近** 的概念,是最近1天还是最近1小时,甚至可以指定单位和浮点数
- 算法的基本原理比较简单,首先获取用户刷单词卡片的时候的时间(后端提供了响应的刷新时间的接口)
- 然后利用最近见过单词的时间和此时的时间(用户打开复习模块刷题的时刻)做时间差,当时间差处于指定范围内时,响应的记录就会被返回,用户生成复习列表
- 前端同样可以利用本接口开发丰富的复习模式,譬如允许用户输入时间范围.

GET

http://127.0.0.1:8000/improver/review/...

发送

保存

删除

随机抽查—组单词 G (sizeable) (成功(cet6))

Params 2

Body

Headers

...

</>

Body

Cookie 1

Header 10

控制

...

Path 参数

	参数名	参数值
<input checked="" type="checkbox"/>	examtype	cet6
<input checked="" type="checkbox"/>	size	55

Pretty

utf8



...

```

14 },
15 {
16     "wordorder": 228,
17     "spelling": "believe"
18 },
19 {
20     "wordorder": 291,
21     "spelling": "bright"
22 },
23 {
24     "wordorder": 316,
25     "spelling": "bus"
26 },
27 {
28     "wordorder": 366,
29     "spelling": "census"
30 },
31 {
32     "wordorder": 545,
33     "spelling":
34     "contrary"
35 }
  
```

CSDN @xuchaoxin1375

- 这是最后一种复习模式用户可以将其作为抽查对全书范围内抽取的一组词汇检验,同样式借助于刷题来完成(原型设计中的四种题型)

7.4 模糊查词的实现

h2

- 模糊查词(模糊匹配)的专业算法设计复杂的理论和推理知识,如果需要带有智能性,还需要人工智能技术来实现
- 本项目的匹配算法不具备智能性,但是也有一定的灵活性和实用性,具体的实现过程如下
 - django 实现朴素/基本模糊拼写候选/纠错
 - 使用到的拼写数据库支持(一角)
 -

	id	spelling	char_set
1	3	abandon	abdno
2	4	abatement	abemnt
3	5	abdomen	abdemno
4	6	abide	abdei
5	7	abnormal	ablmnor
6	8	aboard	abdor
7	9	abolish	abhilos
8	10	abound	abdnou
9	11	above	abeov
10	12	abroad	abdor
11	13	abrupt	abprtua
12	14	absence	abcens
13	15	absent	abenst
14	16	absolute	CSDN @helboxin1375

- 计算单词字符集: `char_set`,计算该属性的目的是为了实现单词间字符构成的相似性匹配
- 处于现实效果的考虑,当用户输入的单词长度不超过4个字符时,我们只返回单词长度相同的并且字符构成一致的单词
- 对于较长的输入,我们会允许一定比例的字符误差(譬如8个字符串长度的输入,允许波动1~2个字符),这样可以匹配到一些字母记错,字母顺序错误的情况,此外,为了使得长度合理,还我还设置了长度波动范围(也是根据比例波动(譬如25%)),这样,可以匹配到比用户拼写的单词更短一些的词汇,这种情况发生的概率较小,但是不可以完全排除
- 后端还基于此开发了丰富的query参数,客户端可以以灵活的方式调用该接口,开发出多功能的查词功能
 - 比如,强制匹配前两个字母(`startwith=2`)
 - 强制匹配终结符(`endwith=2`)
 - 甚至,后台也提供了正则匹配的功能,但是这不太是本程序的重点目标,遂没有将前端实现出来
- 接口效果:

eg0: h3

GET http://127.0.0.1:8000/word/fuzzy/{spelling}/{start_with}

模糊匹配单词(拼写形近词) (成功daed)

Params	2	Body	Headers	Cookies	Auth	前置	后置	设置
path 参数								
<input checked="" type="checkbox"/> 参数名		参数值						
<input checked="" type="checkbox"/> spelling		daed						
<input checked="" type="checkbox"/> start_with								

Body

```
1 [
2   {
3     "id": 1194,
4     "spelling": "dead",
5     "char_set": "ade"
6   }
7 ]
```

CSDN @xuchaoxin1375

eg1: h3

GET http://127.0.0.1:8000/word/fuzzy/{spelling}/{start_with}

模糊匹配单词(拼写形近词) (成功(acquieten))

Params	2	Body	Headers	Cookies	Auth	前置	后置	设置
path 参数								
<input checked="" type="checkbox"/> 参数名		参数值						
<input checked="" type="checkbox"/> spelling		acquieten						
<input checked="" type="checkbox"/> start_with								

Body

```
1 [
2   {
3     "id": 54,
4     "spelling": "acquaint",
5     "char_set": "acinqtu"
6   },
7   {
8     "id": 55,
9     "spelling": "acquaintance",
10    "char_set": "aceinqtu"
11  },
12  {
13    "id": 5382,
14    "spelling": "acquainted",
15    "char_set": "aceinqtu"
16  }
]
```

CSDN @xuchaoxin1375

eg2: h3

GET http://127.0.0.1:8000/word/fuzzy/fhather/1

模糊匹配单词(拼写形近词) (成功fhather)

Params	2	Body	Headers	Cookies	Auth	前置	后置	设置
path 参数								
<input checked="" type="checkbox"/> 参数名		参数值						
<input checked="" type="checkbox"/> spelling		fhather						
<input checked="" type="checkbox"/> start_with		1						

Body

```
1 [
2   {
3     "id": 7359,
4     "spelling": "falter",
5     "char_set": "aeflrt"
6   },
7   {
8     "id": 1789,
9     "spelling": "farther",
10    "char_set": "aefhrt"
11  },
12  {
13    "id": 1798,
14    "spelling": "father",
15    "char_set": "aefhrt"
16  },
17  {
18    "id": 1811,
19    "spelling": "feather",
20    "char_set": "aefhrt"
21  }
]
```

CSDN @xuchaoxin1375

这部分主要讲点名功能的具体实现方案，可以通过用例图、流程图等来辅助说明。并通过截图来展示效果。
实现部分不要只有截图，要有文字说明，讲讲这部分实现时采用什么技术，优缺点是什么，实现难点在哪里。

7.5 用户中心 h2

登录功能方案选型: h3

[前端常见登录实现方案 + 单点登录方案 - 掘金 \(juejin.cn\)](#)

- **Cookie + Session** 历史悠久，适合于简单的后端架构，需开发人员自己处理好安全问题。
 - **Token** 方案对后端压力小，适合大型分布式的后端架构，但已分发出去的 **token**，如果想收回权限，就不是很方便了。
 - SSO 单点登录，适用于中大型企业，想要统一内部所有产品的登录方式的情况。
 - OAuth 第三方登录，简单易用，对用户和开发者都友好，但第三方平台很多，需要选择合适自己的第三方登录平台。
-
- 综合比较,我们选择第一种方案,比较符合项目定位
 - 利用session,后端可以充分利用用户的登录状态(从客户端提交过来的cookie中解析出用户信息从而帮助前端以更加简洁的参数就可以调用经过改进后的api)
 - 另一方面,提供身份认证为数据安全提供了基本保障,减少被攻击的可能

八、系统测试 h1

这部分主要讲系统的测试方案，不要简单贴图，要写出具体的自动测试方案。可以前后端分开来写。

8.1 单元测试 h2

这部分是单元测试的方案，可以列表统计有多少测试，覆盖率是多少。

后端测试 h3

- 后端的测试采用django推荐的unittest来对代码进行测试

- 在开发过程中,我们适当的使用TDD(测试驱动开发)的方式来编写了一部分api,并且认识到了TDD开发的优势
- 编写测试需要花费一些实际,但是随着项目的体积的增长,依靠测试带来的便利就越来越显著,我们对自己的代码正确性也心中有底
- 在编写测试的时候,主要针对以下几个方面:
 - 路由和试图函数的映射是否正确
 - 在这里做测试是因为,随着项目规模的增大,我们的代码越来越复杂,对于每一个接口都要有响应的路由,这就发生后写的路由模式和之前已有的模式发生冲突,或者相互覆盖,导致api调用的时候数据传入到错误的函数中去处理
 - 编写关于视图函数的测试,这时候需要操作临时数据库,或者在测似乎代码中模拟出一些随机的或者特定的数据,总之应该使用能够达到检查目的数据
- 利用APIfox 来测试接口
 - apifox也提供了简单易用的批量测试接口的功能,可以在线上环境中进行测试,也可以在本地进行测试,甚至支持多线程测试和并发测试,而且统计了接口的调用次数和调用时间,比较直观
 - 本项目在开发过程中就受益于批量测试,帮助我在部分接口缺少测试的时候及时发现问题并解决

8.2 集成测试集 h2

- 成测试 (Integration Testing) , 也叫组装测试或联合测试。在单元测试的基础上, 将所有模块按照设计要求 (如根据结构图) 组装成为子系统或系统, 进行集成测试
- 在本项目中,我利用apifox,我定义了若干有步骤之分的测试用例,每个测试用例中可以包含多个来自接口的请求示例,并且可以调整顺序,特别是在测试登录功能的时候,依赖于登录状态的api必须要放在登录后的接口后执行,依次地完成对项目地4个模块的有序测试,所有接口,基本全部通过

8.3 测试部署及结果 h2

- 我们利用github Actions 来执行自动化测试,主要的内容是,当我从本地将项目push到github上时,我们会自动运行测试,并且将测试结果发送到我的邮箱中,这样我们就可以更加方便的了解项目的测试结果,问题和故障可以及时的得到反馈,而不需要每次在本地跑一遍所有测试,可以节约时间
- 不仅如此,团队中的其他人也可以看到测试结果

九、系统部署 [徐超信] h1

自动部署 h2

- 我们通过云主机来部署我们的项目,使得后台服务可以通过公网ip能够访问
- 我们有又利用github+webhook+github Actions实现持续继承和持续交付
- 云端可以及时自动的同步本地的最新项目成果,同时能够经过Actions的检查,依赖于本项目的其他成员就可以直到新的版本情况.

references h3

- [【github 自动部署】github实现自动部署](#)

操作步骤 h3

- 安装webbook

1 | sudo apt-get install webhook

- 编写部署脚本deploy.sh:任务内容

```

1 # cxxu @ cxxuAli in ~/backEnd on git:main x [17:53:07]
2 $ cat deploy.sh
3 #!/bin/bash
4 cd ~/backEnd/
5 #git status
6 git pull origin main
7 echo `date`
8 echo '-----the git pull origin main ran just before-----'
9

```

强拉版本:

```

1 cd /home/cxxu/backEnd/
2 # ls
3 # git checkout main
4 git reset --hard origin/main
5 git log|tail -n 10
6 # git pull origin main
7 git status |head -n 10
8 echo `date`
9 echo '----brute force pull done!(by reset --hard to the remote
origin/main---)'

```

- 编写hooks.json:

- 注意, `execute-command` :是需要运行的脚本的路径(而不是命令,譬如source `deploy.sh`)是不正确的
- `command-working-directory` 是工作目录
 - 我在实验过程中,发现,当 `deploy.sh` 和工作目录在同一目录下,才生效
 - 此外,为了确保脚本的可用性和正确性,在正式使用前应该手动运行一下脚本文件(可以利用 `chmod +x deploy.sh` 赋予脚本可执行的权限)

```

1 [
2   {
3     "id": "deploy",
4     "execute-command": "./deploy.sh",
5     "command-working-directory": "/home/cxxu/backEnd/"
6   }
7 ]
8 ]

```

- 启动服务(临时性实验)

- 进入到 `hooks.json` 所在目录中(或者指定hooks.json的绝对路径)

```
1 | webhook -hooks hooks.json -verbose
```

- 实验webhooks连接
 - 用浏览器(或者其他可以发送http请求的客户端)发送get请求 <http://123.56.72.67:9000/hooks/deploy/>
 - 这时候检查主机终端,如果能够捕获到请求,并且正确执行相关脚本,那么配置成功
- 配置github
 - 如果上述的服务启动可以正常运行,则将上述链接添加到github项目仓库的 webhook中(settings->webhook)
- 长期运行
 - 将webhook的输出内容重定向到log.txt文件中.
 - ```
1 | nohup webhook -hooks hooks.json -verbose >log.txt &
```
    - 将所有输出(包括错误输出重定向到一个文件中)
      - ```
1 | $ nohup webhook -hooks hooks.json -verbose >log.txt 2>&1 &
2 | [1] 29968
3 | 
```

利用github+webhook,实现基本的自动部署

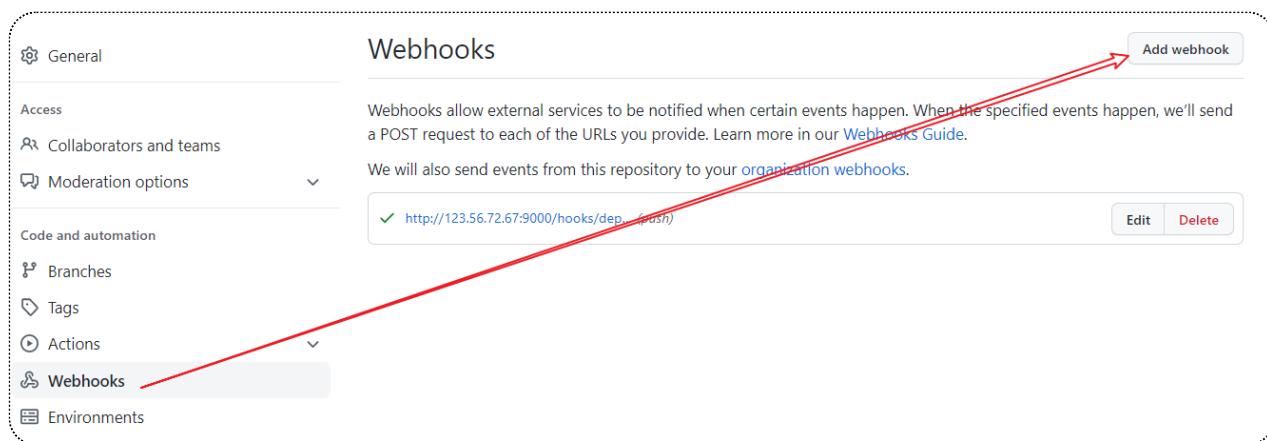


图:github&webhook

- 查看输出日志

```
1 # cxxu @ cxxuAli in ~/backEnd on git:main x [18:04:52]
2 $ cat log.txt
3 [webhook] 2022/06/06 16:44:39 version 2.5.0 starting
4 [webhook] 2022/06/06 16:44:39 setting up os signal watcher
5 [webhook] 2022/06/06 16:44:39 attempting to load hooks from
hooks.json
6 [webhook] 2022/06/06 16:44:39 found 1 hook(s) in file
7 [webhook] 2022/06/06 16:44:39 loaded: deploy
8 [webhook] 2022/06/06 16:44:39 serving hooks on
http://0.0.0.0:9000/hooks/{id}
```

十、功能展示 [潘淼森] h1

- 参看演示视频附件

十一、清单 [徐超信,潘淼森] h1

- 项目github(组织):[MaterialSharing \(github.com\)](#)

这部分列出项目提交的清单，如：

- 前端代码: Front-End
- 后端代码: backEnd
- 原型设计文件: docs/design/design_原型操作逻辑1.pptx
- 项目演示视频: docs/video.mp4
- 各种流图和思维导图文件: docs/design/
-

十二、总结 [徐超信,潘淼森] h1

项目的总结，整个项目的感受以及下一步的计划。

[version control - How do I force "git pull" to overwrite local files? - Stack Overflow](#)

在开发本项目的过程中,我们收获了很多

- 学习,了解并实践了当下流行的开发技术,体验了规范的和相对完善的开发流程
- 培养了我们的主动学习能力,思考能力以及动手能力,为我们今后的工作学习打下了重要的基础

在开发过程中,我们同样遇到了各种问题

- 由于缺乏产品设计经验以及实战开发经验,我们小组在项目之初进度就较为落后,我们深刻的认识到了,需求设计和架构设计一点也不能轻视,轻则拖慢项目开发进展,重则推导重来.
- 小组成员合理分工,沟通协商,团结一致也是分重要,不合理的分工会导致矛盾,缺乏沟通也会导致组内矛盾,影响开发效率和进度,而不团结的队伍也不利于项目的完善
- 此外还有进度安排不签当的任务复杂度估计往往会导致项目开发无法及时完成或者流程不够完善.

十二、参考文献 [徐超信,潘淼森]

h1

系统所参考的文献或者代码, 比如:

- django4: [Django: The web framework for perfectionists with deadlines](https://www.djangoproject.com)
[https://www.djangoproject.com \(google.com\)](https://www.djangoproject.com)
- Django Rest framework: [Django REST framework: Home](https://www.django-rest-framework.org) [https://www.django-rest-framework.org \(google.com\)](https://www.django-rest-framework.org)
- mysql8: [MySQL :: MySQL 8.0 Reference Manual](#)
- wechat 小程序开发文档: [微信开放文档 \(qq.com\)](#)
- vant3: [Vant 3 - 轻量、可靠的移动端组件库 \(youzan.github.io\)](#)
- uni-app: <https://uniapp.dcloud.io/>
- [version control - How do I force "git pull" to overwrite local files? - Stack Overflow](#)
-