

一、项目介绍 [徐超信,潘森森]

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

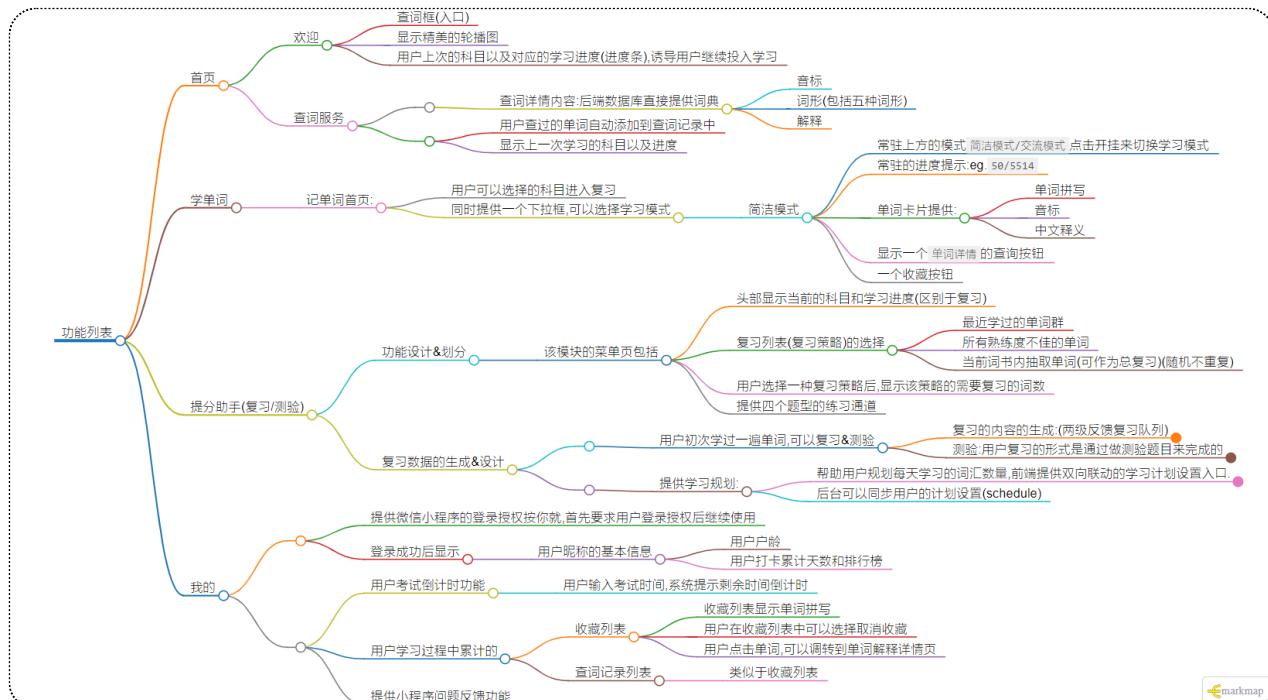
- 当下的词典软件功能很丰富,但是高频使用的功能为数不多,
- 而且充斥着各种广告和产品推销
- 软件体积较大,运行开销不小,想要流畅的使用软件,需要用户有较好的设备
- 千方百计诱导用户充值vip,添加各种限制和不必要的信息,分散用户的精力

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:

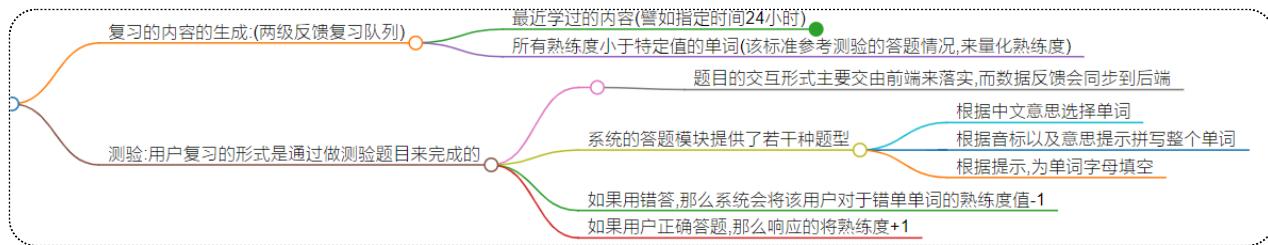
- **cet4/cet6/考研英语** 的词汇记忆学习/复习系统
- 用户可以分享自己对于词汇的记忆方法和技巧,互助学习
- 提供常规的词汇查询功能,用户可以通过输入英文单词来查询单词的基本信息(音标/词形/释义)
- 同时提供模糊查词的功能,帮助用户减轻查词过程中 **摩擦感**
- 同时提供基本的形近词推荐,帮助用户集中记忆相似单词
- 记录用户的使用痕迹和习惯,帮助用户定制学习计划,检验学习效果,集中学习专注度,拒绝分心

项目需求分析 h2

思维导图 h3



思维导图1:总体的功能设计树



思维导图2:复习部分的细节设计树

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



图2.1:首页操作

- 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



图2.2:继续复习

- 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



图2.3查词操作

- 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)

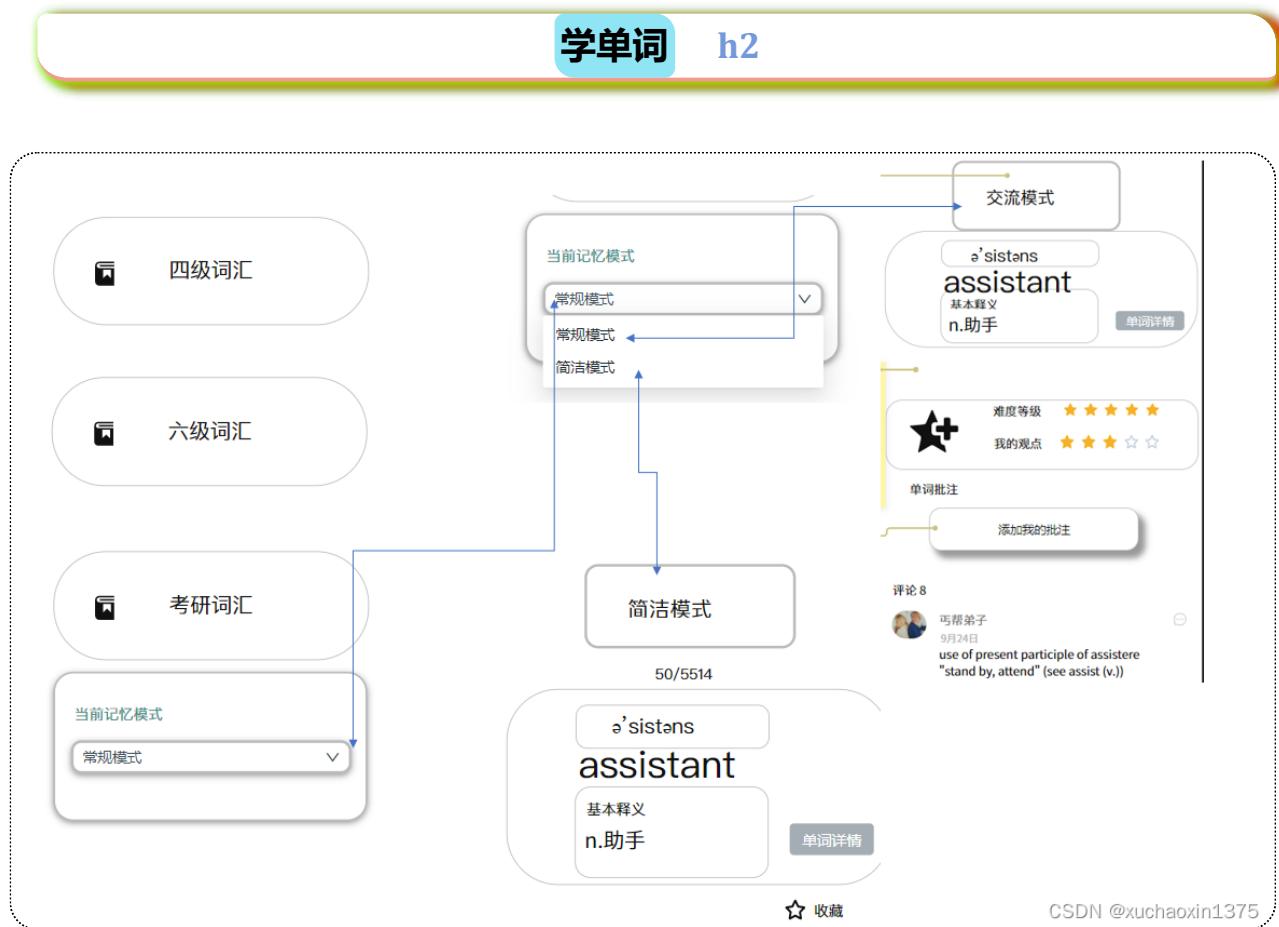


图2.4学单词操作流程

- 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,简洁模式包含内容弄个较少,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

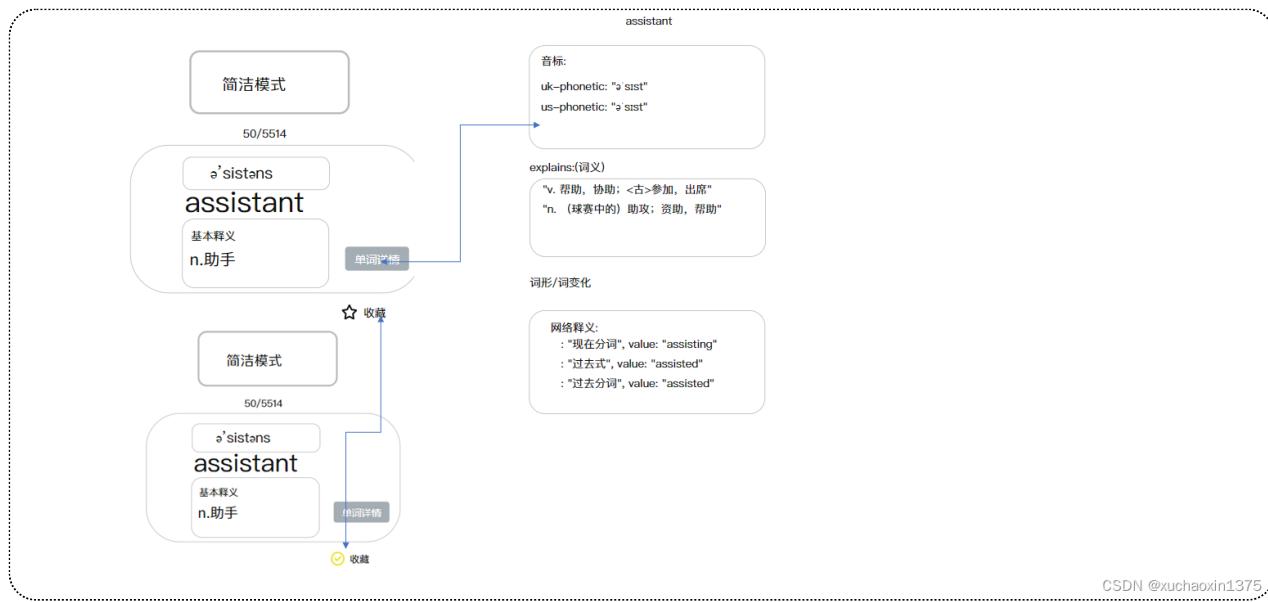


图2.5收藏单词操作

- 这是用户点击单词详情和收藏后分别的出现的响应结果



图2.6交流模式下的批注操作

- 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

复习与测验 h2

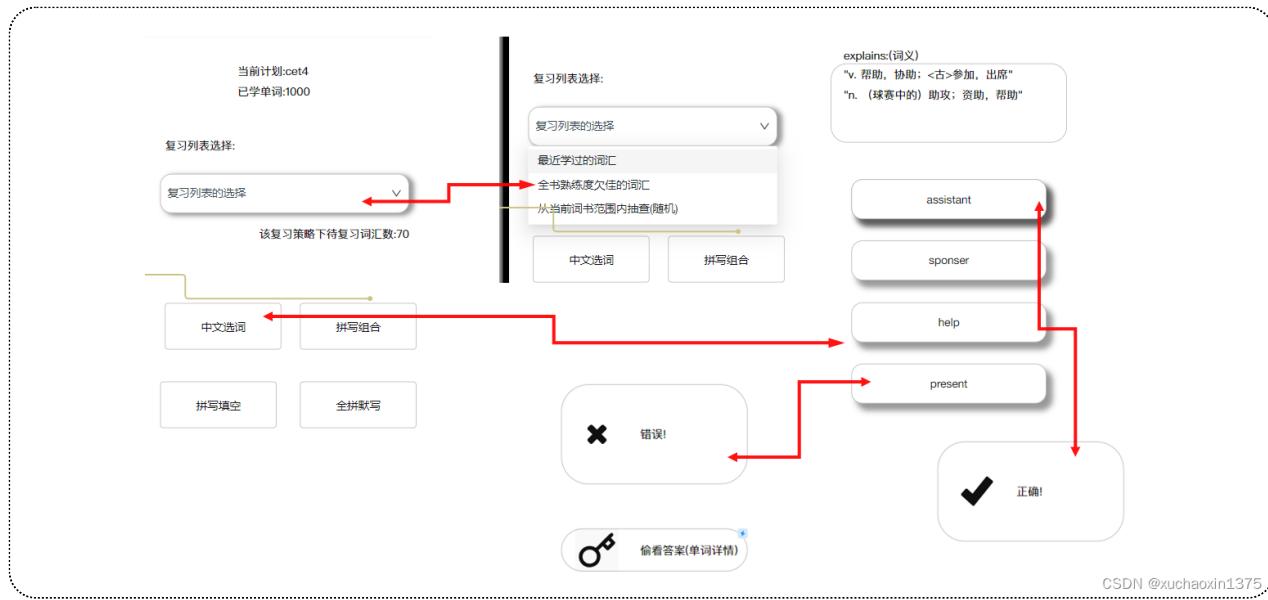


图2.7复习与测验操作流程1

- 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

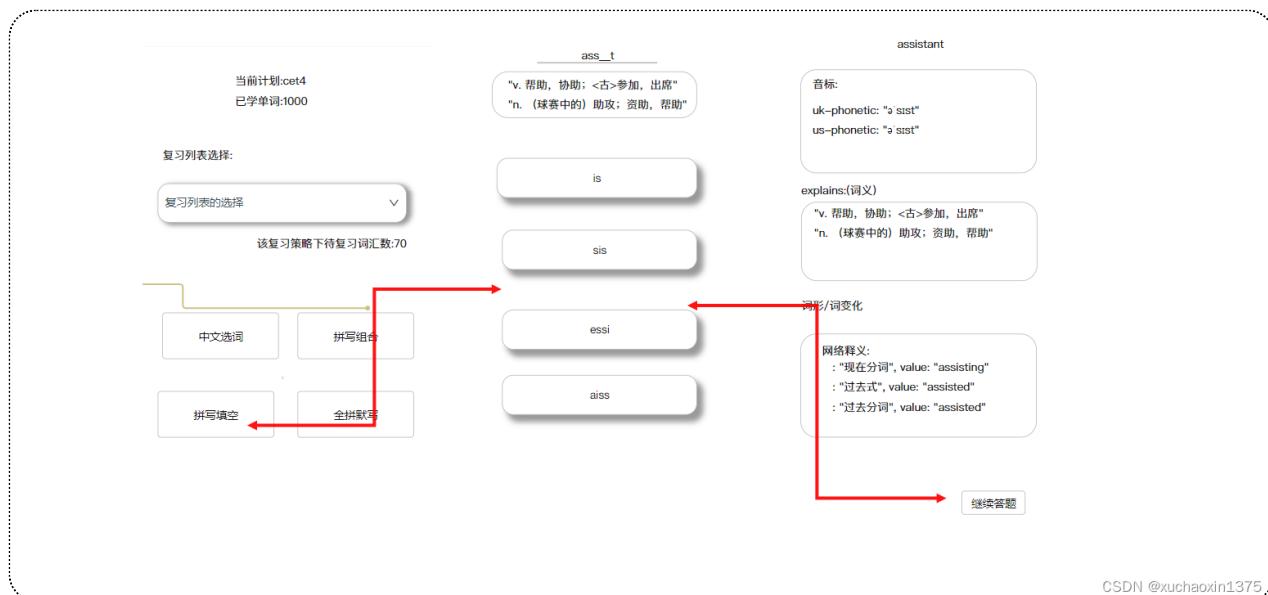


图2.8复习与测验操作流程2

- 这是拼写填空题型下的答题过程



图2.9:我的主页

- 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

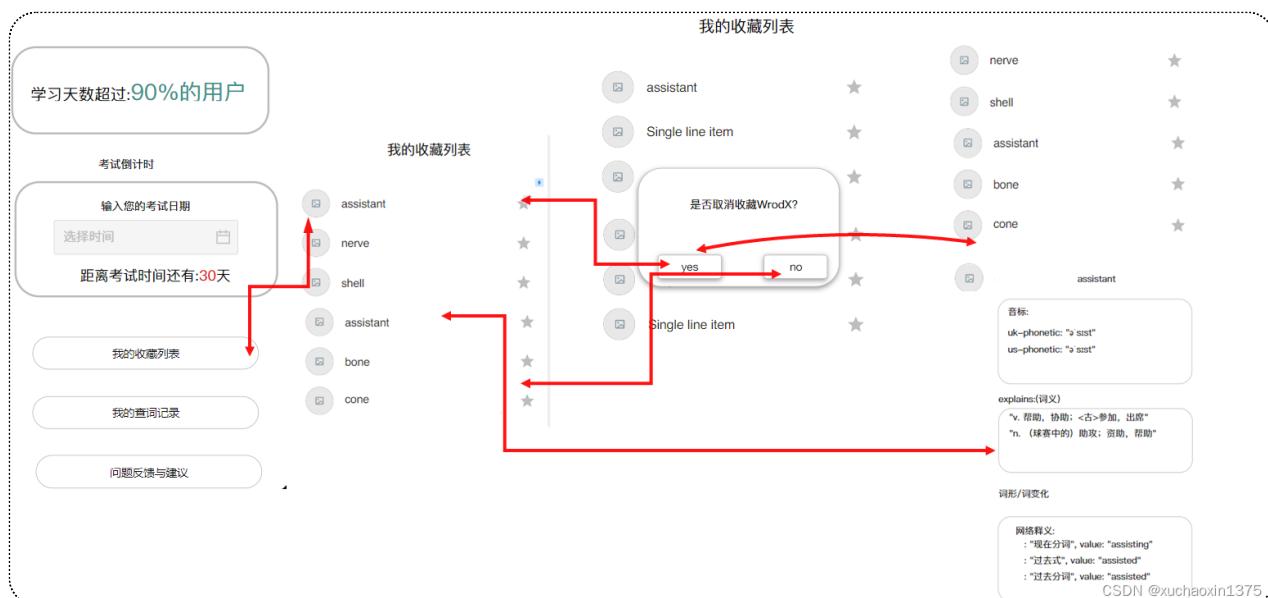


图2.10:我的主页2

- 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



图2.11:我的主页3

- 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) h2

- <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] h1

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

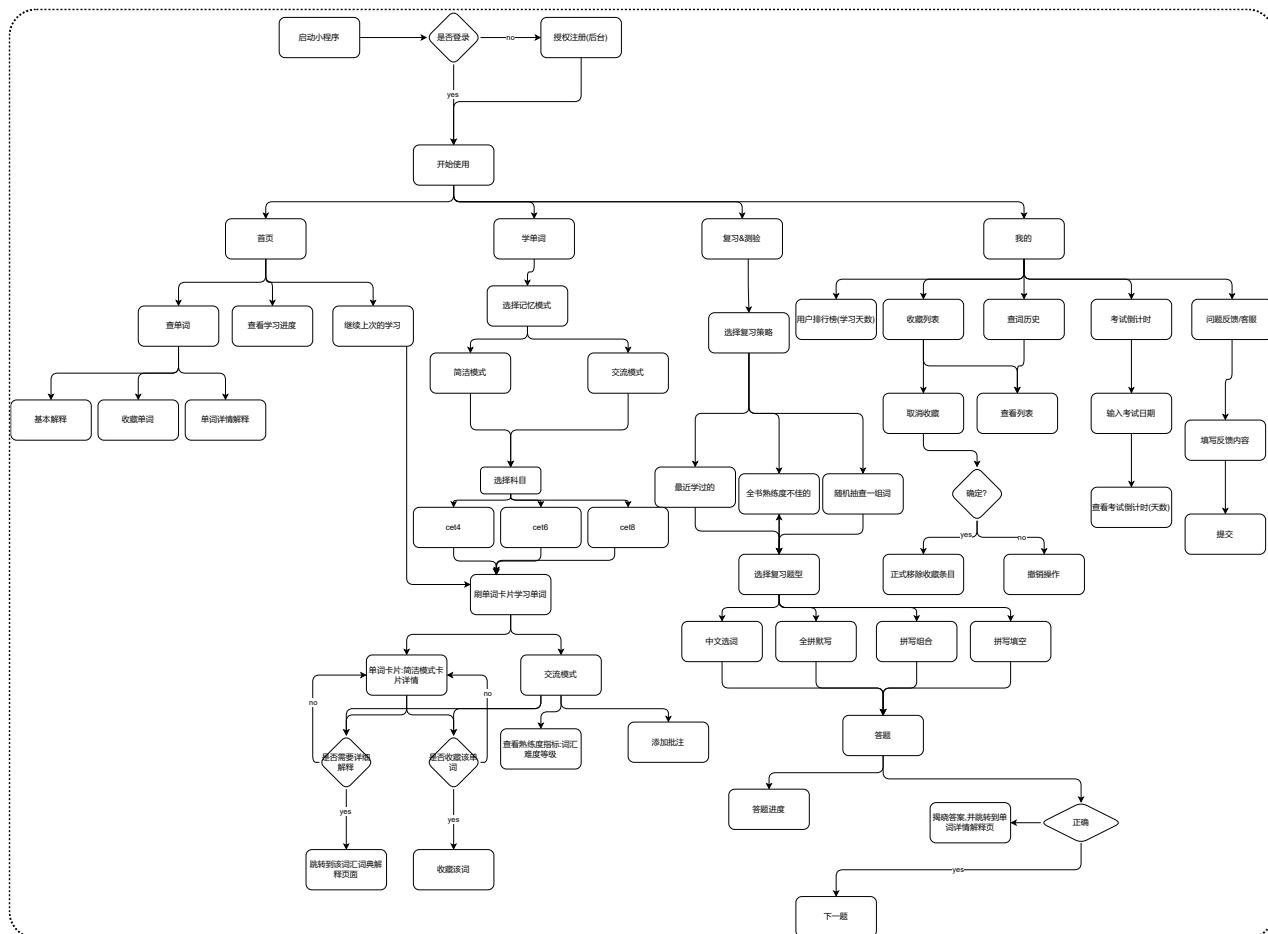


图3.1:程序操作逻辑流图

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
- 采用python mysqlclient模块来对接&管理mysql数据库
- 后端的各个模块内的基本结构(主要采用MVC设计模式来组织后端项目)
 - 根据我们选用的技术和设计模式,后端项目组织的基本规范如下(根据实际需求可以稍作调整)

```

1 PS D:\repos\ELA\backEnd\user> lsd --tree --depth 1
2 .
3 └── __init__.py
4 └── __pycache__
5 └── admin.py
6 └── apps.py
7 └── loginMiddleware.py
  
```

```
8 |   └── migrations
9 |   ├── models.py
10 |   ├── serializer.py
11 |   └── tests
12 |       ├── tests.py
13 |       └── urls.py
14 |   └── views
15 |
```

- 下面解释一下各个文件和目录的作用
- 其中admin.py作为注册后台模块管理的代码,用于管理后台的各个模块
- apps.py向后端注册该Django应用,url.py是该模块管理的子路由
- Middleware.py作为中间件,用于登录验证/或者其他鉴权/自动处理
- serializer.py是该模块使用DRF来数据模型的序列化和反序列化,可以在该文件中指定数据转化规则
- tests目录中存放了该模块下的所有测试代码
- views充当MVC中的Controller,负责存放和组织逻辑处理的代码文件

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/ (局部api接口一览)
- 

图3.2:后端的swagger文档接口

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/"
}

```

- 图:4.1

- 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 h2

- 附件中的api文档是通过apifox导出
- tables:[英语学习助手_api文档.md \(github.com\)](https://github.com)

- swagger:[英语学习助手_api.html \(github.com\)](#)
-

title: 英语学习助手 v1.0.0

language_tabs:

toc_footers: []

includes: []

search: true

code_clipboard: true

highlight_theme: darkula

headingLevel: 2

generator: "@tarslib/widdershins v4.0.11"

英语学习助手 h1

v1.0.0

单词/词典 h1

GET 获取单词释义 h2

GET /dict/{spelling}

请求参数 h3

名称	位置	类型	必选	说明
spelling	path	string	是	none
search	query	string	否	none

返回示例

成功

```
1  {
2      "wordSpelling": "fugiat culpa Excepteur sint",
3      "phonetic": "18129225380",
4      "basicExplain": "aute nostrud fugiat quis",
5      "webMeaning": [
6          "reprehenderit",
```

```
7     "ut dolore Lorem"
8 ],
9   "forms": {
10     "pl": "ea Ut do Lorem culpa",
11     "past": null,
12     "pastParticiple": null,
13     "presentParticiple": null
14   }
15 }

1 {
2   "count": 11322,
3   "next": "http://127.0.0.1:8000/word/dict/?pager=2&search=",
4   "previous": null,
5   "results": [
6     {
7       "wid": 1,
8       "spelling": "abandon",
9       "phonetic": "ə'bændən",
10      "plurality": "NULL",
11      "thirdpp": "NULL",
12      "present_participle": "NULL",
13      "past_tense": "NULL",
14      "past_participle": "NULL",
15      "explains": "[ 'v. 抛弃, 遗弃; (因危险) 离开, 舍弃; 中止, 不再有; 放弃 (信念、信仰或看法) ; 陷入, 沉湎于 (某种情感) ', 'n. 放任, 放纵' ]"
16    },
17    {
18      "wid": 2,
19      "spelling": "abatement",
20      "phonetic": "ə'beɪtmənt",
21      "plurality": "NULL",
22      "thirdpp": "NULL",
23      "present_participle": "NULL",
24      "past_tense": "NULL",
25      "past_participle": "NULL",
26      "explains": "[ 'n. 减少; 消除; 减轻' ]"
27    },
28    {
29      "wid": 3,
30      "spelling": "abdomen",
31      "phonetic": "'æbdəmən",
32      "plurality": "abdomens",
33      "thirdpp": "NULL",
34      "present_participle": "NULL",
35      "past_tense": "NULL",
36      "past_participle": "NULL",
37      "explains": "[ 'n. (人体) 腹, 腹部; (昆虫) 腹部' ]"
38    },
39    {
40      "wid": 4,
41      "spelling": "abide",
42      "phonetic": "ə'baɪd",
43      "plurality": "NULL",
44      "thirdpp": "abides",
45      "present_participle": "abiding",
46      "past_tense": "abided或abode",
47      "past_participle": "abided或abode",
```

```

48     "explains": "[ 'v. 遵守 (abide by) ; 容忍, 忍受; <旧>居住, 逗留; (感情, 记忆) 始
49     终不渝, 持续' ]"
50   },
51   {
52     "wid": 5,
53     "spelling": "abnormal",
54     "phonetic": "æb'nɔ:ml(ə)l",
55     "plurality": "NULL",
56     "thirdpp": "NULL",
57     "present_participle": "NULL",
58     "past_tense": "NULL",
59     "past_participle": "NULL",
60     "explains": "[ 'adj. 反常的, 异常的, 变态的; 不规则的' ]"
61   }
62 }
1 | "sorry, there is no explain for the word:annotation, try fuzzy match api?
    /word/fuzzy/annotation"

```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
201	<u>Created</u>	created!	Inline
404	<u>Not Found</u>	记录不存在	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» count	integer	false	none		none
» next	string	false	none		none
» previous	string	false	none		none
» results	[Word]	false	none		none
»» wid	integer	true	none		none
»» spelling	string	true	none		none
»» phonetic	string	false	none		none
»» plurality	string	false	none		none
»» thirdpp	string	false	none		none

名称	类型	必选	约束	中文名	说明
»» present_participle	string	false	none		none
»» past_tense	string	false	none		none
»» past_participle	string	false	none		none
»» explains	string	false	none		none

GET 获取单词批注list

h2

GET /note/

- 本接口的主要使用场景在于,显示某个单词的所有批注(spelling=xxx)
- 还提供了指定用户留下的批注(uid=xx)

请求参数

h3

名称	位置	类型	必选	说明
spelling	query	string	否	常用,根据拼写得到基于指定单词的相关批注
user	query	integer	否	不常用
size	query	integer	否	none
page	query	integer	否	none

返回示例

成功

```

1  {
2      "count": 42,
3      "next": "http://127.0.0.1:8000/word/note/?page=2&size=6&spelling=&user=",
4      "previous": null,
5      "results": [
6          {
7              "id": 1,
8              "user": null,
9              "spelling": "qrsxj",
10             "content": "@text",
11             "difficulty_rate": 3
12         },
13         {
14             "id": 2,
15             "user": null,
16             "spelling": "assist",

```

```

17     "content": "test word note:assist :help (someone), typically by doing a
share of the work.",
18     "difficulty_rate": 3
19   },
20   {
21     "id": 3,
22     "user": null,
23     "spelling": "assist",
24     "content": "Similar: help aid abet lend a (helping) hand to give
assistance to be of use to oblige",
25     "difficulty_rate": 3
26   },
27   {
28     "id": 4,
29     "user": null,
30     "spelling": "qnvxbvg",
31     "content": "dolore fugiat incididunt consequat",
32     "difficulty_rate": 2
33   },
34   {
35     "id": 5,
36     "user": null,
37     "spelling": "qzlveehov",
38     "content": "@text",
39     "difficulty_rate": 3
40   },
41   {
42     "id": 6,
43     "user": null,
44     "spelling": "vxvo",
45     "content": "@text",
46     "difficulty_rate": 3
47   }
48 ]
49 }

1 {
2   "count": 42,
3   "next": "http://127.0.0.1:8000/word/note/?page=2",
4   "previous": null,
5   "results": [
6     {
7       "id": 1,
8       "user": null,
9       "spelling": "qrsxj",
10      "content": "@text",
11      "difficulty_rate": 3
12    },
13    {
14      "id": 2,
15      "user": null,
16      "spelling": "assist",
17      "content": "test word note:assist :help (someone), typically by doing a
share of the work.",
18      "difficulty_rate": 3
19    },
20    {
21      "id": 3,

```

```
22     "user": null,
23     "spelling": "assist",
24     "content": "Similar: help aid abet lend a (helping) hand to give
assistance to be of use to oblige",
25     "difficulty_rate": 3
26   },
27   {
28     "id": 4,
29     "user": null,
30     "spelling": "qnvxbvg",
31     "content": "dolore fugiat incididunt consequat",
32     "difficulty_rate": 2
33   },
34   {
35     "id": 5,
36     "user": null,
37     "spelling": "qzlveehov",
38     "content": "@text",
39     "difficulty_rate": 3
40   }
41 ]
42 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

POST 单词下用户添加批注 h2

POST /note/

请求头中:

- 包含用户(uid)
- 单词拼写(spelling)
- 可选的难度评级(1-5)

Body 请求参数

```
1  {
2    "id": 21,
3    "uid": 96,
4    "spelling": "ewlxxqmu",
5    "content": "@text",
6    "difficulty_rate": 2
```

7 | }

请求参数

名称	位置	类型	必选	说明
body	body	WordNote	否	none

返回示例

成功

```
1 {  
2   "data": {  
3     "userID": "40",  
4     "noteContent": "in consequat ea"  
5   }  
6 }
```

```
1 {  
2   "id": 43,  
3   "user": null,  
4   "spelling": "vhylrkbk",  
5   "content": "@text",  
6   "difficulty_rate": 4  
7 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
201	Created	成功	Inline

返回数据结构

h3

状态码 201

名称	类型	必选	约束	中文名	说明
» data	object	false	none		none

名称	类型	必选	约束	中文名	说明
»» userID	string	true	none		none
»» noteContent	string	true	none		none

GET 获取单词的平均难度评分

h2

GET /avg-difficulty/{spelling}

- 如果前端使用session登录,请使用平均熟练度 **代替** 本接口
- 由于难度评分主观性较强,不是特别能反映问题(但是模型字段暂时保留着)
- 我们打算将平均难度以后台统计的熟练度作为衡量指标,更加具有客观性!)

请求参数

h3

名称	位置	类型	必选	说明
spelling	path	string	是	none

返回示例

成功

```

1 {
2   "spelling": "assist",
3   "avg_difficulty": 3.6667
4 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构

h3

GET 模糊匹配单词(拼写形近词)

h2

GET /fuzzy/{spelling}/{start_with}

-可以指定是否要求匹配开头(start_with个字符)

- 关于search(包括正则功能,因为返回类型不再是DRF指定的默认类型(或者说支持search查询的类型,故不可用))
可以到dict api中使用search等DRF的增益功能

请求参数 h3

名称	位置	类型	必选	说明
spelling	path	string	是	none
start_with	path	integer	是	start_with一般会更加常用,故而放置再url中,(当然也可以为例整洁一致移动到query参数中)
contain	query	string	否	一般不指定为1(除非返回结果过多,同时用户确定某个单词必定包含指定的字母)
end_with	query	string	否	django有支持直接操做的查询(spelling_endswith);然而,这种规则使用search\$正则会更方便

返回示例

成功

```

1  [
2  {
3      "id": 1215,
4      "spelling": "declare",
5      "char_set": "acdelr"
6  }
7 ]
8
9 [
10 {
11     "id": 12491,
12     "spelling": "leader",
13     "char_set": "adelr"
14 },
15 {
16     "id": 2663,
17     "spelling": "learned",
18     "char_set": "adelnr"
19 },
20 {
21     "id": 2667,
22     "spelling": "leather",
23     "char_set": "aehlrt"
24 }
25 ]

```

```

16 |     }
17 |   ]
18 |
19 | [
20 |   [
21 |     {
22 |       "id": 6301,
23 |       "spelling": "complacent",
24 |       "char_set_str": "acelmnnopt"
25 |     },
26 |     {
27 |       "id": 919,
28 |       "spelling": "complaint",
29 |       "char_set_str": "acilmnnopt"
30 |     },
31 |     {
32 |       "id": 6303,
33 |       "spelling": "complaisant",
34 |       "char_set_str": "acilmnopst"
35 |     },
36 |     {
37 |       "id": 6307,
38 |       "spelling": "compliant",
39 |       "char_set_str": "acilmnnopt"
40 |     },
41 |     {
42 |       "id": 926,
43 |       "spelling": "compliment",
44 |       "char_set_str": "ceilmnnopt"
45 |     }
46 |   ]

```

[返回结果](#) [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

[返回数据结构](#) [h3](#)

状态码 200

名称	类型	必选	约束	中文名	说明
<i>anonymous</i>	[WordMatcher]	false	none		none
» id	integer	true	none		none
» spelling	string	true	none		none
» char_set_str	string	true	none		none

PUT 修改单词批注 h2

PUT /note/{pk}/

- 一般情况下不会去用这个接口(市面上较少软件会开放修改评论的功能)
- difficulty-rate字段已经弃用

Body 请求参数 h3

```
1 {  
2   "uid": 0,  
3   "spelling": "string",  
4   "content": "string"  
5 }
```

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none
body	body	object	否	none
» uid	body	integer	是	none
» spelling	body	string	是	none
» content	body	string	是	none

返回示例

成功

```
1 {  
2   "id": 1,  
3   "user": null,  
4   "spelling": "qrsxj",  
5   "content": "@text",  
6   "difficulty_rate": 3  
7 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构

h3

GET 查询考纲词汇总数

h2

GET /sum/{examtype}/

请求参数

h3

名称	位置	类型	必选	说明
examtype	path	string	是	4/6/8分别表示cet4,6,neep(研)

返回示例

成功

```
1 | {
2 |   "examtype": "cet6",
3 |   "sum": 51
4 | }
```

```
1 | {
2 |   "examtype": "cet6",
3 |   "sum": 6000
4 | }
```

```
1 | {
2 |   "examtype": "cet4",
3 |   "sum": 4500
4 | }
```

```
1 | {
2 |   "examtype": "neep",
3 |   "sum": 5315
4 | }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	OK	成功	WordReqSum
404	Not Found	记录不存在	Inline

返回数据结构

h3

GET 获取单词的用户平均熟练度

h2

GET /avg-familiarity/{spelling}

将平均难度以后台统计的熟练度作为衡量指标,更加具有客观性!)

将三种考试类型的学习记录作为统计数据的来源!

请求参数

h3

名称	位置	类型	必选	说明
spelling	path	string	是	none

返回示例

成功

```
1 {
2   "spelling": "abandon",
3   "avg_familiarity": 3,
4   "validity": true,
5   "msg": "查询成功"
6 }

1 {
2   "spelling": "assist",
3   "avg_familiarity": 3,
4   "validity": false,
5   "msg": "没有找到相关记录,默认使用中间值3"
6 }

1 {
2   "spelling": "noita",
3   "avg_familiarity": 3,
4   "validity": false,
5   "msg": "没有找到相关记录,默认使用中间值3"
6 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
404	<u>Not Found</u>	记录不存在	Inline

返回数据结构

h3

提分助手

h1

PUT 刷新一条学习记录(G) Copy

h2

PUT /{examtype}/refresh/

Body 请求参数

```
1 {  
2   "wid": 0,  
3   "user": 0  
4 }
```

请求参数

h3

名称	位置	类型	必选	说明
examtype	path	string	是	none
body	body	object	否	none
» wid	body	integer	否	none
» user	body	integer	否	none

返回示例

返回结果

h3

状态码	状态码含义	说明	数据模型
201	Created	成功	Study

提分助手/study_aggregate(logged)

h1

PUT 答题正确/错误,熟练度±1(familiarity_change1)

h2

PUT /study/familiarity/{change}/

Body 请求参数

```
1 {  
2   "wid": 0,  
3   "user": 0,  
4   "examtype": "4"  
5 }
```

请求参数 h3

名称	位置	类型	必选	说明
change	path	string	是	none
body	body	object	否	none
» wid	body	integer	否	none
» user	body	integer	否	none
» examtype	body	string	否	none

枚举值 h4

属性	值
» examtype	4
» examtype	6
» examtype	8

返回示例

成功

```
1 {  
2   "id": 3,  
3   "last_see_datetime": "2022-06-09T11:54:57.562314Z",  
4   "examtype": "4",  
5   "familiarity": 2,  
6   "user": 1,  
7   "user_name": "Ronald Taylor",  
8   "wid": 1,  
9   "spelling": "abandon"  
10 }
```

```
1 {  
2   "id": 3,  
3   "last_see_datetime": "2022-06-09T11:55:35.550303Z",  
4   "examtype": "4",  
5   "familiarity": 4,  
6   "user": 1,  
7   "user_name": "Ronald Taylor",  
8   "wid": 1,  
9   "spelling": "abandon"  
10 }
```

```
1 {  
2   "msg": "Study not found"  
3 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline
404	Not Found	记录不存在	Inline

返回数据结构 h3

PUT 更新/添加一条学习记录(examtype) h2

PUT /study/refresh/

- 关于熟练度,在刷单词卡片的时候不应该手动更改,
- 但是,每一条学习记录又包含熟练度字段,这可以交给django模型的字段默认值来处理(默认(初始)熟练度为0)
- 熟练度的后续变更应该只通过特定的api(familiarity/)系列进行操作

Body 请求参数

```
1 {  
2   "wid": 0,  
3   "examtype": "4"  
4 }
```

请求参数 h3

名称	位置	类型	必选	说明

名称	位置	类型	必选	说明
body	body	object	否	none
» wid	body	integer	否	客户端将用户当前在学习的单词卡片的wid(单词序号)
» examtype	body	string	否	科目

枚举值 h4

属性	值
» examtype	4
» examtype	6
» examtype	8

返回示例

成功

```

1  {
2      "id": 13,
3      "last_see_datetime": "2022-06-09T11:49:10.091174Z",
4      "examtype": "8",
5      "familiarity": 0,
6      "user": 112,
7      "user_name": "cxxu",
8      "wid": 53,
9      "spelling": "acquaintance",
10     "msg": "modify the existed obj",
11     "ser": "<class 'rest_framework.serializers.SerializerMetaclass'>"
12 }

1  {
2      "id": 20,
3      "last_see_datetime": "2022-06-09T11:51:30.646249Z",
4      "examtype": "4",
5      "familiarity": 0,
6      "user": 112,
7      "user_name": "cxxu",
8      "wid": 53,
9      "spelling": "acquaintance",
10     "ser": "<class 'scoreImprover.serializer.StudyModelSerializer'>"
11 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
201	<u>Created</u>	成功	Inline

返回数据结构 h3

GET 查看学习记录(examtype) h2

GET /study/

- 由于apiFox中不如DRF自带前端界面直观,注意分页;尤其是查看所有记录的时候!

请求参数 h3

名称	位置	类型	必选	说明
user	query	string	否	用户id
page	query	string	否	计算结果较多时,采用改参数翻页
examtype	query	string	否	考试科目类型

返回示例

成功

```
1  {
2      "count": 20,
3      "next": "http://127.0.0.1:8000/improver/study/?page=2",
4      "previous": null,
5      "results": [
6          {
7              "id": 1,
8              "last_see_datetime": "2022-06-04T11:20:21.974167Z",
9              "examtype": "4",
10             "familiarity": 4,
11             "user": 4,
12             "user_name": "testScriptUser",
13             "wid": 72,
14             "spelling": "additional"
15         },
16         {
17             "id": 2,
```

```
18     "last_see_datetime": "2022-06-04T11:21:42.463210Z",
19     "examtype": "8",
20     "familiarity": 1,
21     "user": 2,
22     "user_name": "create0000_pyt_er",
23     "wid": 79,
24     "spelling": "adjust"
25   },
26   {
27     "id": 3,
28     "last_see_datetime": "2022-06-06T07:08:29.546124Z",
29     "examtype": "4",
30     "familiarity": 4,
31     "user": 1,
32     "user_name": "Ronald Taylor",
33     "wid": 1,
34     "spelling": "abandon"
35   },
36   {
37     "id": 4,
38     "last_see_datetime": "2022-06-04T12:05:15.371155Z",
39     "examtype": "6",
40     "familiarity": 4,
41     "user": 1,
42     "user_name": "Ronald Taylor",
43     "wid": 1,
44     "spelling": "abandon"
45   },
46   {
47     "id": 5,
48     "last_see_datetime": "2022-06-06T05:57:31.894049Z",
49     "examtype": "4",
50     "familiarity": 3,
51     "user": 1,
52     "user_name": "Ronald Taylor",
53     "wid": 65,
54     "spelling": "actress"
55   }
56 ]
57 }

1  {
2    "count": 6,
3    "next": "http://127.0.0.1:8000/improver/study/?examtype=6&page=2",
4    "previous": null,
5    "results": [
6      {
7        "id": 4,
8        "last_see_datetime": "2022-06-04T12:05:15.371155Z",
9        "examtype": "6",
10       "familiarity": 4,
11       "user": 1,
12       "user_name": "Ronald Taylor",
13       "wid": 1,
14       "spelling": "abandon"
15     },
16     {
17       "id": 8,
```

```

18     "last_see_datetime": "2022-06-06T06:03:45.236030Z",
19     "examtype": "6",
20     "familiarity": 3,
21     "user": 112,
22     "user_name": "cxxu",
23     "wid": 645,
24     "spelling": "cap"
25   },
26   {
27     "id": 12,
28     "last_see_datetime": "2022-06-09T09:45:30.217145Z",
29     "examtype": "6",
30     "familiarity": 2,
31     "user": 112,
32     "user_name": "cxxu",
33     "wid": 1,
34     "spelling": "abandon"
35   },
36   {
37     "id": 15,
38     "last_see_datetime": "2022-06-09T09:47:04.915662Z",
39     "examtype": "6",
40     "familiarity": 0,
41     "user": 112,
42     "user_name": "cxxu",
43     "wid": 70,
44     "spelling": "addict"
45   },
46   {
47     "id": 16,
48     "last_see_datetime": "2022-06-09T09:47:08.725863Z",
49     "examtype": "6",
50     "familiarity": 0,
51     "user": 112,
52     "user_name": "cxxu",
53     "wid": 47,
54     "spelling": "accustomed"
55   }
56 ]
57 }

1 {
2   "detail": "Invalid page."
3 }

```

[返回结果](#) [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Study

[提分助手/study_separates](#) [h1](#)

GET 查看学习记录(G) query h2

GET /{examtype}/

- 由于apiFox中不如DRF自带前端界面直观,注意分页;尤其是查看所有记录的时候!

请求参数 h3

名称	位置	类型	必选	说明
examtype	path	string	是	none
user	query	string	否	用户id
page	query	string	否	计算结果较多时,采用改参数翻页

返回示例

成功

```
1  {
2      "count": 3,
3      "next": null,
4      "previous": null,
5      "results": [
6          {
7              "id": 1,
8              "last_see_datetime": "2022-05-15T12:26:16.932885Z",
9              "familiarity": 3,
10             "user": 20,
11             "wid": 1
12         },
13         {
14             "id": 2,
15             "last_see_datetime": "2022-05-15T12:26:21.770543Z",
16             "familiarity": 4,
17             "user": 3,
18             "wid": 4
19         },
20         {
21             "id": 3,
22             "last_see_datetime": "2022-05-15T12:26:34.939649Z",
23             "familiarity": 4,
24             "user": 3,
25             "wid": 4
26         }
27     ]
28 }
```

```
1  {
2      "count": 2,
```

```

3 "next": null,
4 "previous": null,
5 "results": [
6   {
7     "id": 1,
8     "last_see_datetime": "2022-05-15T12:25:30.107155Z",
9     "familiarity": 2,
10    "user": 4,
11    "wid": 5
12  },
13  {
14    "id": 2,
15    "last_see_datetime": "2022-05-15T12:25:32.788455Z",
16    "familiarity": 2,
17    "user": 4,
18    "wid": 5
19  }
20 ]
21 }

```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	<u>Study</u>

PUT 答题正确/错误,熟练度±1(familiarity_change1) h2

PUT /study/{examtype}/familiarity/{change}/

- restful 风格的api不宜使用动词
- 而某某些时候,用动词可以更加贴切的描述api的意图
- 可以采取折衷的方案来命名:将api命名为服务名词(譬如转账服务transction/自增服务)
- 其实,如果希望名词来表征熟练度的变更,可以用penalty/decrement
- 另一方面,使用increment来表针熟练度的提升服务

Body 请求参数

```

1 {
2   "wid": 0,
3   "user": 0,
4   "examtype": "4"
5 }

```

请求参数 h3

名称	位置	类型	必选	说明
----	----	----	----	----

名称	位置	类型	必选	说明
examtype	path	string	是	none
change	path	string	是	none
body	body	object	否	none
» wid	body	integer	否	none
» user	body	integer	否	none
» examtype	body	string	否	none

枚举值 [h4](#)

属性	值
» examtype	4
» examtype	6
» examtype	8

返回示例

成功

```

1  {
2    "id": 6,
3    "last_see_datetime": "2022-06-09T12:32:38.740928Z",
4    "familiarity": 3,
5    "user": 1,
6    "user_name": "Ronald Taylor",
7    "wid": 1,
8    "spelling": "abandon"
9  }

```

返回结果 [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 [h3](#)

[GET 随机抽查一组单词 G \(sizeable\)](#) [h2](#)

GET /review/{examtype}/{size}

每次返回size个单词 h2

请求参数 h3

名称	位置	类型	必选	说明
examtype	path	string	是	none
size	path	string	是	none

返回示例

成功

```
1 [ 
2 { 
3     "wordorder": 47, 
4     "spelling": "njfbckk" 
5 }, 
6 { 
7     "wordorder": 7, 
8     "spelling": "vvrcll" 
9 }, 
10 { 
11     "wordorder": 60, 
12     "spelling": "nkewc" 
13 } 
14 ] 

1 [ 
2 { 
3     "examtype": "cet4", 
4     "queryset": "word.Cet4WordsReq.objects", 
5     "ser": "<class 'word.serializer.Cet4WordsReqModelSerializer'>" 
6 }, 
7 { 
8     "wordorder": 1909, 
9     "spelling": "illiterate" 
10 }, 
11 { 
12     "wordorder": 2117, 
13     "spelling": "kind" 
14 }, 
15 { 
16     "wordorder": 2991, 
17     "spelling": "profile" 
18 }, 
19 { 
```

```
20     "wordorder": 3772,  
21     "spelling": "structure"  
22 }  
23 ]
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	string

GET 检查最近添加的学习记录的单词列表unitable(all users) h2

GET /neep/timedelta/{unit}/{value}

请求参数 h3

名称	位置	类型	必选	说明
unit	path	string	是	none
value	path	number	是	none

返回示例

成功

```
1 [  
2 {  
3     "wid": 79,  
4     "last_see_datetime": "1986-04-27 19:07:08",  
5     "user": 2,  
6     "familiarity": 1,  
7     "id": 8  
8 },  
9 {  
10    "id": 25,  
11    "user": 1,  
12    "last_see_datetime": "1979-11-21 16:45:48",  
13    "wid": 39,  
14    "familiarity": 2  
15 },  
16 {  
17    "id": 11,  
18    "last_see_datetime": "1993-11-13 23:41:29",  
19    "wid": 63,  
20    "user": 4,  
21    "familiarity": 3
```

```
22 | }  
23 | ]
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
<i>anonymous</i>	[Study]	false	none		none
» id	integer	false	none		none
» wid	integer	false	none		none
» last_see_datetime	string	false	none		none
» familiarity	integer	false	none		none
» user	integer	false	none		none
» examtype	string	false	none		none

枚举值 h4

属性	值
examtype	4
examtype	6
examtype	8

PUT 刷新/创建一条学习记录(G) h2

PUT /study/{examtype}/

Body 请求参数

```
1 | {  
2 |   "wid": 0,
```

```
3     "user": 0  
4 }
```

请求参数

h3

名称	位置	类型	必选	说明
examtype	path	string	是	考试科目
body	body	object	否	none
» wid	body	integer	否	none
» user	body	integer	否	none

返回示例

成功

```
1 {  
2   "id": 19,  
3   "last_see_datetime": "2022-05-15T15:59:23.958299Z",  
4   "familiarity": 4,  
5   "user": 5,  
6   "wid": 1,  
7   "examtype": "cet6",  
8   "msg": "modify the existed obj",  
9   "ser": "<class 'rest_framework.serializers.SerializerMetaclass'>"  
10 }  
  
1 {  
2   "id": 26,  
3   "last_see_datetime": "2022-05-15T16:01:51.810349Z",  
4   "familiarity": 1,  
5   "user": 3,  
6   "wid": 2,  
7   "examtype": "neep",  
8   "msg": "modify the existed obj",  
9   "ser": "<class 'rest_framework.serializers.SerializerMetaclass'>"  
10 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
201	Created	成功	Inline

返回数据结构

h3

状态码 201

名称	类型	必选	约束	中文名	说明
» id	integer	false	none		none
» wid	integer	false	none		none
» last_see_datetime	string	false	none		none
» familiarity	integer	false	none		none
» user	integer	false	none		none
» examtype	string	false	none		none

枚举值

h4

属性	值
examtype	4
examtype	6
examtype	8

用户/login h1

DELETE 登出(注销) h2

DELETE /logout/

Body 请求参数

```
1 | {  
2 |   "name": "string"  
3 | }
```

请求参数

h3

名称	位置	类型	必选	说明
body	body	object	否	none

名称	位置	类型	必选	说明
» name	body	string	是	none

返回示例

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
204	<u>No Content</u>	删除成功	Inline

返回数据结构 h3

POST 用户登录 h2

POST /dologin/

admin/login/

Body 请求参数

```
1 | {
2 |   "account": "string",
3 |   "password": "string"
4 | }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	object	否	none
» account	body	string	是	none
» password	body	string	是	none

返回示例

成功

```
1 | {
2 |     "login_status": "success"
3 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
404	<u>Not Found</u>	记录不存在	Inline

返回数据结构 h3

GET 从session获取当前登录的用户信息 h2

GET /fetch-user/

- 登录成功后,服务端会向客户端发放cookie凭证,用户读取被cookie管理的session,可以提取除响应的用户信息
- 调用本接口不需要任何参数
- 可以获得不敏感的用户数据
- 但是应该确保在登录成功才调用本函数

返回示例

成功

```
1 | {
2 |     "uid": 112,
3 |     "nickname": "testNickname",
4 |     "name": "cxxu",
5 |     "status": 0,
6 |     "signin": 0,
7 |     "openid": null,
8 |     "examdate": "2023-08-15",
9 |     "examtype": "6",
10 |    "signupdate": "2022-05-28",
11 |    "schedule": 30
12 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
-----	-------	----	------

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

POST 新建用户(password) h2

POST /register/

可以设置密码的api(但对于小程序来说并不常用)

Body 请求参数

```

1  {
2    "name": "string",
3    "examtype": "string",
4    "examdate": "string",
5    "password": "string"
6  }

```

请求参数 h3

名称	位置	类型	必选	说明
body	body	<u>UserSignupPassword</u>	否	none

返回示例

成功

```

1  {
2    "uid": 161,
3    "name": "cxxu",
4    "password_hash": "4cc70c17134c1cc950f9aef2e6b1a8ca",
5    "password_salt": "7233",
6    "status": 0,
7    "signin": 0,
8    "openid": null,
9    "examdate": "2022-08-28",
10   "examtype": "6",
11   "signupdate": "2022-06-09",
12   "schedule": 30
13 }

```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
201	<u>Created</u>	成功(created)	Inline

返回数据结构 h3

用户/Info h1

PUT 修改用户信息 h2

PUT /info/{id}/

- 本接口用于修改用户信息
- 包括考试日期,考试类型,
- 具体字段参看请求参数/Body中所定义的

Body 请求参数

```
1 | {
2 |   "name": "string",
3 |   "examtype": "string",
4 |   "examdate": "string"
5 | }
```

请求参数 h3

名称	位置	类型	必选	说明
id	path	string	是	none
body	body	<u>UserUpdate</u>	否	none

返回示例

成功

```
1 | {
2 |   "uid": 84,
```

```
3 "name": "Christopher Johnson",
4 "signin": 64,
5 "examtype": "6",
6 "examdate": "1993-02-17",
7 "signupdate": "2007-03-18"
8 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	User

GET 获取用户的学习计划 h2

GET /info/{pk}/schedule/

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none

返回示例

成功

```
1 {
2   "user": 1,
3   "schedule": 67
4 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline
404	Not Found	记录不存在	Inline

返回数据结构 h3

GET 查询用户详情

h2

GET /info/

如果不传入id,则查询所有用户

请求参数

h3

名称	位置	类型	必选	说明
page	query	integer	否	查看第几页数据
size	query	integer	否	本次请求获取多少条记录
search	query	string	否	支持正则搜索名字

返回示例

成功

```
1 {  
2   "uid": 1,  
3   "nickname": "testNickname",  
4   "name": "Ronald Taylor",  
5   "signin": 28,  
6   "openid": null,  
7   "examtype": "6",  
8   "examdate": "2023-01-13",  
9   "signupdate": "1970-01-01"  
10 }  
  
1 {  
2   "userNickName": "鍾娜",  
3   "sumSignIn": 23,  
4   "rankSuperior": null,  
5   "exameCountdown": null,  
6   "exameType": 6  
7 }  
  
1 {  
2   "count": 152,  
3   "next": "http://127.0.0.1:8000/user/info/?page=2",  
4   "previous": null,  
5   "results": [  
6     {  
7       "uid": 13,  
8       "nickname": "testNickname",  
9       "name": "create_ser_pyt",  
10      "status": 0,  
11      "signin": 778,
```

```
12     "openid": null,
13     "examdate": "1970-01-01",
14     "examtype": "4",
15     "signupdate": "1970-01-01",
16     "schedule": 30
17   },
18   {
19     "uid": 3,
20     "nickname": "testNickname",
21     "name": "testScriptUser",
22     "status": 0,
23     "signin": 0,
24     "openid": null,
25     "examdate": "1970-01-01",
26     "examtype": "4",
27     "signupdate": "1970-01-01",
28     "schedule": 30
29   },
30   {
31     "uid": 4,
32     "nickname": "testNickname",
33     "name": "testScriptUser",
34     "status": 0,
35     "signin": 0,
36     "openid": null,
37     "examdate": "1970-01-01",
38     "examtype": "4",
39     "signupdate": "1970-01-01",
40     "schedule": 30
41   },
42   {
43     "uid": 5,
44     "nickname": "testNickname",
45     "name": "testScriptUser",
46     "status": 0,
47     "signin": 0,
48     "openid": null,
49     "examdate": "1970-01-01",
50     "examtype": "4",
51     "signupdate": "1970-01-01",
52     "schedule": 30
53   },
54   {
55     "uid": 31,
56     "nickname": "testNickname",
57     "name": "create_ser_M_pyt",
58     "status": 0,
59     "signin": 8,
60     "openid": null,
61     "examdate": "1970-01-01",
62     "examtype": "4",
63     "signupdate": "1970-01-01",
64     "schedule": 30
65   }
66 ]
67 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» data	<u>User</u>	false	none		none
»» uid	integer	true	none		none
»» name	string	true	none		none
»» signin	integer	true	none		none
»» examtype	string	true	none		none
»» examdate	string	true	none		none
»» signupdate	string	true	none		none

POST 新建一个用户(适合于微信小程序授权的方式(无密码)) h2

POST /info/

- 新建一个用户(适合于微信小程序授权的方式(无密码))
- 在login子模块中,有可以设置密码的api(但对于小程序来说并不常用)

Body 请求参数

```

1 {
2   "name": "string",
3   "examtype": "string",
4   "examdate": "string"
5 }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	<u>UserSignUp</u>	否	none

[返回示例](#)

成功

```

1  {
2    "uid": 35,
3    "name": "Linda Anderson",
4    "signin": 54,
5    "examtype": "4",
6    "examdate": "1974-06-06",
7    "signupdate": "2007-01-25"
8  }

```

[返回结果](#) h3

状态码	状态码含义	说明	数据模型
201	Created	成功	User

[GET 查询指定用户\(通过uid查询\)详情](#) h2

GET /info/{id}

如果不传入id,则查询所有用户

[请求参数](#) h3

名称	位置	类型	必选	说明
id	path	string	是	none

[返回示例](#)

成功

```

1  {
2    "uid": 1,
3    "nickname": "testNickname",
4    "name": "Ronald Taylor",
5    "status": 0,
6    "signin": 27,
7    "openid": null,
8    "examdate": "2020-01-13",
9    "examtype": "4",
10   "signupdate": "1970-01-01",

```

```
11 |     "schedule": 67  
12 | }
```

```
1 | {  
2 |     "detail": "Not found."  
3 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» data	<u>User</u>	false	none		none
»» uid	integer	true	none		none
»» name	string	true	none		none
»» signin	integer	true	none		none
»» examtype	string	true	none		none
»» examdate	string	true	none		none
»» signupdate	string	true	none		none

PUT 更新用户学习计划 h2

PUT /info/{pk}/

info/schedule/

Body 请求参数

```
1 {  
2   "user": 0,  
3   "schedual": 0  
4 }
```

请求参数

h3

名称	位置	类型	必选	说明
pk	path	string	是	none
body	body	object	否	none
» user	body	integer	是	none
» schedual	body	integer	是	none

返回示例

成功

```
1 {  
2   "uid": 1,  
3   "nickname": "testNickname",  
4   "name": "Ronald Taylor",  
5   "status": 0,  
6   "signin": 27,  
7   "openid": null,  
8   "examdate": "2020-01-13",  
9   "examtype": "4",  
10  "signupdate": "1970-01-01",  
11  "schedule": 65  
12 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构

h3

用户/star_logged

GET 查询收藏列表

h2

GET /star-logged/

这个接口主要通过query参数传参;
但是可以配合过滤(query参数来获取有用的信息)

返回示例

成功

```
1  {
2      "count": 8,
3      "next": "http://127.0.0.1:8000/user/star-logged/?page=2",
4      "previous": null,
5      "results": [
6          {
7              "id": 99,
8              "spelling": "apple",
9              "user": 112
10         },
11         {
12             "id": 85,
13             "spelling": "egiunvr",
14             "user": 112
15         },
16         {
17             "id": 86,
18             "spelling": "app",
19             "user": 112
20         },
21         {
22             "id": 92,
23             "spelling": "bar",
24             "user": 112
25         },
26         {
27             "id": 98,
28             "spelling": "bar",
29             "user": 112
30         }
31     ]
32 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

状态码 200

名称	类型	必选	约束	中文名	说明
» count	string	false	none		none
» next	string	false	none		none
» previous	null	false	none		none
» results	[Star]	false	none		none
»» id	integer	true	none		收藏条目id
»» spelling	string	true	none		单词拼写
»» user	integer	true	none		用户id

POST 收藏单词 h2

POST /star-logged/

借助于session,(登录状态下,不需要手动传入user id,
正确用法:post:将需要传入的数据写入到body中,发送

Body 请求参数

```

1 | {
2 |   "spelling": "string"
3 | }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	object	否	none
» spelling	body	string	是	none

返回示例

成功

```
1 | {
```

```
2 |     "id": 99,  
3 |     "spelling": "apple",  
4 |     "user": 112  
5 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
201	<u>Created</u>	成功	<u>Star</u>

返回数据结构 h3

DELETE 删除收藏单词 h2

DELETE /star-logged/

这个接口主要通过query参数传参;
但是可以配合过滤(query参数来获取有用的信息)

Body 请求参数

```
1 | {  
2 |     "spelling": "string"  
3 | }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	object	否	none
» spelling	body	string	是	none

返回示例

成功

```
1 | {  
2 |     "msg": "delete success"
```

返回结果
h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
204	<u>No Content</u>	删除成功	Inline

返回数据结构
h3

状态码 200

名称	类型	必选	约束	中文名	说明
» count	string	false	none		none
» next	string	false	none		none
» previous	null	false	none		none
» results	<u>[Star]</u>	false	none		none
»» id	integer	true	none		收藏条目id
»» spelling	string	true	none		单词拼写
»» user	integer	true	none		用户id

GET 根据收藏id查询收藏详情(deprecated)
h2

GET /star/{id}/

请求参数
h3

名称	位置	类型	必选	说明
id	path	string	是	none

返回示例

返回结果
h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

用户/star(no_log) h1

POST 收藏单词 h2

POST /star/

正确用法:post:将需要传入的数据写入到body中,发送

错误用法(get):bad: user/{uid}/star/{word}

Body 请求参数

```

1  {
2    "user": 0,
3    "spelling": "string"
4 }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	object	否	none
» user	body	integer	是	none
» spelling	body	string	是	none

返回示例

成功

```

1  {
2    "id": 5,
3    "spelling": "ziuuqgvyyp",
4    "user": 1
5 }
```

```
1 {
```

```
2     "id": 100,  
3     "spelling": "apple",  
4     "user": 112  
5 }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
201	<u>Created</u>	成功	<u>Star</u>

GET 查询收藏列表

h2

GET /star/

这个接口主要通过query参数传参;
但是可以配合过滤(query参数来获取有用的信息)

请求参数

h3

名称	位置	类型	必选	说明
user	query	string	否	查询用户1所有收藏记录
spelling	query	string	否	该单词被收藏的情况
search	query	string	否	正则搜索

返回示例

成功

```
1 {  
2   "data": [  
3     {  
4       "spelling": "atjxutl"  
5     },  
6     {  
7       "spelling": "shodjrb"  
8     },  
9     {  
10      "spelling": "merg"  
11    }  
12  ]  
13 }
```

```
1 | {
2 |     "next": "dolor do ex in",
3 |     "results": [
4 |         {
5 |             "id": 1,
6 |             "spelling": "qtrd",
7 |             "user": 17
8 |         },
9 |         {
10 |             "id": 8,
11 |             "spelling": "zggur",
12 |             "user": 11
13 |         }
14 |     ],
15 |     "count": 56,
16 |     "previous": null
17 | }
18 |
19 | {
20 |     "count": 80,
21 |     "next": "http://127.0.0.1:8000/user/star/?page=2&search=&spelling=&user=",
22 |     "previous": null,
23 |     "results": [
24 |         {
25 |             "id": 1,
26 |             "spelling": "second",
27 |             "user": 13
28 |         },
29 |         {
30 |             "id": 2,
31 |             "spelling": "video",
32 |             "user": 13
33 |         },
34 |         {
35 |             "id": 3,
36 |             "spelling": "defeat",
37 |             "user": 13
38 |         },
39 |         {
40 |             "id": 4,
41 |             "spelling": "subm",
42 |             "user": 22
43 |         },
44 |         {
45 |             "id": 5,
46 |             "spelling": "egiunvr",
47 |             "user": 23
48 |         }
49 |     ]
50 | }
51 | {
52 |     "count": 3,
53 |     "next": null,
54 |     "previous": null,
55 |     "results": [
56 |         {
```

```

7     "id": 1,
8     "spelling": "second",
9     "user": 13
10    },
11    {
12      "id": 2,
13      "spelling": "video",
14      "user": 13
15    },
16    {
17      "id": 3,
18      "spelling": "defeat",
19      "user": 13
20    }
21  ]
22 }
```

[返回结果](#) [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

[返回数据结构](#) [h3](#)

状态码 200

名称	类型	必选	约束	中文名	说明
» count	string	false	none		none
» next	string	false	none		none
» previous	null	false	none		none
» results	[Star]	false	none		none
»» id	integer	true	none		收藏条目id
»» spelling	string	true	none		单词拼写
»» user	integer	true	none		用户id

[DELETE 删除收藏单词](#) [h2](#)

DELETE /star/{pk}/

这个接口主要通过path参数传参;

Body 请求参数

1 | {}

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none
body	body	object	否	none

返回示例

成功

```
1 | {  
2 |     "detail": "Not found."  
3 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	string
204	No Content	删除成功	Inline

返回数据结构 h3

用户/history h1

GET 查询用户的查词记录 h2

GET /history/

- 使用query参数进行查询

请求参数 h3

名称	位置	类型	必选	说明

名称	位置	类型	必选	说明
user	query	string	否	查询用户1的所有查词记录

返回示例

成功

```
1 {  
2   "historyList": [  
3     {  
4       "wordSpelling": "evh"  
5     },  
6     {  
7       "wordSpelling": "udhrxc"  
8     }  
9   ]  
10 }
```

```
1 {  
2   "historyList": []  
3 }
```

```
1 {  
2   "historyList": [  
3     {  
4       "user": 3,  
5       "spelling": "wsbcmtnoxy"  
6     },  
7     {  
8       "user": 5,  
9       "spelling": "ymksmvknk"  
10    },  
11    {  
12      "user": 4,  
13      "spelling": "ssuvrl"  
14    },  
15    {  
16      "user": 5,  
17      "spelling": "ace"  
18    },  
19    {  
20      "user": 3,  
21      "spelling": "nbxsqqlxn"  
22    }  
23  ]  
24 }
```

```
1 {  
2   "count": 2,  
3   "next": null,  
4   "previous": null,  
5   "results": [  
6     {
```

```
7     "id": 7,
8     "spelling": "etb",
9     "user": 3
10    },
11    {
12      "id": 9,
13      "spelling": "jeibevvnpz",
14      "user": 3
15    }
16  ]
17 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» count	integer	false	none		none
» next	null	false	none		none
» previous	null	false	none		none
» results	[object]	false	none		none
»» id	integer	true	none		none
»» spelling	string	true	none		none
»» user	integer	true	none		none

POST 添加一条搜索记录 h2

POST /history/

Body 请求参数

```
1  {
2    "user": 0,
3    "spelling": "string"
4  }
```

请求参数 h3

名称	位置	类型	必选	说明
body	body	WordSearchHistory	否	none

返回示例

created!

```
1 {  
2   "user": 1,  
3   "spelling": "usttkh"  
4 }  
  
1 [  
2   {  
3     "spelling": "ztwuwnyoc"  
4   },  
5   {  
6     "spelling": "wxbiq"  
7   },  
8   {  
9     "spelling": "pnvh"  
10  },  
11  {  
12    "spelling": "osbnlhiklw"  
13  },  
14  {  
15    "spelling": "scpaltgo"  
16  }  
17 ]  
  
1 {  
2   "id": 69,  
3   "spelling": "fpbioyermt",  
4   "user": 4  
5 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
201	Created	created!	Study

用户/extr [h1](#)

GET 查询学习进度 logged h2

GET /info/progress/{examtype}/

请求参数 h3

名称	位置	类型	必选	说明
examtype	path	string	是	none

返回示例

成功

```
1 {  
2   "user": 112,  
3   "examtype": "4",  
4   "progress": 15  
5 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» progress	integer	true	none		none

PUT 修改考试类型_logged h2

PUT /info/examtype/

修改考试类型

Body 请求参数

```
1 | {  
2 |     "examtype": "string"  
3 | }
```

请求参数

h3

名称	位置	类型	必选	说明
body	body	object	否	none
» examtype	body	string	否	none

返回示例

成功

```
1 | {  
2 |     "msg": "ok"  
3 | }
```

返回结果

h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline
201	<u>Created</u>	成功	Inline

返回数据结构

h3

PUT 签到天数加一

PUT /info/{pk}/signin/

Body 请求参数

```
1 | {  
2 |     "signin": 9009  
3 | }
```

请求参数

h3

名称	位置	类型	必选	说明
pk	path	string	是	none
body	body	object	否	none
» signin	body	integer	是	none

返回示例

成功

```

1 {
2   "uid": 2,
3   "nickname": "testNickname",
4   "name": "create0000_pyte_r",
5   "signin": 9013,
6   "examtype": "4",
7   "examdate": "1970-01-01",
8   "signupdate": "1970-01-01"
9 }
```

```

1 {
2   "uid": 1,
3   "nickname": "testNickname",
4   "name": "Ronald Taylor",
5   "status": 0,
6   "signin": 28,
7   "openid": null,
8   "examdate": "2020-01-13",
9   "examtype": "4",
10  "signupdate": "1970-01-01",
11  "schedule": 65
12 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» uid	integer	true	none		none

名称	类型	必选	约束	中文名	说明
» nickname	string	true	none		none
» name	string	true	none		none
» signin	integer	true	none		none
» examtype	string	true	none		none
» examdate	string	true	none		none
» signupdate	string	true	none		none

POST 问题反馈 h2

POST /user/feedback/{id}

请求参数 h3

名称	位置	类型	必选	说明
id	path	string	是	none

返回示例

成功

```

1 | {
2 |   "data": "minim laborum dolor commodo"
3 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明

名称	类型	必选	约束	中文名	说明
» data	string	true	none		none

GET 当前用户超过多少用户(排名占比) h2

GET /info/{pk}/rank/

指标为坚持学习的天数

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none

返回示例

成功

```

1 {
2   "rank": 42,
3   "percentage": 0.27450980392156865,
4   "singin": 28
5 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

返回数据结构 h3

GET 考试时间倒计时 h2

GET /timer-days/

- 原本通过url参数{uid}来传递
- 现在,采用登录状态后的session字段来获取用户信息,在后台自动的处理uid.

返回示例

成功

```
1 | {  
2 |     "days_remain": 432  
3 | }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» days_remain	integer	true	none		none

GET 查询学习进度_separate h2

GET /info/{pk}/progress/{examtype}/

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none
examtype	path	string	是	none

返回示例

成功

```
1 | {  
2 |     "progress": 448  
3 | }
```

```
1 | {  
2 |     "user": 1,
```

```
3     "examtype": "neep",
4     "progress": 4
5 }
```

返回结果 h3

状态码	状态码含义	说明	数据模型
200	<u>OK</u>	成功	Inline

返回数据结构 h3

状态码 200

名称	类型	必选	约束	中文名	说明
» progress	integer	true	none		none

用户/review h1

GET 用户的全局复习列表(推荐复习)_separate h2

GET /info/{pk}/review/global/{examtype}

- 主要依据是熟练度
- 不限制初次学习的时间

请求参数 h3

名称	位置	类型	必选	说明
pk	path	string	是	none
examtype	path	string	是	none

返回示例

成功

```

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,
22     "last_see_datetime": "2022-05-15T15:26:30.770387Z",
23     "familiarity": 4,
24     "user": 4,
25     "user_name": "testScriptUser",
26     "wid": 4,
27     "spelling": "above"
28 },
29 {
30     "id": 14,
31     "last_see_datetime": "2022-05-15T15:29:45.367310Z",
32     "familiarity": 1,
33     "user": 4,
34     "user_name": "testScriptUser",
35     "wid": 14,
36     "spelling": "accidental"
37 }
38 ]

```

[返回结果](#) [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

[返回数据结构](#) [h3](#)

状态码 200

名称	类型	必选	约束	中文名	说明
<i>anonymous</i>	[Study]	false	none		none

名称	类型	必选	约束	中文名	说明
» id	integer	false	none		none
» wid	integer	false	none		none
» last_see_datetime	string	false	none		none
» familiarity	integer	false	none		none
» user	integer	false	none		none
» examtype	string	false	none		none

枚举值 h4

属性	值
examtype	4
examtype	6
examtype	8

用户/review/login_review h1

GET 用户的全局复习列表(推荐复习)_aggregate h2

GET /info/review/global/

- 主要依据是熟练度
- 不限制初次学习的时间

请求参数 h3

名称	位置	类型	必选	说明
examtype	query	string	否	none

返回示例

成功

```

1 | [
2 | {
3 |   "id": 2,

```

```
4     "last_see_datetime": "2022-05-15T14:52:16.796170Z",
5     "familiarity": 4,
6     "user": 3,
7     "wid": 4
8 },
9 {
10    "id": 3,
11    "last_see_datetime": "2022-05-15T12:26:34.939649Z",
12    "familiarity": 4,
13    "user": 3,
14    "wid": 4
15 },
16 {
17    "id": 11,
18    "last_see_datetime": "2022-05-15T15:00:50.451619Z",
19    "familiarity": 1,
20    "user": 3,
21    "wid": 2
22 },
23 {
24    "id": 21,
25    "last_see_datetime": "2022-05-16T06:54:04.461780Z",
26    "familiarity": 4,
27    "user": 3,
28    "wid": 4
29 },
30 {
31    "id": 22,
32    "last_see_datetime": "2022-05-20T00:18:22.888060Z",
33    "familiarity": 4,
34    "user": 3,
35    "wid": 4
36 },
37 {
38    "id": 23,
39    "last_see_datetime": "2022-05-20T00:21:41.204381Z",
40    "familiarity": 4,
41    "user": 3,
42    "wid": 4
43 },
44 {
45    "id": 24,
46    "last_see_datetime": "2022-05-20T00:43:53.938043Z",
47    "familiarity": 4,
48    "user": 3,
49    "wid": 4
50 },
51 {
52    "id": 25,
53    "last_see_datetime": "2022-05-20T00:44:47.862966Z",
54    "familiarity": 4,
55    "user": 3,
56    "wid": 4
57 },
58 {
59    "id": 26,
60    "last_see_datetime": "2022-05-20T00:45:31.044283Z",
61    "familiarity": 4,
```

```
62     "user": 3,
63     "wid": 4
64   },
65   {
66     "id": 27,
67     "last_see_datetime": "2022-05-20T00:55:46.312834Z",
68     "familiarity": 4,
69     "user": 3,
70     "wid": 4
71   },
72   {
73     "id": 28,
74     "last_see_datetime": "2022-05-20T01:00:50.078922Z",
75     "familiarity": 4,
76     "user": 3,
77     "wid": 4
78   },
79   {
80     "id": 29,
81     "last_see_datetime": "2022-05-20T01:04:56.426746Z",
82     "familiarity": 4,
83     "user": 3,
84     "wid": 4
85   },
86   {
87     "id": 30,
88     "last_see_datetime": "2022-05-20T01:29:39.040639Z",
89     "familiarity": 4,
90     "user": 3,
91     "wid": 4
92   },
93   {
94     "id": 31,
95     "last_see_datetime": "2022-05-20T11:43:27.431639Z",
96     "familiarity": 4,
97     "user": 3,
98     "wid": 4
99   }
100 ]
```

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

```
19     "wid": 645,
20     "spelling": "cap"
21 },
22 {
23     "id": 10,
24     "last_see_datetime": "2022-06-06T08:31:50.723538Z",
25     "examtype": "4",
26     "familiarity": 1,
27     "user": 112,
28     "user_name": "cxxu",
29     "wid": 85,
30     "spelling": "adolescent"
31 },
32 {
33     "id": 11,
34     "last_see_datetime": "2022-06-09T03:34:55.600855Z",
35     "examtype": "4",
36     "familiarity": 2,
37     "user": 112,
38     "user_name": "cxxu",
39     "wid": 1,
40     "spelling": "abandon"
41 },
42 {
43     "id": 14,
44     "last_see_datetime": "2022-06-09T09:44:42.778349Z",
45     "examtype": "4",
46     "familiarity": 0,
47     "user": 112,
48     "user_name": "cxxu",
49     "wid": 41,
50     "spelling": "account"
51 },
52 {
53     "id": 18,
54     "last_see_datetime": "2022-06-09T11:50:34.181141Z",
55     "examtype": "4",
56     "familiarity": 0,
57     "user": 112,
58     "user_name": "cxxu",
59     "wid": 11,
60     "spelling": "abrupt"
61 },
62 {
63     "id": 20,
64     "last_see_datetime": "2022-06-09T11:51:30.646249Z",
65     "examtype": "4",
66     "familiarity": 0,
67     "user": 112,
68     "user_name": "cxxu",
69     "wid": 53,
70     "spelling": "acquaintance"
71 }
72 ]
```

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline

[返回数据结构](#) [h3](#)

状态码 200

名称	类型	必选	约束	中文名	说明
<i>anonymous</i>	[Study]	false	none		none
» id	integer	false	none		none
» wid	integer	false	none		none
» last_see_datetime	string	false	none		none
» familiarity	integer	false	none		none
» user	integer	false	none		none
» examtype	string	false	none		none

[枚举值](#) [h4](#)

属性	值
examtype	4
examtype	6
examtype	8

[GET 获取最近学过的单词列表_aggregate](#) [h2](#)

GET /info/review/recently/

- 目前支持path参数(后期打算废弃)
- 正在开发query参数支持(更加灵活)

[请求参数](#) [h3](#)

名称	位置	类型	必选	说明
unit	query	string	否	none

名称	位置	类型	必选	说明
value	query	string	否	如果长时间没有学习,给列表可能为空(尤其是value不够大的时候)
examtype	query	string	否	考试类型

[返回示例](#)

成功

```

1  [
2  {
3      "id": 6,
4      "last_see_datetime": "2022-06-06T05:57:44.213902Z",
5      "examtype": "4",
6      "familiarity": 3,
7      "user": 112,
8      "user_name": "cxxu",
9      "wid": 65,
10     "spelling": "actress"
11 },
12 {
13     "id": 7,
14     "last_see_datetime": "2022-06-06T05:58:40.866844Z",
15     "examtype": "4",
16     "familiarity": 3,
17     "user": 112,
18     "user_name": "cxxu",
19     "wid": 645,
20     "spelling": "cap"
21 },
22 {
23     "id": 10,
24     "last_see_datetime": "2022-06-06T08:31:50.723538Z",
25     "examtype": "4",
26     "familiarity": 1,
27     "user": 112,
28     "user_name": "cxxu",
29     "wid": 85,
30     "spelling": "adolescent"
31 },
32 {
33     "id": 11,
34     "last_see_datetime": "2022-06-09T03:34:55.600855Z",
35     "examtype": "4",
36     "familiarity": 2,
37     "user": 112,
38     "user_name": "cxxu",
39     "wid": 1,
40     "spelling": "abandon"

```

```

41 },
42 {
43     "id": 14,
44     "last_see_datetime": "2022-06-09T09:44:42.778349Z",
45     "examtype": "4",
46     "familiarity": 0,
47     "user": 112,
48     "user_name": "cxxu",
49     "wid": 41,
50     "spelling": "account"
51 },
52 {
53     "id": 18,
54     "last_see_datetime": "2022-06-09T11:50:34.181141Z",
55     "examtype": "4",
56     "familiarity": 0,
57     "user": 112,
58     "user_name": "cxxu",
59     "wid": 11,
60     "spelling": "abrupt"
61 },
62 {
63     "id": 20,
64     "last_see_datetime": "2022-06-09T11:51:30.646249Z",
65     "examtype": "4",
66     "familiarity": 0,
67     "user": 112,
68     "user_name": "cxxu",
69     "wid": 53,
70     "spelling": "acquaintance"
71 }
72 ]

```

[返回结果](#) [h3](#)

状态码	状态码含义	说明	数据模型
200	OK	成功	Inline
404	Not Found	记录不存在	Inline

[返回数据结构](#) [h3](#)

状态码 200

名称	类型	必选	约束	中文名	说明
<i>anonymous</i>	[Study]	false	none		none
» id	integer	false	none		none
» wid	integer	false	none		none

名称	类型	必选	约束	中文名	说明
» last_see_datetime	string	false	none		none
» familiarity	integer	false	none		none
» user	integer	false	none		none
» examtype	string	false	none		none

枚举值 h4

属性	值
examtype	4
examtype	6
examtype	8

数据模型 h1

Star h2

```

1 | {
2 |   "id": 0,
3 |   "spelling": "string",
4 |   "user": 0
5 |
6 |

```

属性 h3

名称	类型	必选	约束	中文名	说明
id	integer	true	none		收藏条目id
spelling	string	true	none		单词拼写
user	integer	true	none		用户id

Study h2

```
1 | {
2 |   "id": 0,
3 |   "wid": 0,
4 |   "last_see_datetime": "string",
5 |   "familiarity": 0,
6 |   "user": 0,
7 |   "examtype": "4"
8 | }
9 |
```

属性 h3

名称	类型	必选	约束	中文名	说明
id	integer	false	none		none
wid	integer	false	none		none
last_see_datetime	string	false	none		none
familiarity	integer	false	none		none
user	integer	false	none		none
examtype	string	false	none		none

枚举值 h4

属性	值
examtype	4
examtype	6
examtype	8

UserUpdate h2

```
1 | {
2 |   "name": "string",
3 |   "examtype": "string",
4 |   "examdate": "string"
5 | }
```

属性 h3

名称	类型	必选	约束	中文名	说明
name	string	false	none		none
examtype	string	false	none		none
examdate	string	false	none		none

WordMatcher h2

```

1 {
2   "id": 0,
3   "spelling": "string",
4   "char_set_str": "string"
5 }
6

```

属性 h3

名称	类型	必选	约束	中文名	说明
id	integer	true	none		none
spelling	string	true	none		none
char_set_str	string	true	none		none

WordNote h2

```

1 {
2   "user": 0,

```

```
3   "spelling": "string",
4   "content": "string",
5   "difficulty_rate": 0
6 }
7 }
```

属性

h3

名称	类型	必选	约束	中文名	说明
user	integer	true	none		none
spelling	string	true	none		none
content	string	true	none		none
difficulty_rate	integer	true	none		none

ExamType

h2

```
1 {
2   "examtype": "string"
3 }
4 }
```

属性

h3

名称	类型	必选	约束	中文名	说明
examtype	string	true	none		none

DRF_List

h2

```
1 {
2   "count": 0,
3   "next": "string",
4   "previous": "string",
5   "results": [
```

```
6   "string"
7 ]
8 }
9 }
```

属性 h3

名称	类型	必选	约束	中文名	说明
count	integer	false	none		none
next	string	false	none		none
previous	string	false	none		none
results	[string]	false	none		none

WordReqSum h2

```
1 {
2   "examtype": "cet4",
3   "sum": 0
4 }
5 }
```

属性 h3

名称	类型	必选	约束	中文名	说明
examtype	string	true	none		4/6/8(neep)
sum	integer	true	none		none

枚举值 h4

属性	值
examtype	cet4

属性	值
examtype	cet6
examtype	neep

WordUltra h2

```

1  {
2    "wordSpelling": "string",
3    "phonetic": "string",
4    "basicExplain": "string",
5    "webMeaning": [
6      "string"
7    ],
8    "forms": {
9      "pl": "string",
10     "past": null,
11     "pastParticiple": null,
12     "presentParticiple": null
13   }
14 }
15

```

属性 h3

名称	类型	必选	约束	中文名	说明
wordSpelling	string	false	none		none
phonetic	string	false	none		none
basicExplain	string	false	none		none
webMeaning	[string]	false	none		none
forms	object	false	none		none
» pl	string	true	none		none
» past	null	true	none		none
» pastParticiple	null	true	none		none
» presentParticiple	null	true	none		none

User h2

```
1  {
2    "uid": 0,
3    "name": "string",
4    "signin": 0,
5    "examtype": "string",
6    "examdate": "string",
7    "signupdate": "string"
8  }
9
```

属性 h3

名称	类型	必选	约束	中文名	说明
uid	integer	true	none		none
name	string	true	none		none
signin	integer	true	none		none
examtype	string	true	none		none
examdate	string	true	none		none
signupdate	string	true	none		none

UserSignUp h2

```
1  {
2    "name": "string",
3    "examtype": "string",
4    "examdate": "string"
5  }
6
```

属性 h3

名称	类型	必选	约束	中文名	说明
name	string	true	none		none
examtype	string	false	none		none

名称	类型	必选	约束	中文名	说明
examdate	string	true	none		none

WordNote1 h2

```

1 {
2   "uid": 0,
3   "spelling": "string",
4   "content": "string",
5   "difficulty_rate": 0
6 }
7

```

属性 h3

名称	类型	必选	约束	中文名	说明
uid	integer	true	none		none
spelling	string	true	none		none
content	string	true	none		none
difficulty_rate	integer	false	none		none

User_login h2

```

1 {
2   "account": "string",
3   "password": "string"
4 }
5

```

属性 h3

名称	类型	必选	约束	中文名	说明
account	string	true	none		none

名称	类型	必选	约束	中文名	说明
password	string	true	none		none

UserSignupPassword h2

```

1 {
2   "name": "string",
3   "examtype": "string",
4   "examdate": "string",
5   "password": "string"
6 }
7

```

属性 h3

名称	类型	必选	约束	中文名	说明
name	string	true	none		none
examtype	string	false	none		none
examdate	string	true	none		none
password	string	true	none		none

StudyUpdate h2

```

1 {
2   "wid": 0,
3   "familiarity": 0,
4   "user": 0,
5   "examtype": "4"
6 }
7

```

属性 h3

名称	类型	必选	约束	中文名	说明

名称	类型	必选	约束	中文名	说明
wid	integer	false	none		none
familiarity	integer	false	none		none
user	integer	false	none		none
examtype	string	false	none		none

枚举值 h4

属性	值
examtype	4
examtype	6
examtype	8

Word h2

```

1  {
2    "wid": 0,
3    "spelling": "string",
4    "phonetic": "ə'pi:l",
5    "plurality": "string",
6    "thirdpp": "string",
7    "present_participle": "string",
8    "past_tense": "string",
9    "past_participle": "string",
10   "explains": "string"
11 }
12

```

属性 h3

名称	类型	必选	约束	中文名	说明
wid	integer	true	none		none
spelling	string	true	none		none
phonetic	string	false	none		none
plurality	string	false	none		none

名称	类型	必选	约束	中文名	说明
thirdpp	string	false	none		none
present_participle	string	false	none		none
past_tense	string	false	none		none
past_participle	string	false	none		none
explains	string	false	none		none

WordSearchHistory h2

```

1  {
2    "user": 0,
3    "spelling": "string"
4  }
5

```

属性 h3

名称	类型	必选	约束	中文名	说明
user	integer	true	none		none
spelling	string	true	none		none

Progress h2

```

1  {
2    "examtype": "string",
3    "progress": 0
4  }
5

```

属性 h3

名称	类型	必选	约束	中文名	说明

名称	类型	必选	约束	中文名	说明
examtype	string	true	none		none
progress	integer	true	none		none

SearchHistory h2

```

1  {
2    "id": 0,
3    "spelling": "string",
4    "user": 0
5  }
6

```

属性 h3

名称	类型	必选	约束	中文名	说明
id	integer	true	none		none
spelling	string	true	none		none
user	integer	true	none		none

五、数据库设计 [徐超信] h1

数据库是本项目的重要内容,共有9张表

持久化数据 h2

- 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

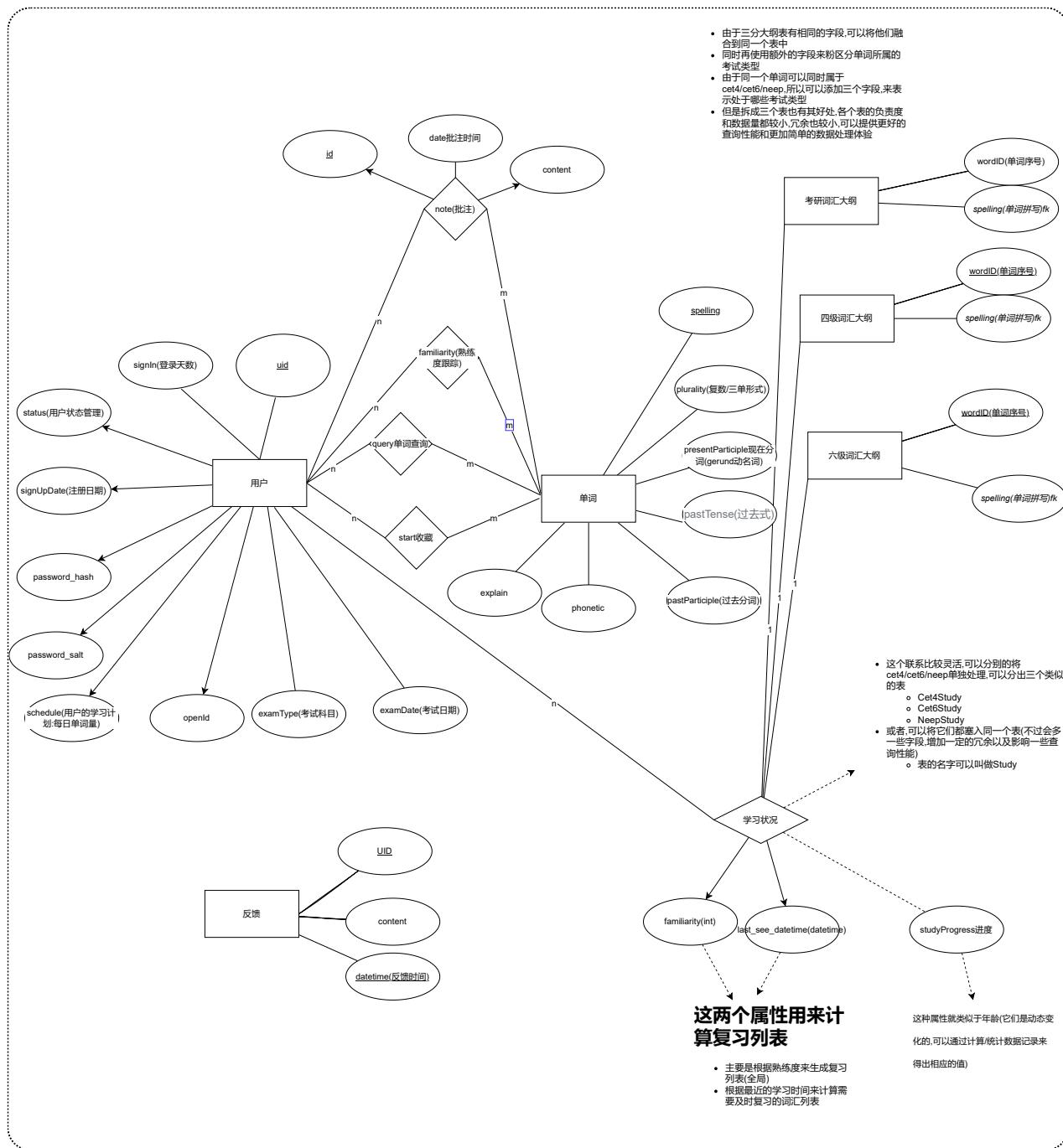


图5.1 ER关系图

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,程序处理起来比方便,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

```
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

实现技术:vant app 组件+微信小程序

6.1 登录退出功能的实现 h2

授权登录和退出



图6.1

- 点击授权登录按钮, 可以获取微信头像和昵称。
- 使用了微信官方文档的 [API:wx.getUserProfile](#) 实现授权登录功能。
- 登录前只能看到反馈建议和在线客服。
- 为了能够保持登录或者退出登录状态, 用了微信官方文档API:wx.setStorageSync进行数据缓存, 用wx.getStorageSync读取缓存。
- 申请获取头像和昵称
- 授权登录后可以看到累计打卡天数、ddl时间、我的收藏和历史记录。
- 点击"退出登录"按钮退出登录。

- 退出登录后看清空数据

6.2 查词功能的实现

h2

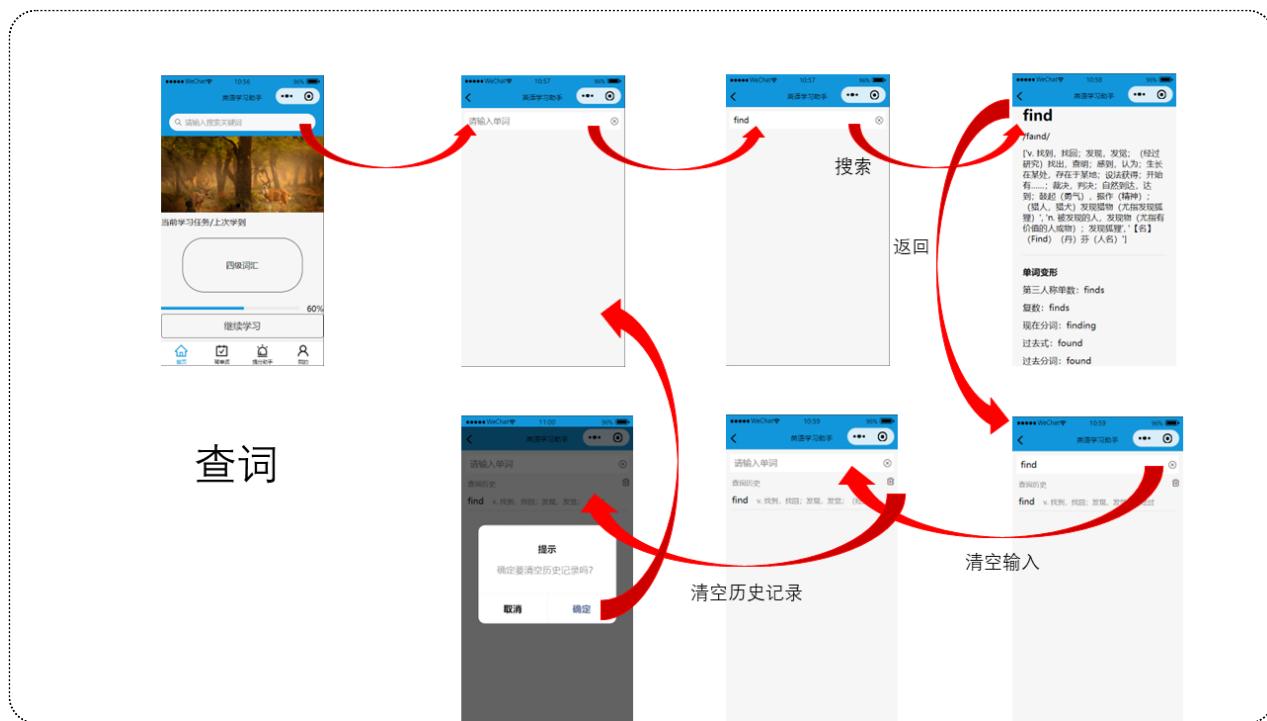


图6.2

- 如果输入123，因为单词数据库中没有该内容因此显示未找到相关内容
- 如果输入一个正确的单词，然后回车或者点击搜索
- 显示单词、音标和释义。
- 如果单词没有变形，则不会显示单词变形内容。
- 如果搜索的单词有变形，则会显示单词变形内容。
- 单词变形内容包括：第三人称单数、复数、现在分词、过去式、过去分词等。
- 点击搜索框右侧的清除按钮，可以清空搜索框
- 点击查询历史右侧的垃圾桶图标，会提示是否清空历史记录。选择“是”，历史记录将被清空。

6.3 学单词功能的实现

h2

背单词



图6.3.1

- 最上方是当前已背单词书/该词书单词总数。
- 中间是单词、音标、释义。如果没有单词变形则不显示，如果有则显示。
- 右滑实现翻页，翻到第一页再往前翻会提示
- 如果滑到第一页还左滑，那么会提示已经到第一个单词了
- 可以查看后端统计的难度等级

发表批注



图6.3.2

- 点击“添加我的批注”按钮添加批注。
- 输入框会提示“分享你的想法...”。
- 如果输入框内容为空，则发表按钮被禁用；如果输入了内容，则发表按钮被启用。
- 点击发表按钮后，回到背单词页面。
- 发表批注后可以在批注区看到自己的微信头像、微信昵称和评论内容。

6.4 查考纲功能的实现 h2

切换考纲



图6.4

- 记忆模式可以切换四级大纲、六级大纲、考研大纲
- 选择相应的大纲后可以展示相应的大纲

6.5 计算ddl(deadline)功能的实现 h2

计算ddl



图6.5

- 点击日期可以切换考试日期。
- 如果还未选择考试日期，显示null天。
- 切换考试日期后自动计算ddl。

6.6 反馈建议&在线客服功能的实现

h2

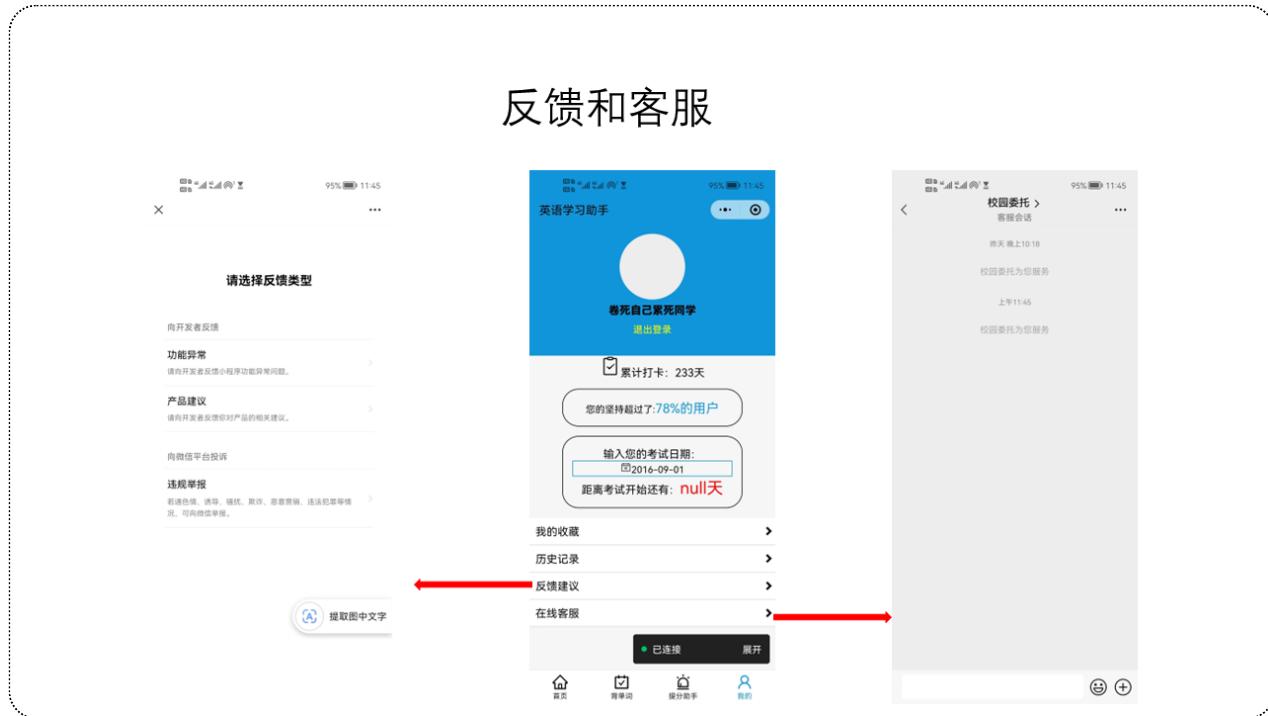


图6.6

- 点击反馈建议可以反馈信息
- 点击在线客服可以与客服会话

七、英语助手后端的实现[徐超信]

h1

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
 -

GET http://123.56.72.67:8000/word/dict/{spelling}

发送 保存 删除

获取单词释义 (成功(eg.word:apply))

Params 2 Body Headers Cookies Auth 前置 后置 设置

Query 参数

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

Path 参数

<input checked="" type="checkbox"/>	参数名	参数值
<input checked="" type="checkbox"/>	spelling	apply

Body Cookie Header 11 控制台 实际请求

Pretty Raw Preview utf8

```

1 {
2   "wid": 232,
3   "spelling": "apply",
4   "phonetic": "əˈplaɪ",
5   "plurality": "NULL",
6   "thirdpp": "applies",
7   "present_participle": "applying",
8   "past_tense": "applied",
9   "past_participle": "applied",
10  "explains": "[v. 申请; 涂, 敷; 施加, 实施; 应用, 运用; 踩(刹车); 适用, 适合; 指称]"
11 }

```

CSDN @xuchaoxin1375

图7.2.1:接口效果

7.2 学单词功能的实现 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

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

Body Cookie 1 Header 10 控制 ...

Pretty utf8

```

1 {
2   "count": 4500,
3   "next": "http://127.0.0.1:8000/word/cet4/?page=2",
4   "previous": null,
5   "results": [
6     {
7       "wordorder": 1,
8       "spelling": "abandon"
9     },
10    {
11      "wordorder": 2,
12      "spelling": "abatement"
13    },
14    {
15      "wordorder": 3,
16      "spelling": "abdomen"
17    }
]

```

CSDN @xuchaoxin1375

图7.2.2

- 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

```

1 {
2   "spelling": "abandon",
3   "avg_familiarity": 3,
4   "validity": true,
5   "msg": "查询成功"
6 }
```

Pretty Raw Preview utf8 ↻ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ...

CSDN @xuchaoxin1375

图7.2.3

- 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 h2

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				

图7.3.1

- 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	...	</>
Query 参数					
<input checked="" type="checkbox"/> 参数名	参数值				
<input checked="" type="checkbox"/> unit	days				
<input checked="" type="checkbox"/> value	7				
<input checked="" type="checkbox"/> examtype	4				

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": "2022-06-06T05:57:44.213902Z", 5 "examtype": "4", 6 "familiarity": 3, 7 "user": 112, 8 "user_name": "cxxu", 9 "wid": 65, 10 "spelling": "actress" 11 }, 12 { 13 "id": 7, 14 "last_see_datetime": "2022-06-06T05:58:40.866844Z", 15 "examtype": "4", 16 "familiarity": 3, 17 "user": 112, 18 "user_name": "cxxu", 19 "wid": 645, 20 "spelling": "can" </pre>				
CSDN @xuchaoxin1375				

图7.3.2

- 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

随机抽查一组单词 G (sizeable) (成功(cet6))

Params 2	Body	Headers	...					
Path 参数								
<table border="1"> <thead> <tr> <th>参数名</th> <th>参数值</th> </tr> </thead> <tbody> <tr> <td>examtype</td> <td>cet6</td> </tr> <tr> <td>size</td> <td>55</td> </tr> </tbody> </table>	参数名	参数值	examtype	cet6	size	55		
参数名	参数值							
examtype	cet6							
size	55							
		Body	Cookie 1 Header 10 控制 ...					
		Pretty ▾	utf8 ▾					
		<input type="button"/>	<input type="button"/>					
		<input type="button"/>	<input type="button"/>					
			...					

```

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

```

图7.3.3

- 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 模糊查询词的实现 h2

- 模糊查询词(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	absolutu

▪ 图7.4.1

- 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}

发送

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 Cookie Header 10 控制台 实际请求

Pretty Raw Preview utf8 ▾

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

CSDN @xuchaoxin1375

图7.4.2

eg1: h3

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

发送 保存

模糊匹配单词(拼写形近词) (成功(acquietenace))

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	acquietenace
<input checked="" type="checkbox"/>	start_with	

Body Cookie Header 10 控制台 实际请求

Pretty Raw Preview utf8 ▾

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

CSDN @xuchaoxin1375

图7.4.3

eg2 h3

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

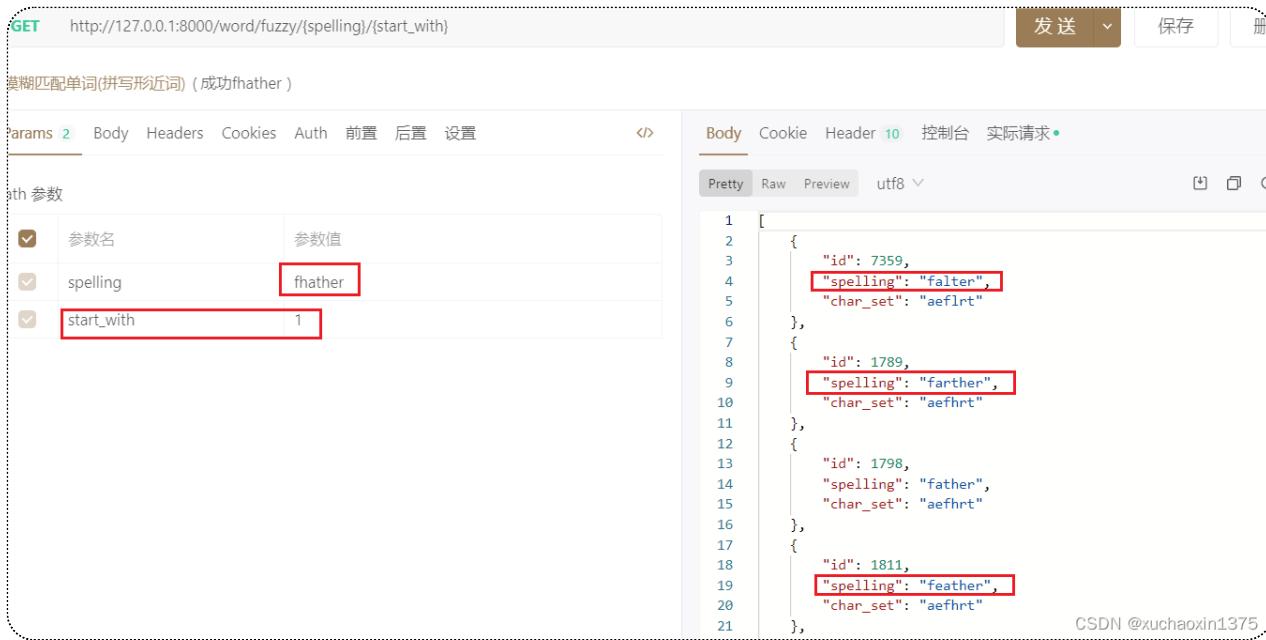


图7.4.3

7.5 用户中心 h2

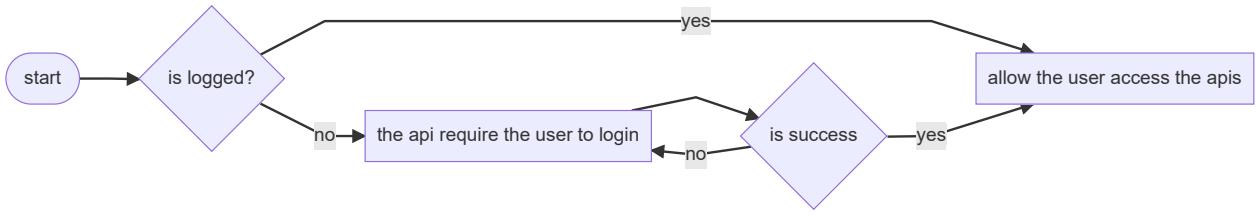
登录功能方案选型: h3

[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

登录流程 h4



八、系统测试 h1

8.1 单元测试 h2

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,或者在测试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

coverage.py提供的测试覆盖率统计报告服务

h4

```
PS D:\repos\ELA\backEnd> coverage run manage.py test word.tests --keepdb
Found 10 test(s).
Using existing test database for alias 'default'...
System check identified no issues (0 silenced).
```

图8.1通过coverage收集测试运行的信息(以项目中的word模块为例)

Name	Stmts	Miss	Cover
backEnd__init__.py	0	0	100%
backEnd\settings.py	27	1	96%
backEnd\urls.py	10	0	100%
blogs__init__.py	0	0	100%
blogs\migrations__init__.py	0	0	100%

图8.2开始统计测试

user\urls.py	17	0	100%
user\views__init__.py	0	0	100%
user\views\history.py	12	1	92%
user\views\login.py	53	35	34%
user\views\star.py	50	29	42%
word\tests__init__.py	0	0	100%
word\tests\test_forms.py	0	0	100%
word\tests\test_models.py	0	0	100%
word\tests\test_urls.py	51	0	100%
word\tests\test_views.py	0	0	100%
word\urls.py	12	0	100%
word\views.py	168	99	41%
TOTAL	1915	803	58%

图8.3,总结当前的测试覆盖率(58%)

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4个模块的有序测试,所有接口,基本全部通过



图8.2.1通过apifox 进行的批量自动化测试,每个接口可以又多个测试实例



图8.2.2

- 这是通过apifox的套件测试,批量测试接口下的测试用例,并且测试保证顺序
- 报告测试结果

8.3 测试部署及结果 h2

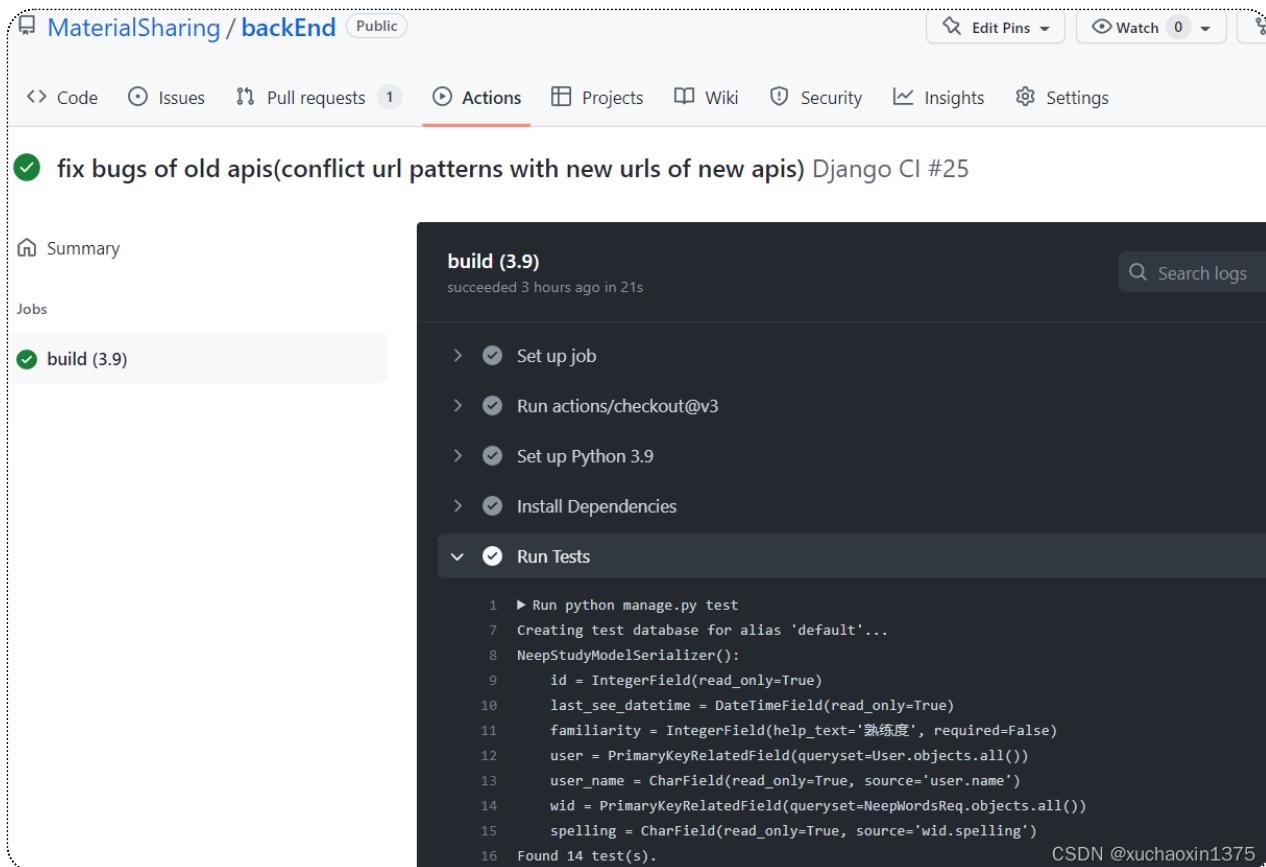


图8.3.1再GitHub上部署并项目并且执行 [Run Tests](#), 报告结果

- 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

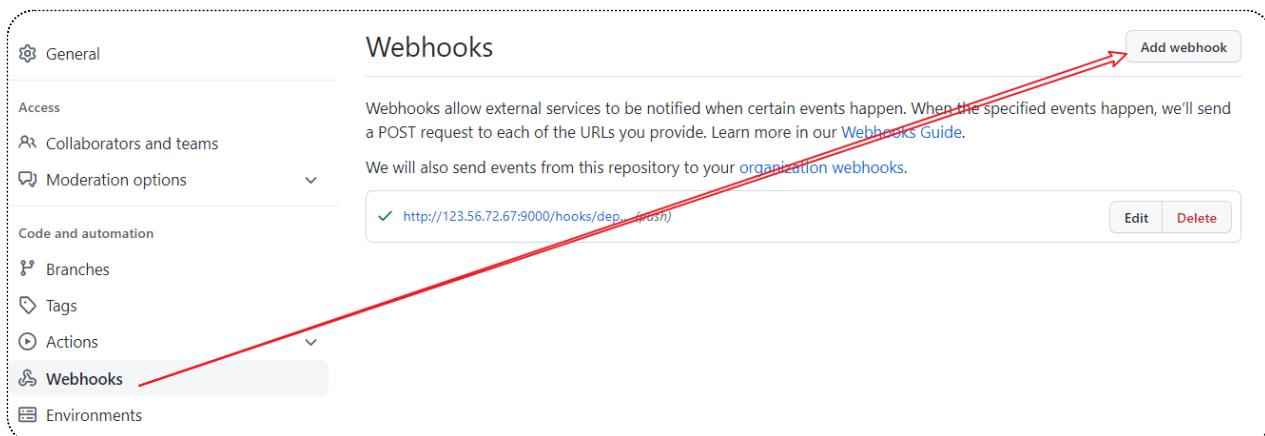


图:github&webhook

- 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
6 hooks.json
7 [webhook] 2022/06/06 16:44:39 found 1 hook(s) in file
8 [webhook] 2022/06/06 16:44:39 loaded: deploy
9 [webhook] 2022/06/06 16:44:39 serving hooks on
http://0.0.0.0:9000/hooks/{id}
```

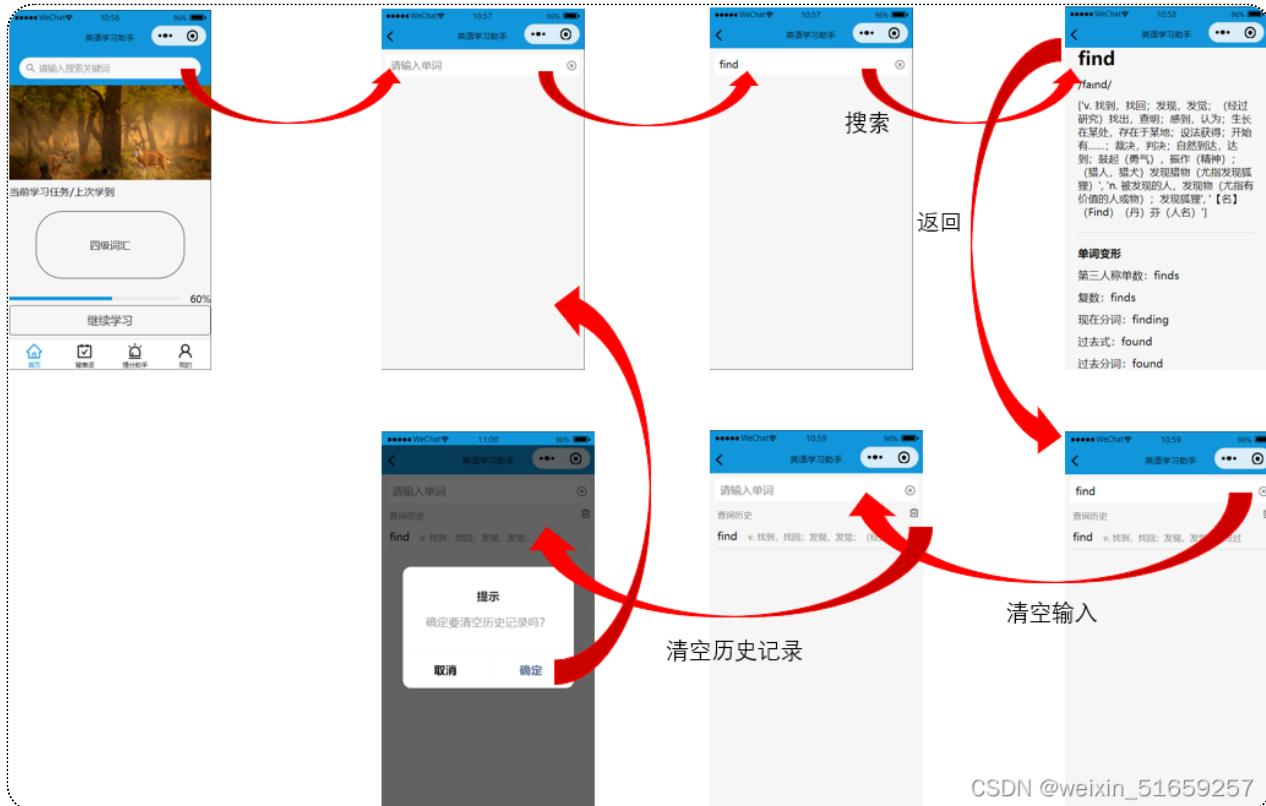
十、功能展示 [潘森森] h1

详细的操作逻辑和实现在第六部分作了解释

- 授权登录与退出



- 查词



CSDN @weixin_51659257

• 背单词，左右滑动实现翻页



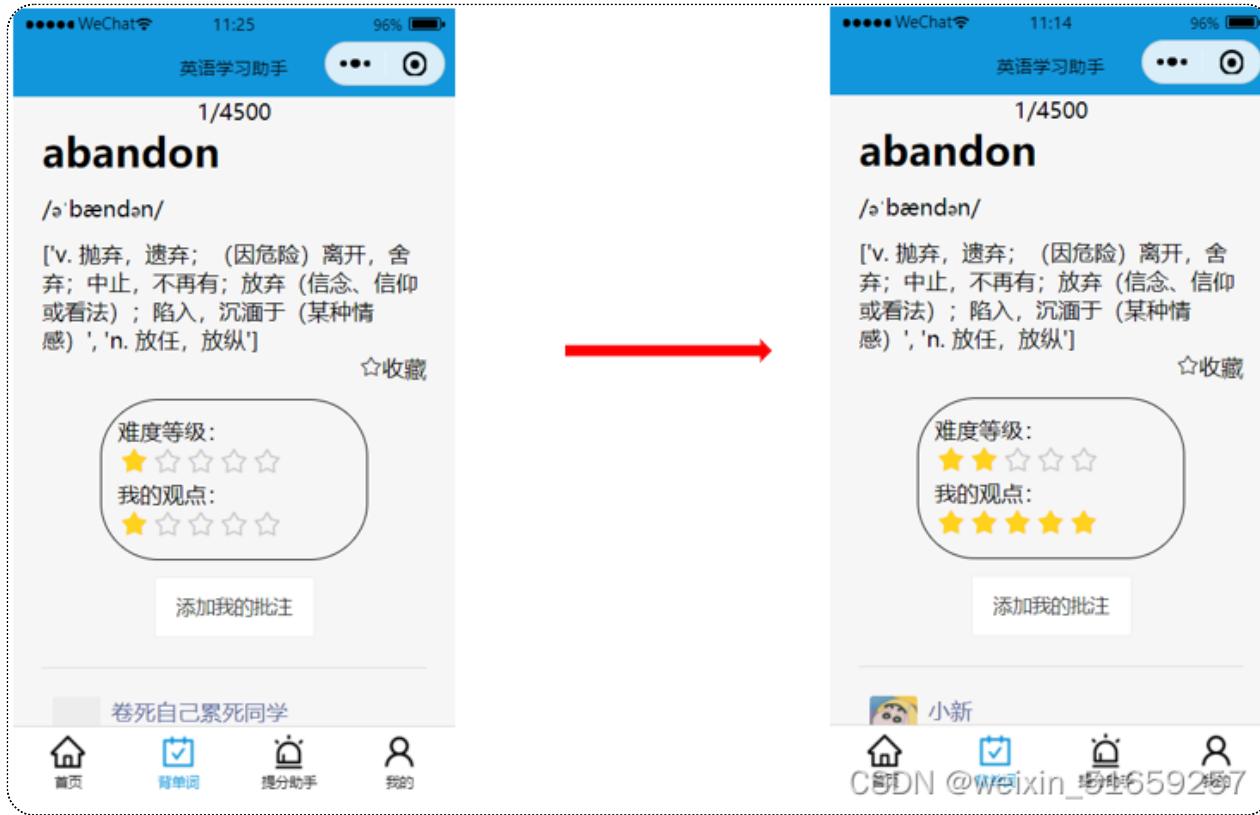
CSDN @weixin_51659257

• 表发表注



CSDN @weixin_51659257

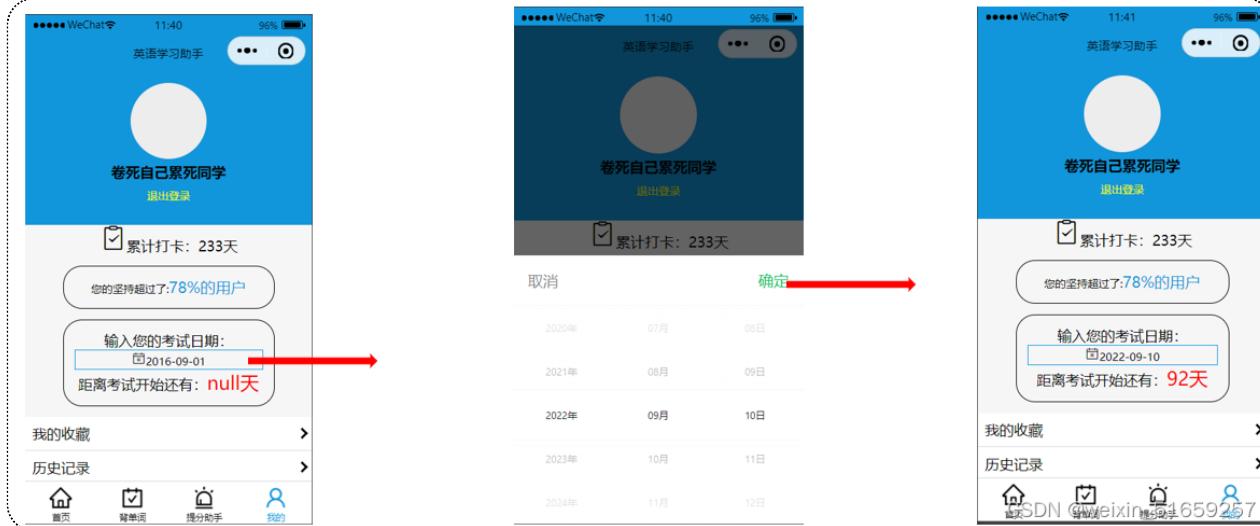
• 难度指标评价



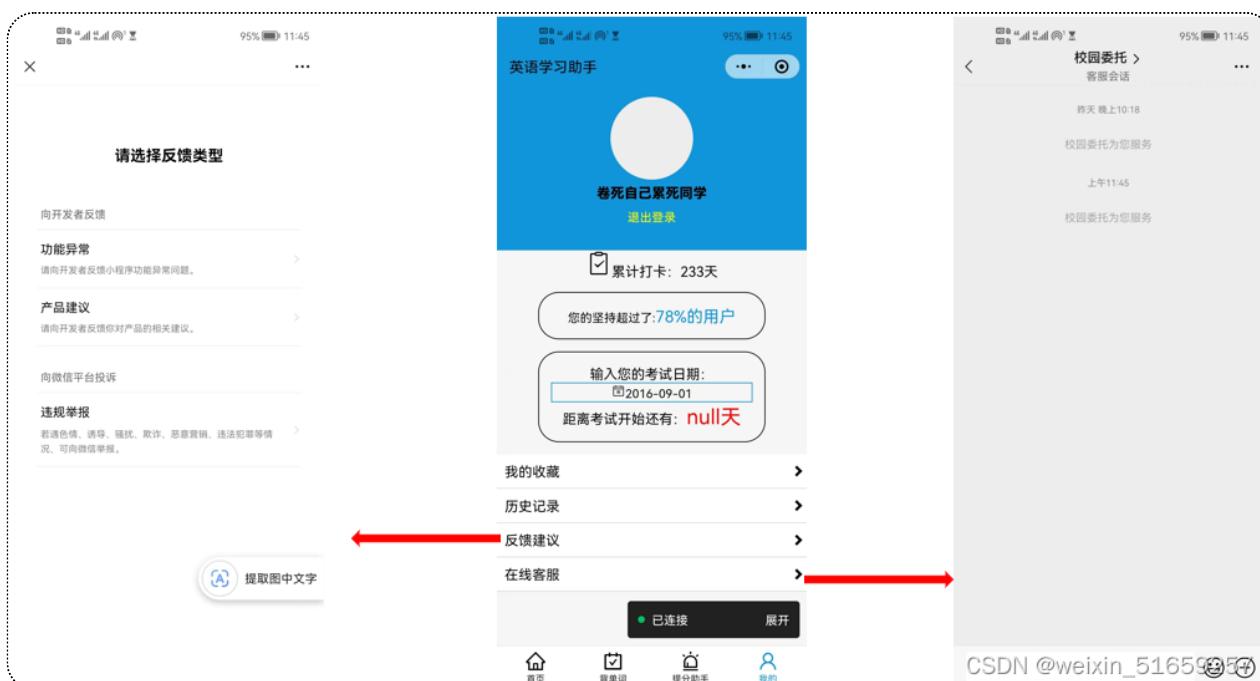
• 切换考纲



• 计算ddl



• 反馈与客服



十一、清单 [徐超信,潘焱森] h1

• 项目github(组织):[MaterialSharing \(github.com\)](https://github.com/MaterialSharing)

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
- 项目ppt:docs/ppt.html (以及ppt.pptx)
- 各种流图和思维导图文件: docs/design/

十二、总结 [徐超信,潘焱森] h1

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,学习并体验了先进的TDD方式开发项目以及各种有利于提高开发&部署效率的技术
- 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.

十三、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](#)
-