



南开大学
Nankai University

南 开 大 学

互联网数据库开发实验报告

实现文档

2111947 王宇卉 信息安全、法学双学位班

2212307 刘欣然 物联网工程

2213254 聂嘉欣 计算机科学与技术

2213029 晏家钰 计算机科学与技术

2024 年 12 月 24 日

目录

一、 前期规划	1
二、 任务分工	1
三、 文件目录展示	2
(一) 前端目录	2
(二) 后端目录	3
(三) 数据库目录	5
四、 主要代码展示	5
(一) 前端部分	5
(二) 后端部分	7
(三) 数据库	9

一、 前期规划

本项目的实验规划主要在小组微信群聊中实现，在规划过程中晏家钰同学将任务分成了前端、后端、数据库设计三部分，分别由王宇卉、晏家钰、刘欣然和聂嘉欣四位同学自行认领并负责完成。

在进一步的讨论中，王宇卉和晏家钰同学确认采用 vue 和 yii2 组成前后端分离的形式来完成网站的搭建；晏家钰和刘欣然、聂嘉欣同学确认了后端与数据库设计的相关信息整合事项。

二、 任务分工

2111947 王宇卉：

- 负责 Vue 前端设计及代码实现
- 负责相关报告的撰写
- 负责报告的细化编排

2213029 晏家钰：

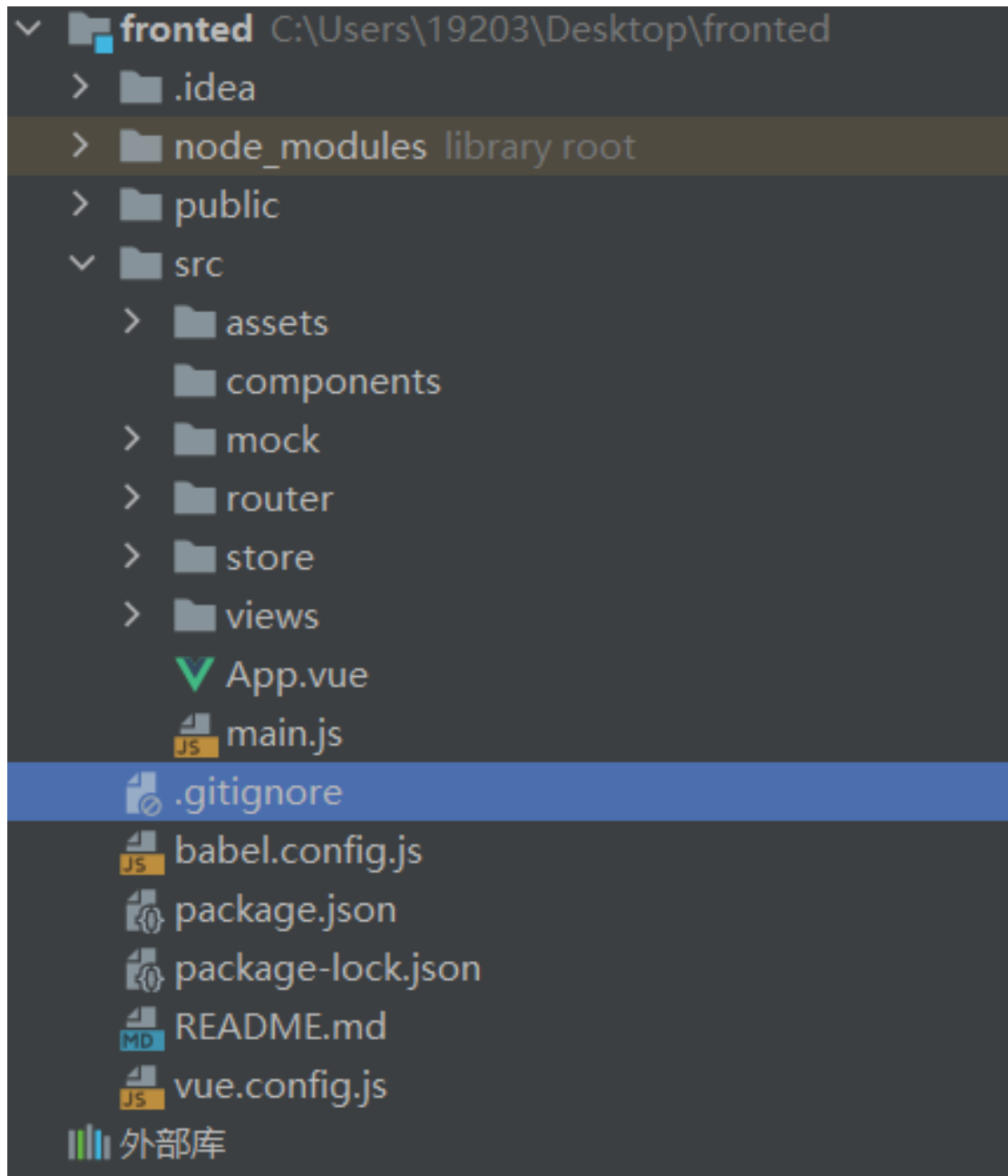
- 负责 yii2 后端设计及代码实现
- 负责相关报告的撰写
- 负责后端与数据库连接
- 前后端连接

2213254 聂嘉欣 2212307 刘欣然：

- 负责数据库设计与构建
- 为表项进行主键设置与外键连接
- 负责相关报告的撰写
- 编写代码抓取数据
- PPT 制作与汇报

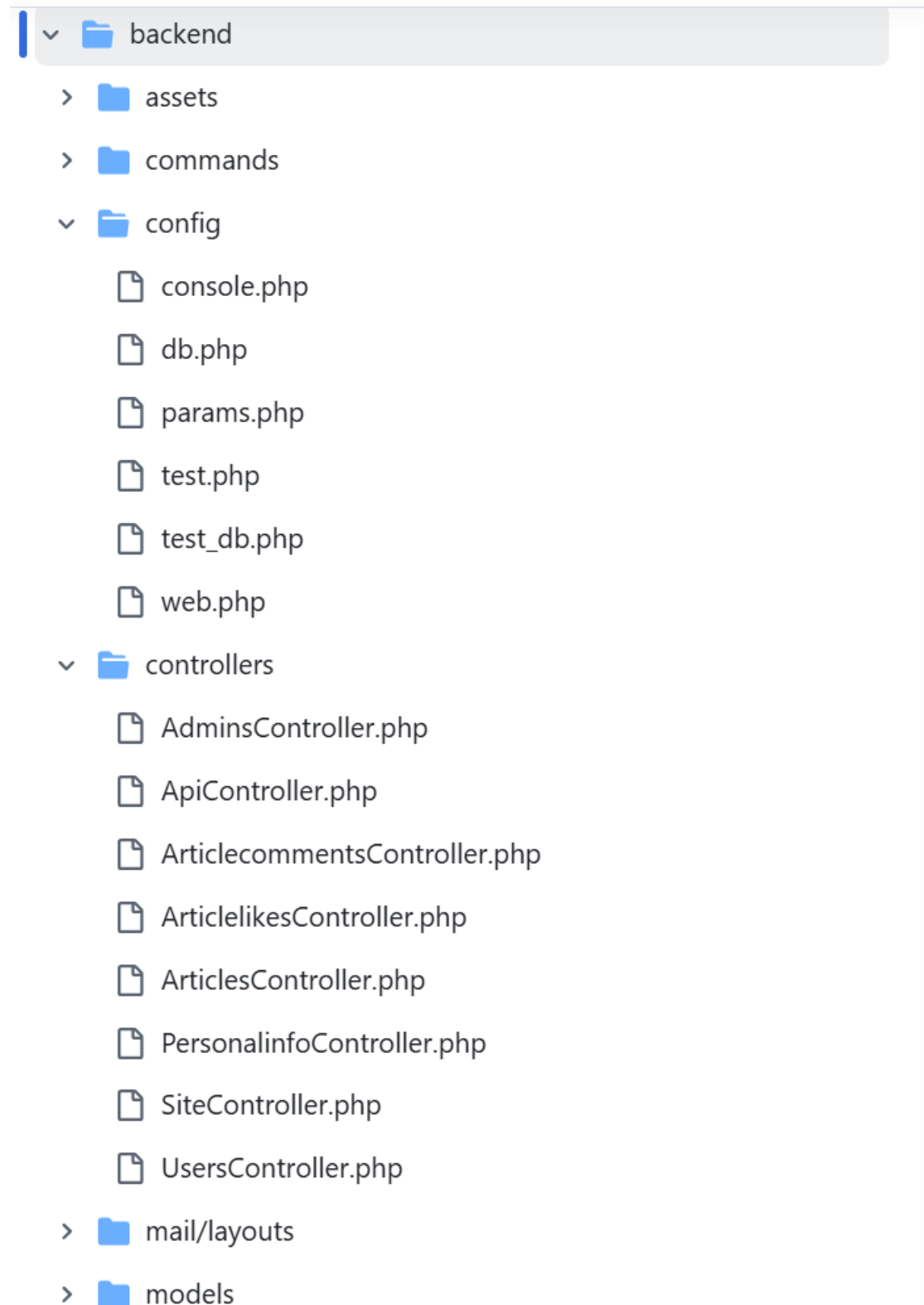
三、 文件目录展示

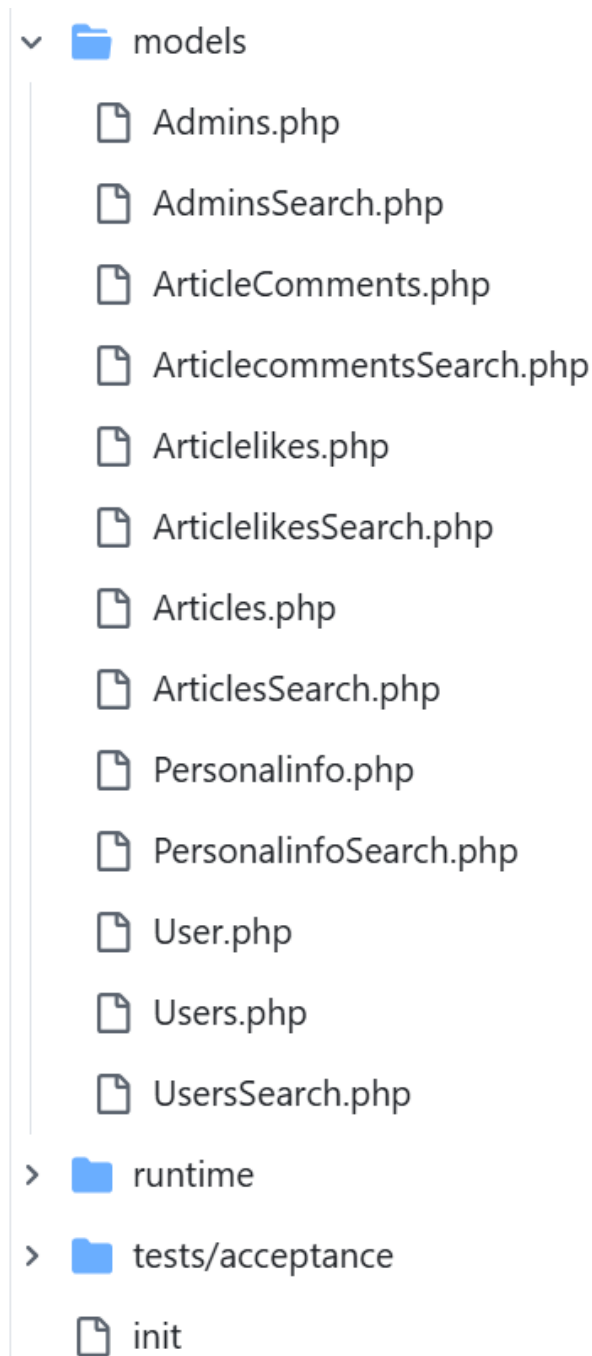
(一) 前端目录



其中 views 是每个页面的代码，components 是网页代码可以嵌入的模块，App.vue 是项目的主入口。

(二) 后端目录



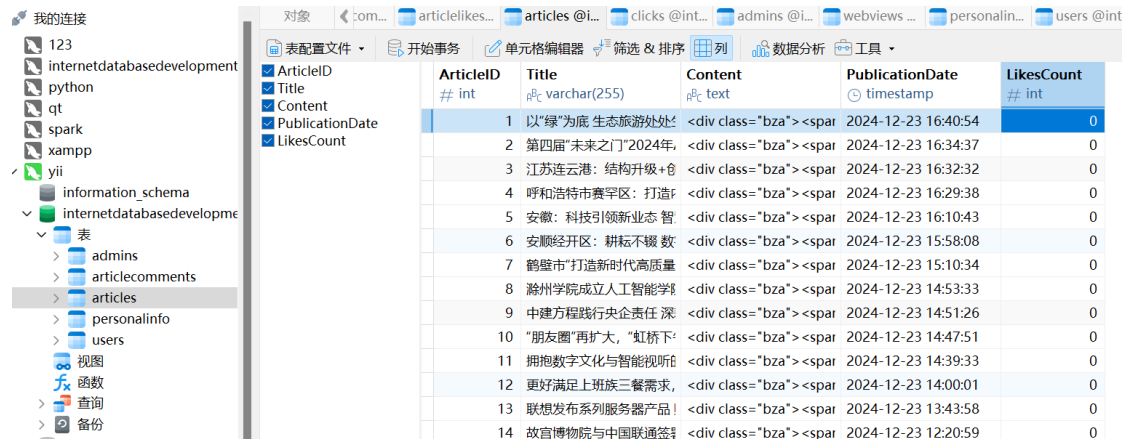


config 文件夹：包含了应用程序的配置文件，用于配置不同环境下的运行参数。

controllers 文件夹：用于存放控制器类文件，控制器负责处理用户请求并响应相应的操作。

models 文件夹：主要用于存放模型类文件，模型类是应用程序与数据库表之间的映射。

(三) 数据库目录



The screenshot shows a database management interface. On the left, a tree view displays the database structure, including tables like 'admins', 'articlecomments', 'articles', 'personalinfo', and 'users'. The 'articles' table is selected. The main area shows the table's structure and data. The structure includes columns: ArticleID (int), Title (varchar(255)), Content (text), PublicationDate (timestamp), and LikesCount (int). The data table lists 14 articles with their respective titles, content snippets, publication dates, and like counts.

ArticleID	Title	Content	PublicationDate	LikesCount
1	以“绿”为底 生态旅游处处	<div class="bza"> <spar	2024-12-23 16:40:54	0
2	第四届“未来之门”2024年	<div class="bza"> <spar	2024-12-23 16:34:37	0
3	江苏连云港：结构升级+创	<div class="bza"> <spar	2024-12-23 16:32:32	0
4	呼和浩特市赛罕区：打造	<div class="bza"> <spar	2024-12-23 16:29:38	0
5	安徽：科技引领新业态 智	<div class="bza"> <spar	2024-12-23 16:10:43	0
6	安顺经开区：耕耘不辍 数	<div class="bza"> <spar	2024-12-23 15:58:08	0
7	鹤壁市“打造新时代高质量	<div class="bza"> <spar	2024-12-23 15:10:34	0
8	滁州学院成立人工智能学	<div class="bza"> <spar	2024-12-23 14:53:33	0
9	中建方践行央企责任 深	<div class="bza"> <spar	2024-12-23 14:51:26	0
10	“朋友圈”再扩大，“虹桥下	<div class="bza"> <spar	2024-12-23 14:47:51	0
11	拥抱数字文化与智能视听	<div class="bza"> <spar	2024-12-23 14:39:33	0
12	更好满足上班族三餐需求，	<div class="bza"> <spar	2024-12-23 14:00:01	0
13	联想发布系列服务器产品！	<div class="bza"> <spar	2024-12-23 13:43:58	0
14	故宫博物院与中国联通签	<div class="bza"> <spar	2024-12-23 12:20:59	0

使用 MySQL 进行数据库的设计与构建。

四、 主要代码展示

(一) 前端部分

前端代码最主要的部分就是与后端交互并且将后端传入的数据写入网页中，在前端中主要通过登录接口、注册接口、新闻列表接口、团队成员接口、点赞接口来实现主要功能，完成前后端的交互，以下为具体的实现代码：

，登录接口’

```
1 // 模拟登录接口
2 Mock.mock('/api/login', 'post', (options) => {
3   const { username, password } = JSON.parse(options.body)
4   if (username === 'admin' && password === '123456') {
5     return {
6       code: 0,
7       message: '登录成功',
8       token: 'mock-token-' + Date.now()
9     }
10  }
11  return {
12    code: 1,
13    message: '用户名或密码错误'
14  }
15 })
```

，注册接口’

```
1 // 模拟注册接口
2 Mock.mock('/api/register', 'post', (options) => {
3   const { username, password } = JSON.parse(options.body)
4   if (username && password) {
```

```
5     return {
6         code: 0,
7         message: '注册成功'
8     }
9 }
10 return {
11     code: 1,
12     message: '注册失败'
13 }
14 })
```

' 新闻列表接口'

```
1 // 新闻列表接口
2 Mock.mock('/api/news/list', 'get', {
3     code: 0,
4     message: '获取成功',
5     'data|8': [{
6         'id|+1': 1,
7         title: '@ctitle(10, 20)',
8         description: '@cparagraph(2)',
9         'image|1': [
10             'https://picsum.photos/200/150?random=1',
11             'https://picsum.photos/200/150?random=2',
12             'https://picsum.photos/200/150?random=3',
13             'https://picsum.photos/200/150?random=4'
14         ],
15         time: '@date("yyyy-MM-dd")',
16         'likes|0-200': 0,
17         isLiked: false,
18         'author': '@cname',
19         'category|1': ['科技', '娱乐', '体育', '财经', '教育'],
20         'tags|1-3': ['@ctitle(2,4)']
21     }]
22 })
```

' 团队成员接口'

```
1 // 团队成员接口
2 Mock.mock('/api/team/members', 'get', {
3     code: 0,
4     message: '获取成功',
5     data: [
6         {
7             id: 1,
8             name: '张三',
9             position: '技术总监',
10            description: '拥有10年以上互联网开发经验，精通前端技术栈，曾主导多个大型项目的架构设计与开发。',

```



```

11     avatar: 'https://picsum.photos/200/200?random=1',
12     'skills |4': ['@ctitle(2,4)']
13 },
14 ]
15 })

```

’点赞接口’

```

1 // 新闻点赞接口
2 Mock.mock(/\/api\/news\/like\/\d+/, 'post', {
3     code: 0,
4     message: '操作成功',
5     data: {
6         'likes | +1': 1
7     }
8 })

```

(二) 后端部分

后端完成了两件事情，第一件事情是连接数据库并在后端页面编写了一些 view 视图文件进行后端页面的展示，其中还可以对数据库进行增删改查的操作：以用户表为例，具体是通过控制器实现增删改查的方法，例如：

’控制器增删改查’

```

1 public function actionCreate()
2 {
3     $model = new Users();
4
5     if (\Yii::$app->request->isPost) {
6         if ($model->load(\Yii::$app->request->post()) && $model->save())
7             {
8                 return \Yii::$app->redirect(['view', 'UserID' => $model->
9                     UserID]);
10             }
11     } else {
12         $model->loadDefaultValues();
13     }
14
15     return $this->render('create', [
16         'model' => $model,
17     ]);
18 }
19 public function actionUpdate($UserID)
20 {
21     $model = $this->findModel($UserID);
22
23     if (\Yii::$app->request->isPost && $model->load(\Yii::$app->request->
24         post()) && $model->save()) {
25
26     }
27 }

```

```

22         return $this->redirect(['view', 'UserID' => $model->UserID]);
23     }
24
25     return $this->render('update', [
26         'model' => $model,
27     ]);
28 }
29 public function actionDelete($UserID)
30 {
31     $this->findModel($UserID)->delete();
32
33     return $this->redirect(['index']);
34 }
35 protected function findModel($UserID)
36 {
37     if (($model = Users::findOne(['UserID' => $UserID])) !== null) {
38         return $model;
39     }
40
41     throw new NotFoundException('The requested page does not exist.')
42     ;
43 }

```

第二件事情就是和前端进行连接，将处理好的数据传递给前端，具体实现方法是通过给前端提供 api 接口实现的：首先解决好跨域问题，这里网上有很多文档不赘述，然后定义好 api 接口的规则，例如：

‘定义 api 接口’

```

1     'urlManager' => [
2         'enablePrettyUrl' => true,
3         'showScriptName' => false,
4         'rules' => [
5             'api/login' => 'api/login',
6             'api/signup' => 'api/signup',
7             'api/adminlogin' => 'api/adminlogin',
8             'api/getarticle' => 'api/getarticle',
9             'api/getarticlecomment' => 'api/getarticlecomment',
10            'api/addarticlecomment' => 'api/addarticlecomment',
11            'api/getpersonalinfo' => 'api/getpersonalinfo',
12            'api/getarticlelikes' => 'api/getarticlelikes',
13            'api/addarticlelikes' => 'api/addarticlelikes',
14        ],
15    ],

```

左侧为前端路由调用的 api，右侧为后端提供的 api，然后再实现对应的 api 控制器，以获取文章的 api 接口为例：

‘获取文章 api 接口’

```

1     public function actionGetarticle()

```

```

2      {
3          \Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
4
5          // 获取页数
6          $page = \Yii::$app->request->get('page');
7          $intpage = (int)$page;
8          $id = \Yii::$app->request->get('id');
9          if ($id !== null) {
10             $articles = Articles::find()->select(['ArticleID', 'Title', '
                Content', 'PublicationDate', 'LikesCount'])->where(['ArticleID
                ' => $id])->one();
11         }
12         else{
13             // 查询数据库获取对应页数的文章信息
14             $articles = Articles::find()->select(['ArticleID', 'Title', '
                Content', 'PublicationDate', 'LikesCount'])->offset(15*(
                $intpage-1))->limit(15)->all();
15         }
16         // 格式化为 JSON 并返回
17         return $articles;
18     }

```

后端通过对数据库表的处理后，返回一个 JSON 的响应给前端，前端可以通过 axios 的 post 请求得到这个响应，然后根据响应返回的内容对前端页面进行相应的处理。

(三) 数据库

使用 MySQL 进行数据库的设计与建立，部分主要代码如下：

‘数据库代码’

```

1 SET NAMES utf8mb4;
2 SET FOREIGN_KEY_CHECKS = 0;
3
4 —————
5 — Table structure for admins
6 —————
7 DROP TABLE IF EXISTS admins;
8 CREATE TABLE admins (
9     AdminID int NOT NULL AUTO_INCREMENT,
10     Username varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
11     PRIMARY KEY (AdminID) USING BTREE,
12     INDEX admins_ibfk_1 (Username ASC) USING BTREE,
13     CONSTRAINT admins_ibfk_1 FOREIGN KEY (Username) REFERENCES users (username) ON
        DELETE RESTRICT ON UPDATE RESTRICT
14 ) ENGINE = InnoDB AUTO_INCREMENT = 2 CHARACTER SET = utf8mb4 COLLATE =
        utf8mb4_general_ci ROW_FORMAT = DYNAMIC;
15
16 —————
17 — Records of admins

```

```

18  —————
19  INSERT INTO admins VALUES (1, 'admin');
20
21  —————
22  — Table structure for articlecomments
23  —————
24  DROP TABLE IF EXISTS articlecomments;
25  CREATE TABLE articlecomments (
26      CommentID int NOT NULL AUTO_INCREMENT,
27      ArticleID int NOT NULL,
28      Comment text CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
29      CommentDate timestamp NULL DEFAULT CURRENT_TIMESTAMP,
30      Username varchar(50) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
31      PRIMARY KEY (CommentID) USING BTREE,
32      INDEX ArticleID(ArticleID ASC) USING BTREE,
33      INDEX fk_article_comments_username(Username ASC) USING BTREE,
34      CONSTRAINT article_comments_ibfk_2 FOREIGN KEY (ArticleID) REFERENCES articles (
          articleid) ON DELETE RESTRICT ON UPDATE RESTRICT,
35      CONSTRAINT fk_article_comments_username FOREIGN KEY (Username) REFERENCES users
          (username) ON DELETE RESTRICT ON UPDATE RESTRICT
36  ) ENGINE = InnoDB AUTO_INCREMENT = 13 CHARACTER SET = utf8mb4 COLLATE =
          utf8mb4_general_ci ROW_FORMAT = DYNAMIC;
37
38  —————
39  — Records of articlecomments
40  —————
41  INSERT INTO articlecomments VALUES (10, 9, '还行', '2024-12-24 20:39:28', 'user
          ');
42  INSERT INTO articlecomments VALUES (11, 13, '挺好', '2024-12-24 20:42:22', 'user
          ');
43  INSERT INTO articlecomments VALUES (12, 1, '还不错', '2024-12-24 21:22:50', '
          user123');
44
45  —————
46  — Table structure for articles
47  —————
48  DROP TABLE IF EXISTS articles;
49  CREATE TABLE articles (
50      ArticleID int NOT NULL AUTO_INCREMENT,
51      Title varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
52      Content text CHARACTER SET utf8 COLLATE utf8_general_ci NULL,
53      PublicationDate timestamp NULL DEFAULT CURRENT_TIMESTAMP,
54      LikesCount int NULL DEFAULT NULL,
55      PRIMARY KEY (ArticleID) USING BTREE
56  ) ENGINE = InnoDB AUTO_INCREMENT = 351 CHARACTER SET = utf8 COLLATE =
          utf8_general_ci ROW_FORMAT = DYNAMIC;
57
58  —————

```

59 — Records of articles

60 —

实现了如下表项:

数据库实现



进行数据抓取获取文章数据时写了 Python 脚本从“人民网搜索”(http://search.people.cn)上抓取关于人工智能的搜索结果, 主要代码如下:

’ 获取文章 api 接口’

```

1  import datetime
2  import re
3
4  import pandas as pd
5  import requests
6  from bs4 import BeautifulSoup
7
8  cookies = {
9      '__jsluid_h': '3d8df91ed2a29590b298231f57fc02d5',
10     'sso_c': '0',
11     'sfr': '1',
12 }
13
14 headers = {
15     'Accept': 'application/json, text/plain, */*',
16     'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8',
17     'Cache-Control': 'no-cache',
18     'Connection': 'keep-alive',
19     'Content-Type': 'application/json; charset=UTF-8',
20     '# 'Cookie': '__jsluid_h=3d8df91ed2a29590b298231f57fc02d5; sso_c=0; sfr=1',
21     'Origin': 'http://search.people.cn',
22     'Pragma': 'no-cache',
23     'Referer': 'http://search.people.cn/s?keyword=%E8%B5%B5%E7%B4%AB%E9%98%B3

```

```
%E8%B0%83%E7%A0%94&st=0&_ =1730518116527',
24 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit
    /537.36 (KHTML, like Gecko) Chrome/130.0.0.0 Safari/537.36',
25 }
26 key_word = ["人工智能"]
27 data_all = []
28 for key in key_word:
29
30     json_data = {
31         'key': key,
32         'page': 1,
33         'limit': 100,
34         'hasTitle': True,
35         'hasContent': True,
36         'isFuzzy': False,
37         'type': 0,
38         'sortType': 0,
39         'startTime': 0,
40         'endTime': 0,
41     }
42     response = requests.post(
43         'http://search.people.cn/search-platform/front/search',
44         cookies=cookies,
45         headers=headers,
46         json=json_data,
47         verify=False,
48     )
49     res = response.json()['data']['records']
50     for item in res:
51         data_one = {}
52         print(item)
53         #标题, 链接, 时间, 内容
54         # data_one['标题'] = item['title']
55         data_one['标题'] = re.sub(r'<[^>+>', '', item['title']).replace("&
            nbsp;", " ").replace("&gt;", ">").replace(
56             "&lt;", "<").replace("&quot;", "'").replace("&";", "&")
57
58         # data_one['链接'] = item['url']
59         # 发布时间是时间戳, 需要转换为年-月-日
60         # data_one['发布时间'] = item['displayTime']
61         # 时间戳转换为年-月-日
62         timestamp = int(item['displayTime'])
63         # timestamp = int(data_one['发布时间'])
64         # 判断是否为毫秒级时间戳
65         if timestamp > 1e12:
66             timestamp /= 1000 # 转换为秒级时间戳
67         formatted_time = datetime.datetime.fromtimestamp(timestamp).strftime
            ("%Y-%m-%d %H:%M:%S")
```

```
68     data_one['发布时间'] = formatted_time
69     # 根据链接获取内容
70     res_content = requests.get(item['url'], headers=headers, cookies=
        cookies)
71     if "charset=GB2312" in res_content.text:
72         res_content.encoding = 'GB2312'
73     elif "charset=UTF-8" in res_content.text:
74         res_content.encoding = 'utf-8'
75     with open('test.html', 'w', encoding='utf-8') as f:
76         f.write(res_content.text)
77     soup = BeautifulSoup(res_content.text, 'html.parser')
78     # 内容可能在不同的类中, 有时候是: rm_txt_con, 有时候是在: text_con,
        有时候在text_c
79     classnames = ["rm_txt_con", "text_con", "text_c"]
80     content = None
81
82     for classname in classnames:
83         content_element = soup.find(class_=classname)
84         if content_element:
85             content = content_element.decode_contents()
86             break
87     data_one['内容'] = content
88     data_all.append(data_one)
89     print("finish", key)
90 df = pd.DataFrame(data_all)
91 df.to_excel("人工智能.xlsx")
```

发送 HTTP 请求:

使用 requests.post 方法发送 POST 请求请求中包含搜索参数(如关键词、页码等)以及必要的 Cookies 和 Headers。

使用 requests.get 方法发送 GET 请求到每个搜索结果的 URL, 以获取完整的网页内容。

解析响应:

对于 POST 请求的响应, 使用.json() 方法将其解析为 JSON 对象, 并从中提取 data.records 字段, 它包含了搜索结果的具体信息。

对于 GET 请求的响应, 首先检查网页内容中的字符集信息, 并相应地设置正确的编码。

提取数据:

从每个搜索结果中提取标题、发布时间和 URL。

标题使用正则表达式进行清洗, 去除 HTML 标签和特殊字符。

发布时间是一个时间戳, 需要将其转换为可读的日期时间格式。

使用 BeautifulSoup 解析网页内容, 尝试从多个可能的类名中提取文章正文。

存储数据:

将提取到的数据存储到字典中, 然后将这些字典添加到列表中。

使用 pandas 库将列表转换为 DataFrame, 并将其保存到 Excel 文件中。