

Codehub应用文档

Codehub应用文档

Codehub简介

应用架构

应用功能

功能实现

界面实现

API调用

gpt

正常传输

流式传输

bing

web search

搜索建议

github

搜索仓库

mediawiki

维基百科搜索

StackExchange

文章推荐

api使用示例

获取联想问题

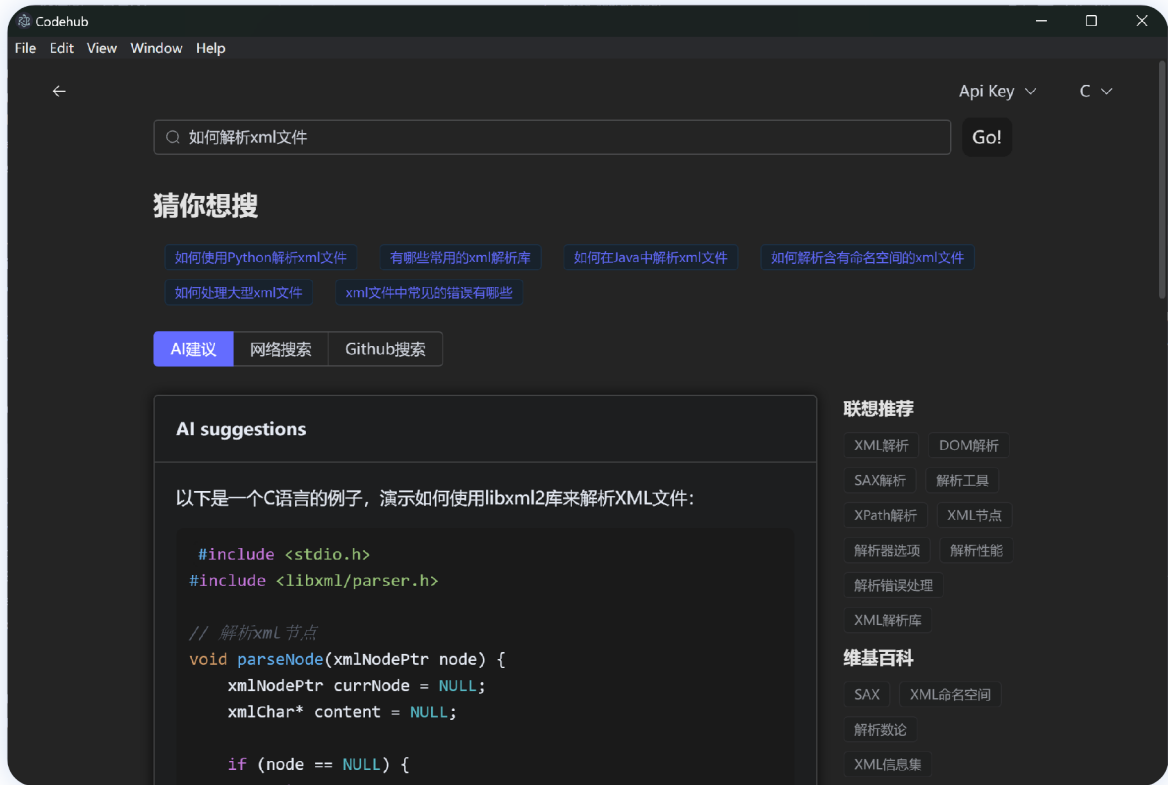
获取web搜索结果

api密钥储存

Codehub简介

Codehub 是一款跨平台帮助编程初学者学习代码的软件，他集成了 [Ai](#)，[Web搜索](#)，[github](#)，[wikipedia](#) 以及 [Stackoverflow](#) 等多方面的信息来源来帮助初学者给出相应的代码样例与相关教程。

- 主要界面



应用架构

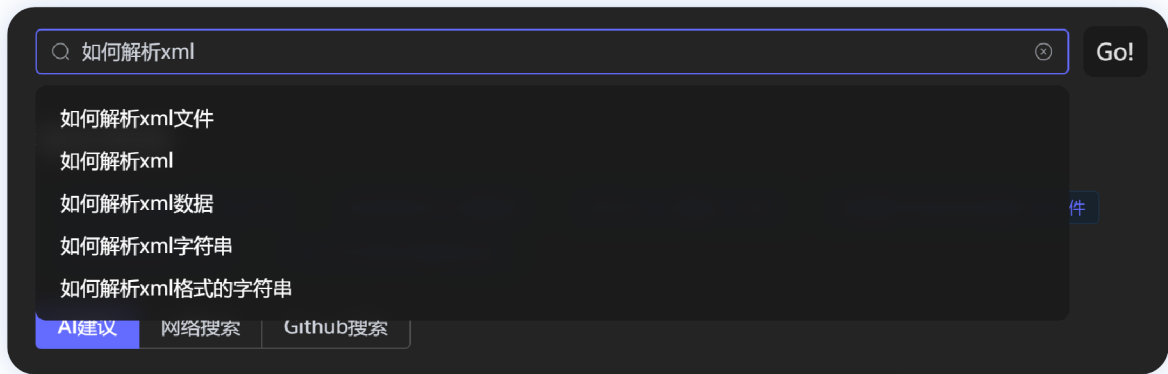
Codehub 采用 electron 与 Vue 实现，electron 是一个基于 nodejs 的跨平台桌面应用开发框架，他可以让我们使用 html , css , js , Vue.js 等前端技术来开发桌面应用。Vue 是一个轻量级的前端框架，他可以让我们更加方便的开发前端应用。

应用功能

Codehub 的主要功能有：问题搜索，Ai代码建议，问题联想，关键词联想，Wiki百科介绍，Web搜索，github仓库搜索，Stackoverflow文章推荐。

• 问题搜索

用户可以在搜索框中键入想要了解的编程问题，同时搜索框会给出问题建议，帮助用户更快捷的搜索。



- **Ai代码建议**

用户可以得到来自Ai对于所搜索问题的相关建议与回答，一般包含若干个代码示例，与相应的代码解释。



- **问题联想**

用户输入问题后，软件会根据用户所输入的问题进行问题联想，给用户更多的搜索建议

猜你想搜

如何使用Python解析xml文件

有哪些常用的xml解析库

如何在Java中解析xml文件

如何解析含有命名空间的xml文件

如何处理大型xml文件

xml文件中常见的错误有哪些

- 关键词联想

用户输入问题后，软件会根据用户所输入的问题进行关键词提取与联想，给出解决用户问题中可能会需要用到的一些关键知识点，点击对应关键词会直接在浏览器中对该关键词进行搜索。

联想推荐

XML解析

DOM解析

SAX解析

解析工具

XPath解析

XML节点

解析器选项

解析性能

解析错误处理

XML解析库

- Wiki百科介绍

Wiki百科介绍会根据用户的问题，直接给出可能用到的知识的百科链接，点击后会直接进入到了百科页面。

维基百科

SAX

XML命名空间

解析数论

XML信息集

解析表达文法

性能分析

XPath

XML外部实体攻击

- **Web搜索**

软件会根据用户输入的问题直接在网络上进行相关搜索，给出相应的链接，点击后会跳转到对应的链接页面。

[AI建议](#)[网络搜索](#)[Github搜索](#)

Web search

C语言解析XML中的数据_c xml解析-CSDN博客

c语言解析xml根据导师的要求，要用C语言解析一个xml文件，用于对升级文件进行合法性判断，进而对软件进行升级。上网搜了一些关于这方面的资料，发现有一些是用C++或Java等语言写的，当然也可以直接下载开源的代码，也可以下载库文件直接使用。

Linux 下C/C++解析XML文件（一） - CSDN博客

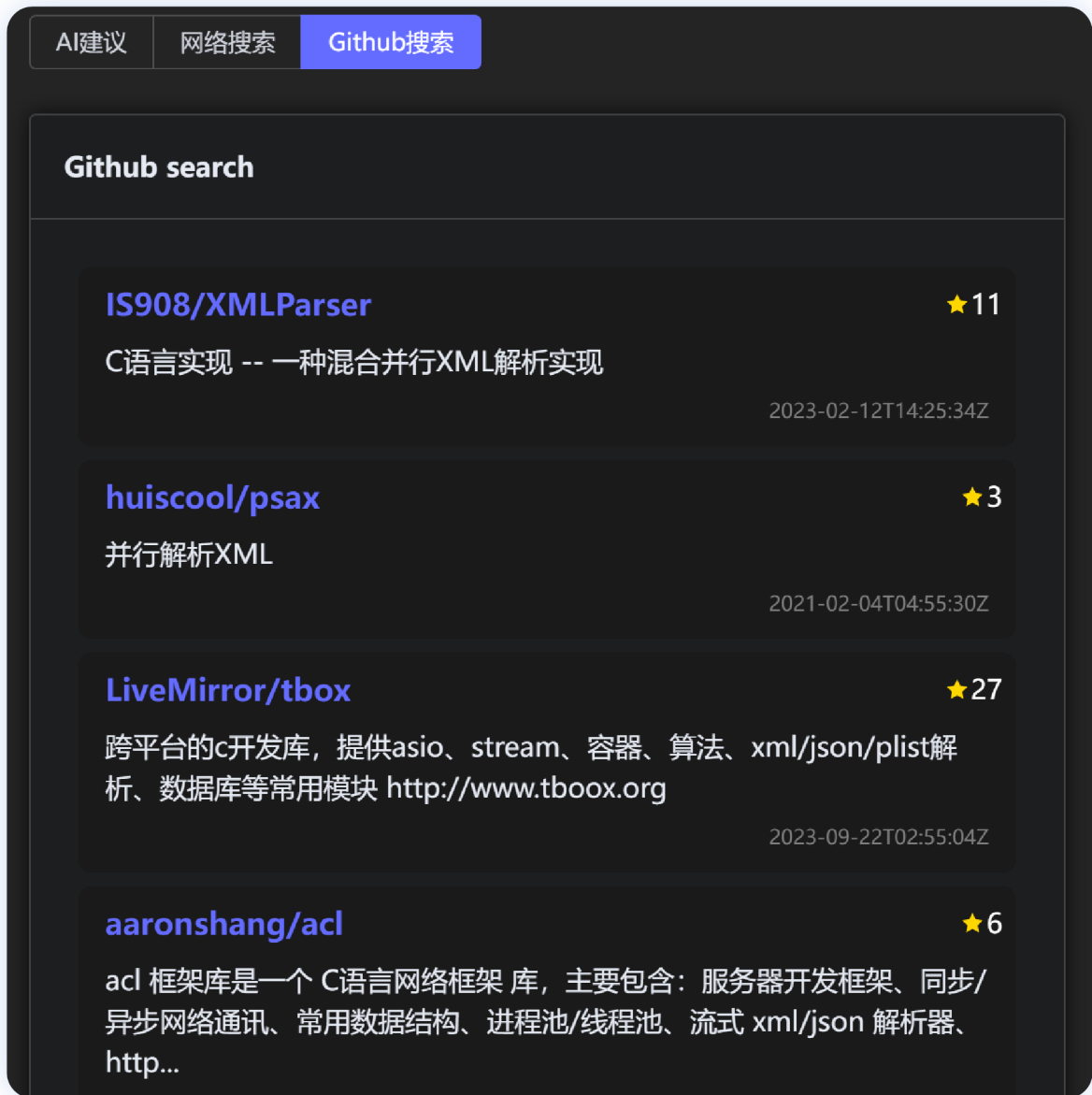
Linux 下C/C++解析XML文件（一） 在工程项目中我们的项目需要根据不同的环境配置不同的程序参数，而常用的两种文件分别是ini文件和XML文件，接下来我来分析下在Linux下解析XML文件过程。我们首先使用linux自带的libxml2来解析XML文件。1. libxml2数据结构 在libxml2中比较重要的数据结构是xmlNodePtr，它在libxml/tree.h中定义为

XML的创建、解析-C语言 - 逆袭之路666 - 博客园

2. 解析XML文档（1）XML解析流程 解析一个XML文档，从中取出想要的信息，例如节点中包含的文字，或者某个节点的属性。其流程如下：① 用xmlReadFile函数读入一个文件，并返回一个文档指针doc。② 用xmlDocGetRootElement函数得到根节点curNode。③ 此时curNode->xmlChildrenNode就是根节点的首个儿子节点，该儿子节点的兄弟节点可用next指针进行轮询。④ 轮询所有子节点，找到所需的节点，用xmlNodeGetContent取出其内容。⑤ 用xmlHasProp查找含有某个属性的节点，属性

- [github仓库搜索](#)

软件还会根据用户的问题，在github的仓库中搜索相关内容，点击链接后会跳转到对应的仓库界面。



- **Stackoverflow文章推荐**

进入软件界面，软件会随机推荐Stackoverflow上的热点问题，并列出其问题涉及的相关方面，点击问题界面后，会进入到问题详情。

推荐文章

Stackoverflow

clarkk

go database/sql can not connect to localhost

mysql

go

浏览量: 71 回答数: 3

Manish Giri

Cannot import a package defined in own Go module

go

module

go-modules

gopath

浏览量: 28 回答数: 2

clarkk

How to handle context and gracefully stop HTTP request from further processing

go

浏览量: 25 回答数: 1

Darwin von Corax

Importing a Go module from a private git repo using ssh

浏览量: 22 回答数: 1

功能实现

界面实现

界面使用 `Vue` 实现，主要使用了 `element-plus` 组件库与 `tailwind css`，`element-plus` 组件库提供了美观的组件，`tailwind css` 提供了使用原子类来代替传统 `css` 的开发，通过这些技术，可以方便快捷地开发出美观的界面。以下是部分界面代码：

```
1  <script setup lang="ts">
2  import {Ref} from 'vue'
3  defineProps({
4    questions: {
5      type: Array<String>,
6      required: true
7    },
8  })
9
10 const emit = defineEmits(['select'])
11 const handleSelect = (ques: string) => {
12   emit('select', ques)
13 }
```



```

14     </script>
15
16     <template>
17         <div class="flex flex-col w-full justify-center items-center mb-2">
18             <div class="w-full flex justify-start items-center flex-wrap">
19                 <el-tag v-for="(ques,index) in questions" :key="index"
class="mx-2.5 my-1" @click="handleSelect(ques.toString())">{{ ques }}</el-
tag>
20             </div>
21         </div>
22     </template>
23
24     <style scoped>
25     .el-tag {
26         cursor: pointer;
27     }
28     </style>

```

API调用

Codehub 总共使用了来自[gpt](#), [bing](#), [github](#), [mediawiki](#), [StackExchange](#)五个来源的api, 各个api的调用实现如下:

gpt

正常传输

- `json schema`

```

1  {
2      "id": "chatcmpl-8CPEoTKCVzwF0tZ2d008izSNQZfXP",
3      "object": "chat.completion",
4      "created": 1697967642,
5      "model": "gpt-3.5-turbo-0613",
6      "choices": [
7          {
8              "index": 0,
9              "message": {
10                 "role": "assistant",
11                 "content": "{如何定义一个函数? }\n\n{什么是递归函数? 如何实现递归函数? }\n\n{什么是面向对象编程? 如何定义一个类? }\n\n{如何判断两个变量是否相等? }\n\n{如何调试代码? 有哪些常用的调试方法? }\n\n{什么是异常处理? 如何在代码中进行异常处理? }"
12             },
13             "finish_reason": "stop"
14         }

```

```

15     ],
16     "usage": {
17         "prompt_tokens": 57,
18         "completion_tokens": 99,
19         "total_tokens": 156
20     },
21     "code": 0,
22     "msg": "ok"
23 }

```

- 代码示例

```

1  import axios from 'axios'
2
3  export const getGptResponse = (params:any, key:string) => {
4      return axios({
5          method:'post',
6
7          url:'http://flag.smarttrot.com/index.php/api/v1/chat/completions',
8          data:params,
9          headers:{
10              'Content-Type':'application/json',
11              'Authorization':'Bearer '+key
12          }
13      }).then(res=>{
14          if(res.status===200&&res.data){
15              if(res.data.msg==='ok'){
16
17                  if(res.data.choices[0].finish_reason==='stop'&&res.data.choices[0].message){
18                      return res.data.choices[0].message.content
19                  }
20              }else{
21                  return 'api error '+res.data.msg
22              }
23          }else{
24              return 'network error'
25          }
26      })
27  }

```

流式传输

由于gpt生成完整回答的时间过长，用户使用体验并不好，因此api还提供了流式传输的选项，可以实现gpt一边生成，一边传给客户端响应。使用流式传输需要在调用参数中指定 `stream: true` 选项，接口会将结果作为EventStream的一系列事件（event）返回,例如：

```
1  async fetch(messages: GptMsgs) {
2      return await
3      fetch('http://flag.smarttrot.com/index.php/api/v1/chat/completions', {
4          method: 'POST',
5          body: JSON.stringify({
6              model: 'gpt-3.5-turbo',
7              messages,
8              stream: true
9          }),
10         headers: {
11             'Content-Type': 'application/json',
12             Authorization: `Bearer ${this.key}`
13         }
14     })
15 }
```

同时发起请求后，需要从response中通过 `getReader` 方法获取 `reader`，同时通过通过while循环 + `reader.read()` 获取每次传输的数据，每次获取的数据格式为 `Uint8Array`，通过浏览器原生支持的 `TextDecoder` 将buffer解析成字符串，再通过正则表达式匹配其中的 `json` 数据

```
1  const parsePack = (str: string) => {
2      // 定义正则表达式匹配模式
3      const pattern = /data:\s*({.*?})\s*\n/g
4      // 定义一个数组来存储所有匹 配到的 JSON 对象
5      const result = []
6      // 使用正则表达式匹配完整的 JSON 对象并解析它们
7      let match
8      while ((match = pattern.exec(str)) !== null) {
9          const jsonStr = match[1]
10         try {
11             const json = JSON.parse(jsonStr)
12             result.push(json)
13         } catch (e) {
14             console.log(e)
15         }
16     }
17     // 输出所有解析出的 JSON 对象
18     return result
19 }
```

```
19 }
```

最后获得gpt的回答。

```
1  async stream(prompt: string, history: GptMsgs = []) {
2      let finish = false
3      let count = 0
4      // 触发onStart
5      this.onStart(prompt)
6      // 发起请求
7      const res = await this.fetch([...history, { 'role': 'user', content:
prompt }])
8      if (!res.body) return
9      // 从response中获取reader
10     const reader = res.body.getReader()
11     const decoder: TextDecoder = new TextDecoder()
12     // 循环读取内容
13     while (!finish) {
14         const { done, value } = await reader.read()
15         // console.log(value)
16         if (done) {
17             finish = true
18             this.onDone()
19             break
20         }
21         count++
22         const jsonArray = parsePack(decoder.decode(value))
23         if (count === 1) {
24             this.onCreated()
25         }
26         jsonArray.forEach((json: any) => {
27             if (!json.choices || json.choices.length === 0) {
28                 return
29             }
30             const text = json.choices[0].delta.content
31             this.onPatch(text)
32         })
33     }
34 }
```

bing

web search

通过直接使用bing的 `web search` api, 得到有关网页的内容, 并排列表示出来

- `json schema`

```
1  {
2      "_type": "SearchResponse",
3      "queryContext": {
4          "originalQuery": "如何定义一个函数? C"
5      },
6      "webPages": {
7          "webSearchUrl": "https://www.bing.com/search?
q=%E5%A6%82%E4%BD%95%E5%AE%9A%E4%B9%89%E4%B8%80%E4%B8%AA%E5%87%BD%E6%95
%B0%EF%BC%9F+C",
8          "totalEstimatedMatches": 56500000,
9          "value": [
10             {
11                 "id":
12                 "https://api.bing.microsoft.com/api/v7/#WebPages.0",
13                 "name": "C语言函数定义 (C语言自定义函数) - C语言中文网",
14                 "url": "http://c.biancheng.net/view/1851.html",
15                 "isFamilyFriendly": true,
16                 "displayUrl": "c.biancheng.net/view/1851.html",
17                 "snippet": "C语言函数定义 (C语言自定义函数) . 函数是一段可以重
复使用的代码, 用来独立地完成某个功能, 它可以接收用户传递的数据, 也可以不接收. . 接收
用户数据的函数在定义时要指明参数, 不接收用户数据的不需要指明, 根据这一点可以将函数分
为有参函数和无 ... ",
18                 "deepLinks": [
19                     {
20                         "name": "C语言枚举类型",
21                         "url":
22                         "http://c.biancheng.net/view/2034.html",
23                         "snippet": "C语言枚举类型 (C语言enum用法) 详解 在实
际编程中, 有些数据的取值往往是有限的, 只能是非常少量的整数, 并且最好为每个值都取一个
名字, 以方便在后续代码中使用, 比如一个星期只有七天, 一年只有十二个月, 一个班每周有六
门课程等.",
24                     },
25                     {
26                         "name": "什么是函数",
27                         "url":
28                         "http://c.biancheng.net/view/1850.html",
```

```
27         "snippet": "C语言自带的函数称为 库函数 (Library  
Function) 。库 (Library) 是编程中的一个基本概念，可以简单地认为它是一系列函数的集  
合，在磁盘上往往是一个文件夹。C语言自带的库称为 标准库 (Standard Library) ，其他  
公司或个人开发的库称为 第三方库 (Third。",  
28         "deepLinks": []  
29     },  
30     {  
31         "name": "函数声明以及函数原型",  
32         "url":  
"http:\\\\c.biancheng.net\\view\\1857.html",  
33         "snippet": "学完《C语言多文件编程》，你对C语言的认识  
将会有质的提升，瞬间豁然开朗，轻松超越 90% 的C语言程序员。 函数参考手册 最后再补充  
一点，函数原型给出了使用该函数的所有细节，当我们不知道如何使用某个函数时，需要查找的  
是它的原型，而不是它的定义，我们往往不关心它的实现。",  
34         "deepLinks": []  
35     },  
36     {  
37         "name": "带实例演示",  
38         "url":  
"http:\\\\c.biancheng.net\\view\\1861.html",  
39         "snippet": "C语言递归函数（递归调用）详解[带实例演示]  
一个函数在它的函数体内调用它自身称为 递归调用 ，这种函数称为 递归函数 。 执行递归函  
数将反复调用其自身，每调用一次就进入新的一层，当最内层的函数执行完毕后，再一层一层地  
由里到外退出。",  
40         "deepLinks": []  
41     },  
42     {  
43         "name": "指针",  
44         "url": "http:\\\\c.biancheng.net\\c\\80\\",  
45         "snippet": "C语言指针是什么？1分钟彻底理解C语言指针的  
概念 2. C语言指针变量的定义和使用（精华） 3. C语言指针变量的运算（加法、减法和比较  
运算） 4. C语言数组指针（指向数组的指针）详解 5. C语言字符串指针（指向字符串的指  
针）详解 6. C语言数组灵活多变的"  
46     },  
47     {  
48         "name": "循环结构和选择结构",  
49         "url": "http:\\\\c.biancheng.net\\c\\32\\",  
50         "snippet": "C语言顺序结构就是让程序按照从头到尾的顺序  
依次执行每一条C语言代码，不重复执行任何代码，也不跳过任何代码。 C语言选择结构也称分  
支结构，就是让程序“拐弯”，有选择性的执行代码；换句话说，可以跳过没用的代码，只执行有  
用的代码。",  
51         "deepLinks": []  
52     }  
53 ],  
54     "dateLastCrawled": "2023-10-16T04:59:00.0000000Z",  
55     "language": "zh_chs",
```

```

56         "isNavigational": true
57     },
58     {
59         "id":
        "https://api.bing.microsoft.com/api/v7/#WebPages.2",
60         "name": "C语言基础：函数的声明与定义 - 知乎 - 知乎专栏",
61         "url": "https://zhuanlan.zhihu.com/p/33885407",
62         "isFamilyFriendly": true,
63         "displayUrl":
        "https://zhuanlan.zhihu.com/p/33885407",
64         "snippet": "函数的声明就是告诉编译器我们想要定义一个函数，并明
        确规定其返回值（输出）、函数名、参数表（输入）。声明函数的语法如下：. type
        function_name (type var); 下面我们来看几个声明函数的例子：. int max (int a,
        int b); \/\返回两个变量中值较大的值。 . float sum (float ...",
65         "dateLastCrawled": "2023-10-19T19:35:00.0000000Z",
66         "language": "zh_chs",
67         "isNavigational": false
68     },
69     {
70         "id":
        "https://api.bing.microsoft.com/api/v7/#WebPages.3",
71         "name": "C语言函数的定义 - C语言教程 - C语言网",
72         "url": "https://www.dotcpp.com/course/25",
73         "isFamilyFriendly": true,
74         "displayUrl": "https://www.dotcpp.com/course/25",
75         "snippet": "C语言函数的定义。C源程序是由 函数 组成的。最简单的
        程序有一个主函数main ()，但实用程序往往由多个函数组成，由主函数调用其他函数，其他
        函数也可以互相调用。函数 是C源程序的基本模块，程序的许多功能是通过函数模块的调用
        来实现的，学会编写和 ...",
76         "dateLastCrawled": "2023-10-20T06:24:00.0000000Z",
77         "language": "zh_chs",
78         "isNavigational": false
79     },
80     {
81         "id":
        "https://api.bing.microsoft.com/api/v7/#WebPages.4",
82         "name": "C语言函数详解（包括声明、定义、使用等）",
83         "url": "http://c.biancheng.net/c/71/",
84         "isFamilyFriendly": true,
85         "displayUrl": "c.biancheng.net/c/71",
86         "snippet": "C语言函数的概念 2. C语言函数定义（C语言自定义函
        数） 3. C语言形参和实参的区别（非常详细） 4. C语言return的用法详解，C语言函数返回
        值详解 5. C语言函数调用详解（从中发现程序运行的秘密） 6. C语言函数声明以及函数原型
        7. C语言全局变量和局部变量（带 ...",
87         "dateLastCrawled": "2023-10-16T10:21:00.0000000Z",
88         "language": "zh_chs",

```

```

89         "isNavigational": false
90     },
91     {
92         "id":
93         "https://api.bing.microsoft.com/api/v7/#WebPages.5",
94         "name": "C 语言函数详解 - 知乎",
95         "url": "https://zhuanlan.zhihu.com/p/610310388",
96         "isFamilyFriendly": true,
97         "displayUrl":
98         "https://zhuanlan.zhihu.com/p/610310388",
99         "snippet": "C 语言函数详解。函数是 C 语言中的重要组成部分，它
100        可以将程序分解为模块，提高代码的可读性和可维护性。函数由函数头和函数体组成，函数头
101        包括函数名、返回值类型和参数列表，函数体包括函数执行的语句块。本文将详细介绍 C 语
102        言中的函数，包括 ...",
103         "dateLastCrawled": "2023-10-17T15:59:00.0000000Z",
104         "language": "zh_chs",
105         "isNavigational": false
106     }
107 ]
108 },
109 "relatedSearches": {
110     "id":
111     "https://api.bing.microsoft.com/api/v7/#RelatedSearches",
112     "value": [
113         {
114             "text": "c 定义一个函数指针",
115             "displayText": "c 定义一个函数指针",
116             "webSearchUrl": "https://www.bing.com/search?
117             q=c+%E5%AE%9A%E4%B9%89%E4%B8%80%E4%B8%AA%E5%87%BD%E6%95%B0%E6%8C%87%E9%
118             92%88"
119         },
120         {
121             "text": "c语言怎么自定义一个函数",
122             "displayText": "c语言怎么自定义一个函数",
123             "webSearchUrl": "https://www.bing.com/search?
124             q=c%E8%AF%AD%E8%A8%80%E6%80%8E%E4%B9%88%E8%87%AA%E5%AE%9A%E4%B9%89%E4%B
125             8%80%E4%B8%AA%E5%87%BD%E6%95%B0"
126         },
127         {
128             "text": "c语言自定义输入函数",
129             "displayText": "c语言自定义输入函数",
130             "webSearchUrl": "https://www.bing.com/search?
131             q=c%E8%AF%AD%E8%A8%80%E8%87%AA%E5%AE%9A%E4%B9%89%E8%BE%93%E5%85%A5%E5%8
132             7%BD%E6%95%B0"
133         },
134     ]
135 }

```



```

123     },
124     "rankingResponse": {
125         "mainline": {
126             "items": [
127                 {
128                     "answerType": "WebPages",
129                     "resultIndex": 0,
130                     "value": {
131                         "id":
132                         "https://api.bing.microsoft.com/api/v7/#WebPages.0"
133                     },
134                 {
135                     "answerType": "WebPages",
136                     "resultIndex": 1,
137                     "value": {
138                         "id":
139                         "https://api.bing.microsoft.com/api/v7/#WebPages.1"
140                     },
141                 {
142                     "answerType": "WebPages",
143                     "resultIndex": 2,
144                     "value": {
145                         "id":
146                         "https://api.bing.microsoft.com/api/v7/#WebPages.2"
147                     },
148                 {
149                     "answerType": "WebPages",
150                     "resultIndex": 3,
151                     "value": {
152                         "id":
153                         "https://api.bing.microsoft.com/api/v7/#WebPages.3"
154                     },
155                 {
156                     "answerType": "WebPages",
157                     "resultIndex": 4,
158                     "value": {
159                         "id":
160                         "https://api.bing.microsoft.com/api/v7/#WebPages.4"
161                     },
162                 {
163                     "answerType": "WebPages",

```

```

164         "resultIndex": 5,
165         "value": {
166             "id":
167             "https://api.bing.microsoft.com/api/v7/#WebPages.5"
168         },
169         {
170             "answerType": "WebPages",
171             "resultIndex": 6,
172             "value": {
173                 "id":
174                 "https://api.bing.microsoft.com/api/v7/#WebPages.6"
175             },
176             {
177                 "answerType": "WebPages",
178                 "resultIndex": 7,
179                 "value": {
180                     "id":
181                     "https://api.bing.microsoft.com/api/v7/#WebPages.7"
182                 },
183                 {
184                     "answerType": "WebPages",
185                     "resultIndex": 8,
186                     "value": {
187                         "id":
188                         "https://api.bing.microsoft.com/api/v7/#WebPages.8"
189                     },
190                     {
191                         "answerType": "RelatedSearches",
192                         "value": {
193                             "id":
194                             "https://api.bing.microsoft.com/api/v7/#RelatedSearches"
195                         }
196                     }
197                 }
198             }
199         }

```

- api调用代码

分页调用，传入pagesize与page

```
1 import axios from "axios"
```

```

2   export const bingWebSearch=
   (query:any,page:number,pagesize:number,SUBSCRIPTION_KEY:string)⇒{
3       return axios({
4           method:'get',
5           url:'https://api.bing.microsoft.com/v7.0/search?
q='+encodeURIComponent(query)+'&page=${page}&pagesize=${pagesize}',
6           headers:{
7               'Ocp-Apim-Subscription-Key':SUBSCRIPTION_KEY
8           }
9       }).then(res⇒{
10          if(res.status===200&&res.data){
11              let pages = res.data.webPages.value.map((page:any)⇒{
12                  return {
13                      url:page.url,
14                      snippet:page.snippet,
15                      name:page.name
16                  }
17              })
18              return pages
19          }else if(res.status===403){
20              return []
21          }else{
22              return []
23          }
24      })
25  }
26
27  export const bingWebSearchSize=(query:any,SUBSCRIPTION_KEY:string)⇒{
28      return axios({
29          method:'get',
30          url:'https://api.bing.microsoft.com/v7.0/search?
q='+encodeURIComponent(query),
31          headers:{
32              'Ocp-Apim-Subscription-Key':SUBSCRIPTION_KEY
33          }
34      }).then(res⇒{
35          if(res.status===200&&res.data){
36              return res.data.webPages.totalEstimatedMatches
37          }else{
38              return -1
39          }
40      })
41  }

```

搜索建议

搜索建议用于在用户输入问题时，实时根据问题的内容给出候选内容

- json schema

```
1  {
2      "_type": "Suggestions",
3      "queryContext": {
4          "originalQuery": "如何定义一个函数? "
5      },
6      "suggestionGroups": [
7          {
8              "name": "Web",
9              "searchSuggestions": [
10                 {
11                     "url": "https://www.bing.com/search?
12 q=%E5%A6%82%E4%BD%95%E5%AE%9A%E4%B9%89%E4%B8%80%E4%B8%AA%E5%87%BD%E6%95%
13 B0matlab&FORM=USBAPI",
14                     "displayText": "如何定义一个函数matlab",
15                     "query": "如何定义一个函数matlab",
16                     "searchKind": "WebSearch"
17                 },
18                 {
19                     "url": "https://www.bing.com/search?
20 q=%E5%A6%82%E4%BD%95%E5%AE%9A%E4%B9%89%E4%B8%80%E4%B8%AA%E5%87%BD%E6%95%
21 B0python&FORM=USBAPI",
22                     "displayText": "如何定义一个函数python",
23                     "query": "如何定义一个函数python",
24                     "searchKind": "WebSearch"
25                 }
26             ]
27         }
28     ]
29 }
```

- api调用代码

```
1  export const bingAutoSuggest = async (query: string): Promise<string[]>
2  => {
3      return axios({
4          method: 'get',
5          url: `https://api.bing.microsoft.com/v7.0/suggestions?
6 q=${encodeURIComponent(query)}`,
7          headers: {
```

```

6      'Ocp-Apim-Subscription-Key': SUBSCRIPTION_KEY,
7    },
8  }).then((res) => {
9    if (res.status === 200 && res.data && res.data.suggestionGroups) {
10     const suggestions =
11     res.data.suggestionGroups[0].searchSuggestions.map((suggestion: any) =>
12     suggestion.displayText);
13     return suggestions;
14   } else {
15     return [];
16   }
17 });
18 };

```

github

搜索仓库

直接使用github的开放api，通过传入搜索关键词与搜索语言，可以搜索到有关的github仓库，api调用采用分页方式，避免一次性传输过多数据量。

- json schema

```

1  {
2    "total_count": 116,
3    "incomplete_results": false,
4    "items": [
5      {
6        "id": 28324684,
7        "node_id": "MDEwOJlJlcG9zaXRvcnkyODMyNDY4NA==",
8        "name": "Data-Structures-and-Algorithms-in-C",
9        "full_name": "LeechanX/Data-Structures-and-Algorithms-in-C",
10       "private": false,
11       "owner": {
12         "login": "LeechanX",
13         "id": 3104306,
14         "node_id": "MDQ6VXNlcjMxMDQzMjY=",
15         "avatar_url": "https://avatars.githubusercontent.com/u/3104306?v=4",
16         "gravatar_id": "",
17         "url": "https://api.github.com/users/LeechanX",
18         "html_url": "https://github.com/LeechanX",
19         "followers_url":
20         "https://api.github.com/users/LeechanX/followers",
21         "following_url":
22         "https://api.github.com/users/LeechanX/following{/other_user}",

```

```
21         "gists_url":
22         "https://api.github.com/users/LeechanX/gists{/gist_id}",
23         "starred_url":
24         "https://api.github.com/users/LeechanX/starred{/owner}/{/repo}",
25         "subscriptions_url":
26         "https://api.github.com/users/LeechanX/subscriptions",
27         "organizations_url":
28         "https://api.github.com/users/LeechanX/orgs",
29         "repos_url": "https://api.github.com/users/LeechanX/repos",
30         "events_url":
31         "https://api.github.com/users/LeechanX/events{/privacy}",
32         "received_events_url":
33         "https://api.github.com/users/LeechanX/received_events",
34         "type": "User",
35         "site_admin": false
36     },
37     "html_url": "https://github.com/LeechanX/Data-Structures-and-
38     Algorithms-in-C",
39     "description": "所有基础数据结构和算法的纯C语言实现，如各自排序、链表、
40     栈、队列、各种树以及应用、图算法、字符串匹配算法、回溯、并查集等，献丑了",
41     "fork": false,
42     "url": "https://api.github.com/repos/LeechanX/Data-Structures-
43     and-Algorithms-in-C",
44     "forks_url": "https://api.github.com/repos/LeechanX/Data-
45     Structures-and-Algorithms-in-C/forks",
46     "keys_url": "https://api.github.com/repos/LeechanX/Data-
47     Structures-and-Algorithms-in-C/keys{/key_id}",
48     "collaborators_url": "https://api.github.com/repos/LeechanX/Data-
49     Structures-and-Algorithms-in-C/collaborators{/collaborator}",
50     "teams_url": "https://api.github.com/repos/LeechanX/Data-
51     Structures-and-Algorithms-in-C/teams",
52     "hooks_url": "https://api.github.com/repos/LeechanX/Data-
53     Structures-and-Algorithms-in-C/hooks",
54     "issue_events_url": "https://api.github.com/repos/LeechanX/Data-
55     Structures-and-Algorithms-in-C/issues/events{/number}",
56     "events_url": "https://api.github.com/repos/LeechanX/Data-
57     Structures-and-Algorithms-in-C/events",
58     "assignees_url": "https://api.github.com/repos/LeechanX/Data-
59     Structures-and-Algorithms-in-C/assignees{/user}",
60     "branches_url": "https://api.github.com/repos/LeechanX/Data-
61     Structures-and-Algorithms-in-C/branches{/branch}",
62     "tags_url": "https://api.github.com/repos/LeechanX/Data-
63     Structures-and-Algorithms-in-C/tags",
64     "blobs_url": "https://api.github.com/repos/LeechanX/Data-
65     Structures-and-Algorithms-in-C/git/blobs{/sha}",
```

```
46     "git_tags_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/git/tags{/sha}",
47     "git_refs_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/git/refs{/sha}",
48     "trees_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/git/trees{/sha}",
49     "statuses_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/statuses/{sha}",
50     "languages_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/languages",
51     "stargazers_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/stargazers",
52     "contributors_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/contributors",
53     "subscribers_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/subscribers",
54     "subscription_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/subscription",
55     "commits_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/commits{/sha}",
56     "git_commits_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/git/commits{/sha}",
57     "comments_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/comments{/number}",
58     "issue_comment_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/issues/comments{/number}",
59     "contents_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/contents/{+path}",
60     "compare_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/compare/{base}...{head}",
61     "merges_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/merges",
62     "archive_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/{archive_format}/{ref}",
63     "downloads_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/downloads",
64     "issues_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/issues{/number}",
65     "pulls_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/pulls{/number}",
66     "milestones_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/milestones{/number}",
67     "notifications_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/notifications{?
since,all,participating}",
```

```
68     "labels_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/labels{/name}",
69     "releases_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/releases{/id}",
70     "deployments_url": "https://api.github.com/repos/LeechanX/Data-
Structures-and-Algorithms-in-C/deployments",
71     "created_at": "2014-12-22T04:40:26Z",
72     "updated_at": "2023-10-20T08:50:16Z",
73     "pushed_at": "2019-10-13T14:17:20Z",
74     "git_url": "git://github.com/LeechanX/Data-Structures-and-
Algorithms-in-C.git",
75     "ssh_url": "git@github.com:LeechanX/Data-Structures-and-
Algorithms-in-C.git",
76     "clone_url": "https://github.com/LeechanX/Data-Structures-and-
Algorithms-in-C.git",
77     "svn_url": "https://github.com/LeechanX/Data-Structures-and-
Algorithms-in-C",
78     "homepage": null,
79     "size": 5669,
80     "stargazers_count": 807,
81     "watchers_count": 807,
82     "language": "C",
83     "has_issues": true,
84     "has_projects": true,
85     "has_downloads": true,
86     "has_wiki": true,
87     "has_pages": false,
88     "has_discussions": false,
89     "forks_count": 353,
90     "mirror_url": null,
91     "archived": false,
92     "disabled": false,
93     "open_issues_count": 1,
94     "license": null,
95     "allow_forking": true,
96     "is_template": false,
97     "web_commit_signoff_required": false,
98     "topics": [
99
100    ],
101     "visibility": "public",
102     "forks": 353,
103     "open_issues": 1,
104     "watchers": 807,
105     "default_branch": "master",
106     "score": 1.0
```



```
107     },
108     {
109         "id": 4604718,
110         "node_id": "MDEwOJlJlcG9zaXRvcnk0NjA0NzE4",
111         "name": "sortMethods",
112         "full_name": "hackvilin/sortMethods",
113         "private": false,
114         "owner": {
115             "login": "hackvilin",
116             "id": 1831116,
117             "node_id": "MDQ6VXNlcjE4MzExMTY=",
118             "avatar_url": "https://avatars.githubusercontent.com/u/1831116?v=4",
119             "gravatar_id": "",
120             "url": "https://api.github.com/users/hackvilin",
121             "html_url": "https://github.com/hackvilin",
122             "followers_url":
123             "https://api.github.com/users/hackvilin/followers",
124             "following_url":
125             "https://api.github.com/users/hackvilin/following{/other_user}",
126             "gists_url":
127             "https://api.github.com/users/hackvilin/gists{/gist_id}",
128             "starred_url":
129             "https://api.github.com/users/hackvilin/starred{/owner}/{/repo}",
130             "subscriptions_url":
131             "https://api.github.com/users/hackvilin/subscriptions",
132             "organizations_url":
133             "https://api.github.com/users/hackvilin/orgs",
134             "repos_url": "https://api.github.com/users/hackvilin/repos",
135             "events_url":
136             "https://api.github.com/users/hackvilin/events{/privacy}",
137             "received_events_url":
138             "https://api.github.com/users/hackvilin/received_events",
139             "type": "User",
140             "site_admin": false
141         },
142         "html_url": "https://github.com/hackvilin/sortMethods",
143         "description": "C语言几种排序算法实现",
144         "fork": false,
145         "url": "https://api.github.com/repos/hackvilin/sortMethods",
146         "forks_url":
147         "https://api.github.com/repos/hackvilin/sortMethods/forks",
148         "keys_url":
149         "https://api.github.com/repos/hackvilin/sortMethods/keys{/key_id}",
```

```
140         "collaborators_url":
            "https://api.github.com/repos/hackvilin/sortMethods/collaborators{/collaborator}",
141         "teams_url":
            "https://api.github.com/repos/hackvilin/sortMethods/teams",
142         "hooks_url":
            "https://api.github.com/repos/hackvilin/sortMethods/hooks",
143         "issue_events_url":
            "https://api.github.com/repos/hackvilin/sortMethods/issues/events{/number}",
144         "events_url":
            "https://api.github.com/repos/hackvilin/sortMethods/events",
145         "assignees_url":
            "https://api.github.com/repos/hackvilin/sortMethods/assignees{/user}",
146         "branches_url":
            "https://api.github.com/repos/hackvilin/sortMethods/branches{/branch}",
147         "tags_url":
            "https://api.github.com/repos/hackvilin/sortMethods/tags",
148         "blobs_url":
            "https://api.github.com/repos/hackvilin/sortMethods/git/blobs{/sha}",
149         "git_tags_url":
            "https://api.github.com/repos/hackvilin/sortMethods/git/tags{/sha}",
150         "git_refs_url":
            "https://api.github.com/repos/hackvilin/sortMethods/git/refs{/sha}",
151         "trees_url":
            "https://api.github.com/repos/hackvilin/sortMethods/git/trees{/sha}",
152         "statuses_url":
            "https://api.github.com/repos/hackvilin/sortMethods/statuses/{sha}",
153         "languages_url":
            "https://api.github.com/repos/hackvilin/sortMethods/languages",
154         "stargazers_url":
            "https://api.github.com/repos/hackvilin/sortMethods/stargazers",
155         "contributors_url":
            "https://api.github.com/repos/hackvilin/sortMethods/contributors",
156         "subscribers_url":
            "https://api.github.com/repos/hackvilin/sortMethods/subscribers",
157         "subscription_url":
            "https://api.github.com/repos/hackvilin/sortMethods/subscription",
158         "commits_url":
            "https://api.github.com/repos/hackvilin/sortMethods/commits{/sha}",
159         "git_commits_url":
            "https://api.github.com/repos/hackvilin/sortMethods/git/commits{/sha}",
160         "comments_url":
            "https://api.github.com/repos/hackvilin/sortMethods/comments{/number}",
```

```
161         "issue_comment_url":
            "https://api.github.com/repos/hackvilin/sortMethods/issues/comments{/nu
            mber}",
162         "contents_url":
            "https://api.github.com/repos/hackvilin/sortMethods/contents/{+path}",
163         "compare_url":
            "https://api.github.com/repos/hackvilin/sortMethods/compare/{base}...
            {head}",
164         "merges_url":
            "https://api.github.com/repos/hackvilin/sortMethods/merges",
165         "archive_url":
            "https://api.github.com/repos/hackvilin/sortMethods/{archive_format}
            {/ref}",
166         "downloads_url":
            "https://api.github.com/repos/hackvilin/sortMethods/downloads",
167         "issues_url":
            "https://api.github.com/repos/hackvilin/sortMethods/issues{/number}",
168         "pulls_url":
            "https://api.github.com/repos/hackvilin/sortMethods/pulls{/number}",
169         "milestones_url":
            "https://api.github.com/repos/hackvilin/sortMethods/milestones{/number}
            ",
170         "notifications_url":
            "https://api.github.com/repos/hackvilin/sortMethods/notifications{?
            since,all,participating}",
171         "labels_url":
            "https://api.github.com/repos/hackvilin/sortMethods/labels{/name}",
172         "releases_url":
            "https://api.github.com/repos/hackvilin/sortMethods/releases{/id}",
173         "deployments_url":
            "https://api.github.com/repos/hackvilin/sortMethods/deployments",
174         "created_at": "2012-06-09T03:46:58Z",
175         "updated_at": "2022-03-29T07:14:54Z",
176         "pushed_at": "2012-06-09T04:51:01Z",
177         "git_url": "git://github.com/hackvilin/sortMethods.git",
178         "ssh_url": "git@github.com:hackvilin/sortMethods.git",
179         "clone_url": "https://github.com/hackvilin/sortMethods.git",
180         "svn_url": "https://github.com/hackvilin/sortMethods",
181         "homepage": null,
182         "size": 104,
183         "stargazers_count": 10,
184         "watchers_count": 10,
185         "language": "C",
186         "has_issues": true,
187         "has_projects": true,
188         "has_downloads": true,
```

```

189         "has_wiki": true,
190         "has_pages": false,
191         "has_discussions": false,
192         "forks_count": 7,
193         "mirror_url": null,
194         "archived": false,
195         "disabled": false,
196         "open_issues_count": 0,
197         "license": null,
198         "allow_forking": true,
199         "is_template": false,
200         "web_commit_signoff_required": false,
201         "topics": [
202
203         ],
204         "visibility": "public",
205         "forks": 7,
206         "open_issues": 0,
207         "watchers": 10,
208         "default_branch": "master",
209         "score": 1.0
210     },
211 ]
212 }

```

- api调用代码

分页获取内容

```

1  import axios from "axios"
2
3  //分页获取
4  export const githubSearchRepo =
    (query:string,lang:string,page:number,perPage:number) => {
5      const url = `https://api.github.com/search/repositories?
        q=`+encodeURIComponent(query)+`+language:`+encodeURIComponent(lang)
6      return axios({
7          method: 'get',
8          url:url,
9          params:{
10              page:`${page}`,
11              per_page:`${perPage}`
12          }
13      }).then((res:any)=>{
14          if(res.status===200&&res.data){
15              let repos = res.data.items.map((repo:any)=>{
16                  return {

```

```

17         full_name:repo.full_name,
18         description:repo.description,
19         url:repo.html_url,
20         updated_at:repo.updated_at,
21         stars:repo.stargazers_count
22     }
23 })
24 if(repos.length===0){
25     repos = [{
26         //搜索结果为空, star为-1
27         full_name:'',
28         description:'',
29         url:'',
30         updated_at:'',
31         stars:-1
32     }]
33 }
34 return repos
35 }else{
36     return [{
37         //网络错误, star为-2
38         full_name:'网络错误',
39         description:'请检查网络情况',
40         url:'',
41         updated_at:'',
42         stars:-2
43     }]
44 }
45 })
46 }
47
48 export const githubSearchRepoSize = (query:string,lang:string) => {
49     const url = `https://api.github.com/search/repositories?
50     q=`+encodeURIComponent(query)+`+language:`+encodeURIComponent(lang)
51     return axios({
52         method:'get',
53         url:url,
54     }).then((res:any)=>{
55         if(res.status===200&&res.data){
56             return res.data.total_count
57         }else{
58             return -1
59         }
60     })
61 }

```

mediawiki

维基百科搜索

将gpt联想的关键词, 通过 `mediawiki` 的api在维基百科上进行搜索, 返回给对应关键词的维基百科链接与标题, 方便用户直接查看维基百科的内容.

- `json schema`

```
1  {
2      "pages": [
3          {
4              "id": 345741,
5              "key": "实时操作系统",
6              "title": "实时操作系统",
7              "excerpt": "<span class=\"searchmatch\">时</span>。这种特性保证了各个任务的及时执行。 设计<span class=\"searchmatch\">实时</span>操作<span class=\"searchmatch\">系统</span>的首要目标不是高的吞吐量, 而是保证任务在特定时间内完成, 因此衡量一个<span class=\"searchmatch\">实时</span>操作<span class=\"searchmatch\">系统</span>坚固性的重要指标, 是<span class=\"searchmatch\">系统</span>从接收一个任务, 到完成该任务所需的时间, 其时间的变化称为抖动。可以依抖动將<span class=\"searchmatch\">实时</span>操作<span class=\"searchmatch\">系统</span>分為兩種: 硬实<span class=\"searchmatch\">时</span>操作<span class=\"searchmatch\">系统</span>及软<span class=\"searchmatch\">实时</span>操作<span class=\"searchmatch\">系统</span>, 硬实<span class=\"searchmatch\">时</span>操作系统比软实时操作系统有更少的抖动: ",
8              "matched_title": null,
9              "description": null,
10             "thumbnail": null
11         }
12     ]
13 }
```

- `api调用`

```
1  import axios from 'axios';
2
3  export interface SearchResult {
4      title: string;
5      key: string;
6      url: string;
7  }
8
9  export const wikipediaSearch=(keyword: string):Promise<SearchResult>=> {
```

```

10     const apiUrl = 'https://zh.wikipedia.org/w/rest.php/v1/search/page?
    q='+encodeURIComponent(keyword)+'&limit=1';
11     let searchResult: SearchResult
12     return axios({
13         method: 'get',
14         url: apiUrl,
15     }).then((response) => {
16         if(response.status !== 200) {
17             searchResult={
18                 title: 'null',
19                 key: 'null',
20                 url: 'null',
21             }
22             throw new Error('wikipedia search failed');
23         }else if(response.data.pages.length === 0) {
24             searchResult={
25                 title: 'null',
26                 key: 'null',
27                 url: 'null',
28             }
29         }else {
30             searchResult={
31                 title: response.data.pages[0].title,
32                 key: response.data.pages[0].key,
33                 url:
34                 'https://zh.wikipedia.org/wiki/'+response.data.pages[0].key,
35             }
36             return searchResult;
37         });
38     }

```

StackExchange

文章推荐

在首页展示 `stackoverflow` 的推荐问题，展示问题内容，浏览记录，回答数与分类标签。

- `json schema`

```

1  {
2      "items": [
3          {
4              "tags": [
5                  "reactjs",
6                  "firebase",

```

```

7         "firebase-authentication"
8     ],
9     "owner": {
10         "account_id": 23015920,
11         "reputation": 778,
12         "user_id": 17137694,
13         "user_type": "registered",
14         "profile_image":
15         "https://www.gravatar.com/avatar/7d086148ca0d3e34dde09c77d3301e4b?
16         s=256&d=identicon&r=PG",
17         "display_name": "aabdulelahad",
18         "link":
19         "https://stackoverflow.com/users/17137694/aabdulelahad"
20     },
21     "is_answered": false,
22     "view_count": 6,
23     "answer_count": 1,
24     "score": 0,
25     "last_activity_date": 1697976815,
26     "creation_date": 1697975977,
27     "question_id": 77339859,
28     "content_license": "CC BY-SA 4.0",
29     "link":
30     "https://stackoverflow.com/questions/77339859/custom-claims-and-
31     checking-for-pro-membership",
32     "title": "Custom Claims and Checking for Pro Membership"
33 },
34 {
35     "tags": [
36         "reactjs",
37         "tailwind-css"
38     ],
39     "owner": {
40         "account_id": 27427891,
41         "reputation": 51,
42         "user_id": 22247455,
43         "user_type": "registered",
44         "profile_image":
45         "https://www.gravatar.com/avatar/122769b1fcce74fc19a6454ba7c04716?
46         s=256&d=identicon&r=PG&f=y&so-version=2",
47         "display_name": "testing123",
48         "link":
49         "https://stackoverflow.com/users/22247455/testing123"
50     },
51     "is_answered": true,
52     "view_count": 1986,

```



```

45         "answer_count": 4,
46         "score": 4,
47         "last_activity_date": 1697976699,
48         "creation_date": 1689695468,
49         "last_edit_date": 1689696737,
50         "question_id": 76714567,
51         "content_license": "CC BY-SA 4.0",
52         "link": "https://stackoverflow.com/questions/76714567/cant-
use-shadcn-components",
53         "title": "Can't use Shadcn components"
54     }
55 ],
56     "has_more": true,
57     "quota_max": 300,
58     "quota_remaining": 266
59 }

```

- api调用代码

```

1  import axios from 'axios';
2
3  export interface StackOverflowQuestion {
4      title: string;
5      link: string;
6      view_count?: number;
7      answer_count?: number;
8      tags?: string[];
9      owner_name?: string;
10 }
11
12 export const getStackOverflowQuestions = async (tag:string):
    Promise<StackOverflowQuestion[]> => {
13     const apiUrl =
14         `https://api.stackexchange.com/2.3/questions?
&order=desc&sort=activity&tagged=${tag}&site=stackoverflow`;
15     const response = await axios({
16         method: 'get',
17         url: apiUrl,
18     });
19     if (response.status !== 200) {
20         throw new Error('get Stack Overflow questions failed');
21     } else {
22         const questions = response.data.items.map((item: any) => ({
23             title: item.title,
24             link: item.link,
25             view_count: item.view_count,

```

```

26         answer_count: item.answer_count,
27         tags: item.tags,
28         owner_name: item.owner.display_name,
29     }));
30     return questions;
31 }
32 };

```

api使用示例

获取联想问题

通过给gpt设定prompt，让其以规定格式返回我所需的内容，再通过正则表达式匹配，得到我需要的联想问题，同时传给对应联想问题展示组件，将问题展示出来。

以下是得到联想问题的函数

```

1  const getSuggestQuestion=(question:string='')=>{
2      suggestQuestionsLoading.value=true
3      let prompt=`假设你是搜索引擎助手,我给出问题,请你给出6个我接下来可能会问的相关联想
   问题,每一个联想问题单独用{}括起来,例如{如何解析json文件},请不要包含任何多余的内容,我要
   问的问题的内容是${question}`
4      if(question.length===0){
5          prompt=`请给我6个有关于编程技术方面的问题,问题用{}单独括起来,例如{如何解析
   json文件},请不要包含任何多余的内容`
6      }
7      getGptResponse({
8          'messages':[
9              {'role':'user','content':prompt},
10         ]
11     },gptApiKey.value).then((res:string)>=>{
12         //通过正则表达式匹配
13         const getQuestions=async (res:string):Promise<string[]>=>{
14             const regex = /{([^}]+)}/g;
15             const questions=res.match(regex)
16             return questions?.map((q: string) => q.slice(1, -1)) || [];
17         }
18         getQuestions(res).then((res)>=>{
19             suggestQuestions.value=res
20             suggestQuestionsLoading.value=false
21         })
22     }).catch((err)>=>{
23         console.log(err)
24         suggestQuestions.value=[err.message||'error']
25         suggestQuestionsLoading.value=false

```

```

26     })
27   }

```

传值给对应子组件，将问题展示出来，

```

1   <div class="w-3/4">
2     <div class="w-full mb-4 mt-5">
3       <h2 class="text-2xl font-bold">猜你想搜</h2>
4     </div>
5     <el-skeleton :rows="1" animated :loading="suggestQuestionsLoading">
6       <template #default>
7         <SuggestQuestions :questions="suggestQuestions"
@select="handleSuggestionsSelect"></SuggestQuestions>
8       </template>
9     </el-skeleton>
10  </div>

```

同理，可以展示出问题的联想关键词。

获取web搜索结果

在主组件中调用web搜索api，获取到搜索结果后，传值给子组件，通过子组件渲染界面。

以下是主组件中调用api的函数

```

1   bingWebSearch(question.value+`
    ${preferredLanguage.value}`,webCurrentPage.value,webPageSize.value,bingApiK
ey.value).then((res)⇒{
2     if(res.length>0){
3       webSearchResults.value=res
4       webLoading.value=false
5     }else{
6       console.log('no result')
7       webLoading.value=false
8     }
9   })

```

界面渲染

```

1   <!-- web search -->
2   <el-card class="my-2.5 w-full" v-if="resultSource===sources.WEB">
3     <template #header>
4       <span class="flex justify-start font-bold">Web search</span>
5     </template>

```

```

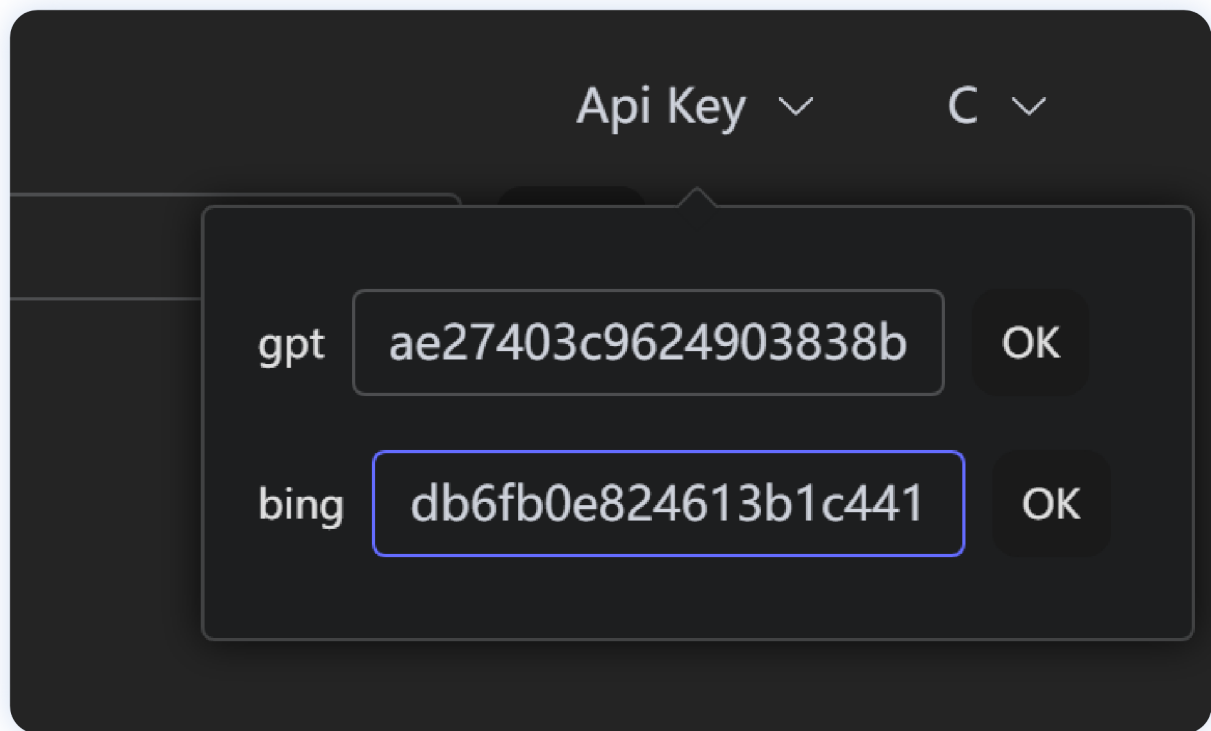
6      <el-skeleton :rows="8" :loading="webLoading" animated>
7          <template #template>
8              <div class="github-loading-skeleton">
9                  <div class="github-loading-skeleton-single" v-for="(index)
10 in 10" :key="index">
11                      <el-skeleton-item variant="p" class="mt-1.5 mb-2"
12 style="width: 80%;"></el-skeleton-item>
13                      <el-skeleton-item variant="text" class="my-1.5"
14 style="width: 33%;"></el-skeleton-item>
15                      <el-skeleton-item variant="text" class="my-1.5"
16 style="width: 100%;"></el-skeleton-item>
17                      <el-skeleton-item variant="text" class="my-1.5"
18 style="width: 45%;"></el-skeleton-item>
19                  </div>
20              </div>
21          </template>
22          <template #default>
23              <Weblink :links="webSearchResults"></Weblink>
24          </template>
25      </el-skeleton>
26      <div class="flex justify-center items-center">
27          <el-pagination :total="webTotalCount" v-model:current-
28 page="webCurrentPage" v-model:page-size="webPageSize"
29 layout="prev,pager,next" @current-change="handleWebPageChange"></el-
30 pagination>
31      </div>
32  </el-card>

```

同理可以得到github搜索界面与stackoverflow文章界面

api密钥储存

api密钥明文储存在代码中是不安全的行为，因此我使用 `electron store` 模块，将api密钥本地存储。第一次使用软件时，软件会提示要求输入gpt与bing的api密钥，输入完成后才能够使用。



- 储存与读取api密钥

由于 `electron-store` 模块在渲染进程中使用会有问题，因此需要在主进程中提供事件监听函数，通过 `electron` 的 `ipcMain` 与 `ipcRender` 模块在主进程与渲染进程间进行通信，具体代码如下

- 主进程

```
1  import Store from 'electron-store'
2  const store = new Store()
3  // IPC listener
4  ipcMain.on('electron-store-get', (event, val) => {
5    event.returnValue = store.get(val);
6  });
7  ipcMain.on('electron-store-set', (event, key, val) => {
8    store.set(key, val);
9  });
10
11  ipcMain.on('electron-store-clear', (event) => {
12    store.clear();
13  });
```

- 渲染进程

```
1  const submitGptApiKey={()=>{
2    if(gptApiKey.value.length===0){
```

```
3         ElMessage({
4             message: 'key不能为空',
5             type: 'warning'
6         })
7         return
8     }
9     ipcRenderer.send('electron-store-set', 'gpt-api-
key', gptApiKey.value)
10    ipcRenderer.send('electron-store-set', 'gpt-api-key-filled', true)
11    setTimeout(() => {
12        isGptApiFilled.value=ipcRenderer.sendSync('electron-store-
get', 'gpt-api-key-filled')
13        gptApiKey.value=ipcRenderer.sendSync('electron-store-
get', 'gpt-api-key')
14    }, 100);
15 }
16
17 const submitBingApiKey=()=>{
18     if (bingApiKey.value.length===0) {
19         ElMessage({
20             message: 'key不能为空',
21             type: 'warning'
22         })
23         return
24     }
25     ipcRenderer.send('electron-store-set', 'bing-api-
key', bingApiKey.value)
26     ipcRenderer.send('electron-store-set', 'bing-api-key-filled', true)
27     setTimeout(() => {
28         isBingApiFilled.value=ipcRenderer.sendSync('electron-store-
get', 'bing-api-key-filled')
29         bingApiKey.value=ipcRenderer.sendSync('electron-store-
get', 'bing-api-key')
30     }, 100);
31 }
```