实现文档

前期规划

在项目实现之前，我们团队首先通过飞书会议进行了详细讨论，并确定网站的技术架构 ——— Vue 和 Yii2 构建的前后端分离模式。

同时组长对项目的整体工作进行模块化拆分，并在 Tower 上进行了每位组员的任务分配。

任务分工

2212023 李嘉桐 （组长）

- 统筹和完善整体工作
- 完成登录注册功能
- 负责文章专区功能的实现（包括检索、点赞、评论）与信息收集
- 完成相应后台数据管理以及需求文档撰写。

2211819 杨峥芃 （组员）

- 完成团队主页
- 网页浏览量记录功能的实现
- 完成相应后台数据管理
- 项目展示PPT制作

2211820 高艺轩 （组员）

- 视频数据爬取
- 视频专区功能的实现
- 完成相应后台数据管理
- 设计文档和用户手册撰写

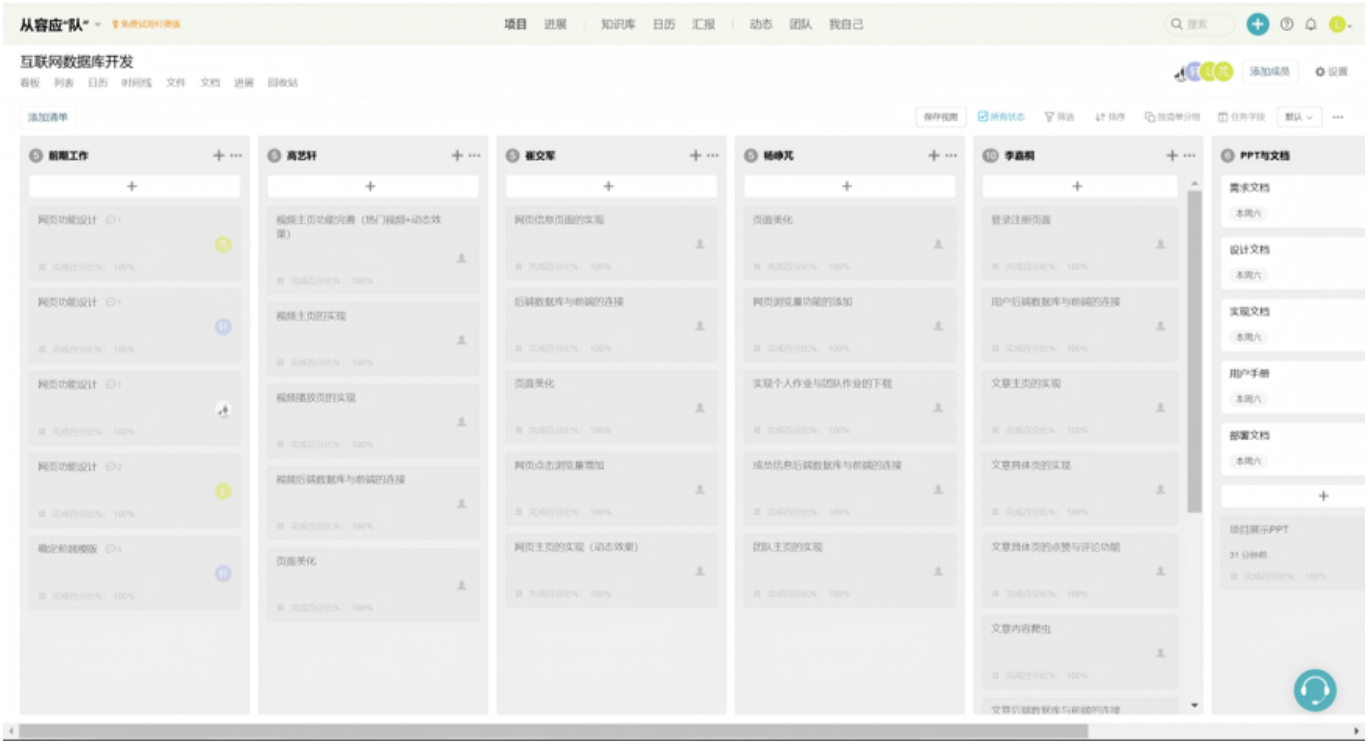
2210199 崔交军 （组员）

- 负责网站主页和项目介绍界面
- 网页浏览量记录功能的实现
- 完成相应后台数据管理
- 实现文档和部署文档撰写

完成过程

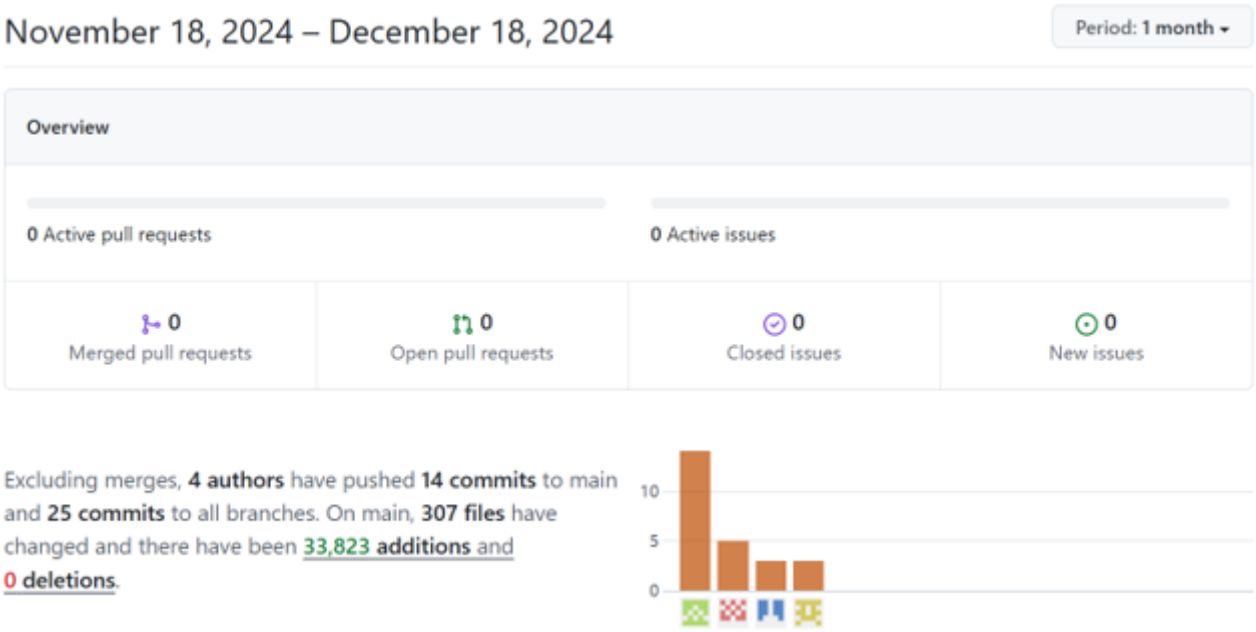
团队分别于11月10日、11月21日、12月10日、12月16日和12月18日进行了多次线下讨论，确保了项目技术问题及时解决、团队协作顺畅，保障了项目顺利推进。

tower分工记录



通过在 Tower 上明确任务分工、按时完成任务和清晰的模块设计，团队高效推动了项目的顺利实施。

git提交记录

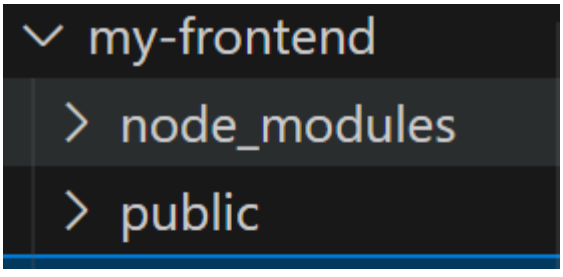


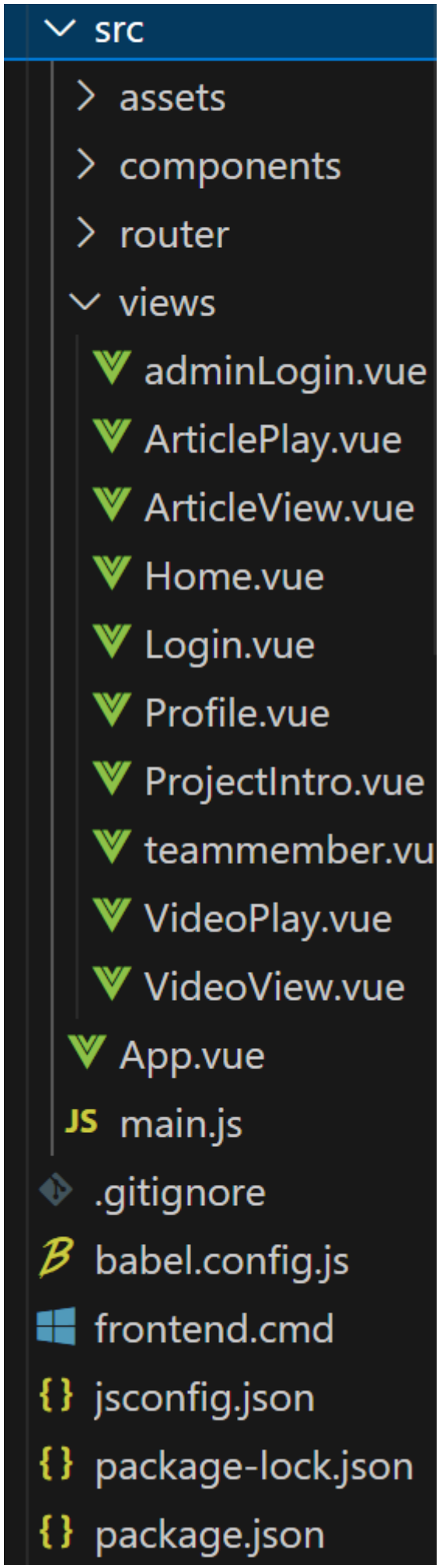

```
38fe2f0 - ljtZM, 5 minutes ago : 修复bug
a4b30f4 - ljtZM, 29 hours ago : 完善
19f2af7 - ljtZM, 3 days ago : team表+website表
e01ec28 - ljtZM, 3 days ago : team表+website表
f306039 - ljtZM, 5 days ago : 合并
2aae878 - ljtZM, 5 days ago : 撤销大文件
c16d17f - ljtZM, 5 days ago : 合并
6273a09 - ljtZM, 5 days ago : 合并
330cc56 - ljtZM, 5 days ago : 合并
ecdfef5 - ljtZM, 6 days ago : 样式修改
67bae44 - ljtZM, 6 days ago : 完善backend/views
5970f77 - ljtZM, 6 days ago : backend/views完善
48a68ac - ljtZM, 6 days ago : 合并
beb81f7 - FilotimoGao, 7 days ago : 修改文件名
34a3333 - FilotimoGao, 7 days ago : 撰写ApiControllers.php
e326e7b - ljtZM, 7 days ago : 合并
6135993 - pp, 7 days ago : 2211819_update on the memberpage
4c94986 - 崔交军, 7 days ago : 添加团队主页并且成功显示浏览量
f02ac5d - FilotimoGao, 7 days ago : update controllers and models
9dafcc1 - ljtZM, 7 days ago : 后台管理界面+文章点赞评论
56893c9 - ljtZM, 8 days ago : 实现文章评论+点赞
195f1e0 - ljtZM, 8 days ago : 合并
ac60c69 - 1223BETTYYANG, 8 days ago : update web.php
d97a03a - 1223BETTYYANG, 8 days ago : update web.php
e78d805 - 崔交军, 8 days ago : HomePage+Webviews
0df7231 - 1223BETTYYANG, 8 days ago : teaminfo ver1.0
29ea2cc - 1223BETTYYANG, 8 days ago : teaminfo ver1.0
58aa37c - FilotimoGao, 8 days ago : branch-2211820
2587a85 - ljtZM, 10 days ago : 数据库表
1d8700b - ljtZM, 11 days ago : yii框架+vue前端
7540953 - ljtZM, 5 weeks ago : initial
```

通过多次提交和协同作业，团队共修改了3,3823行代码。

文件目录展示

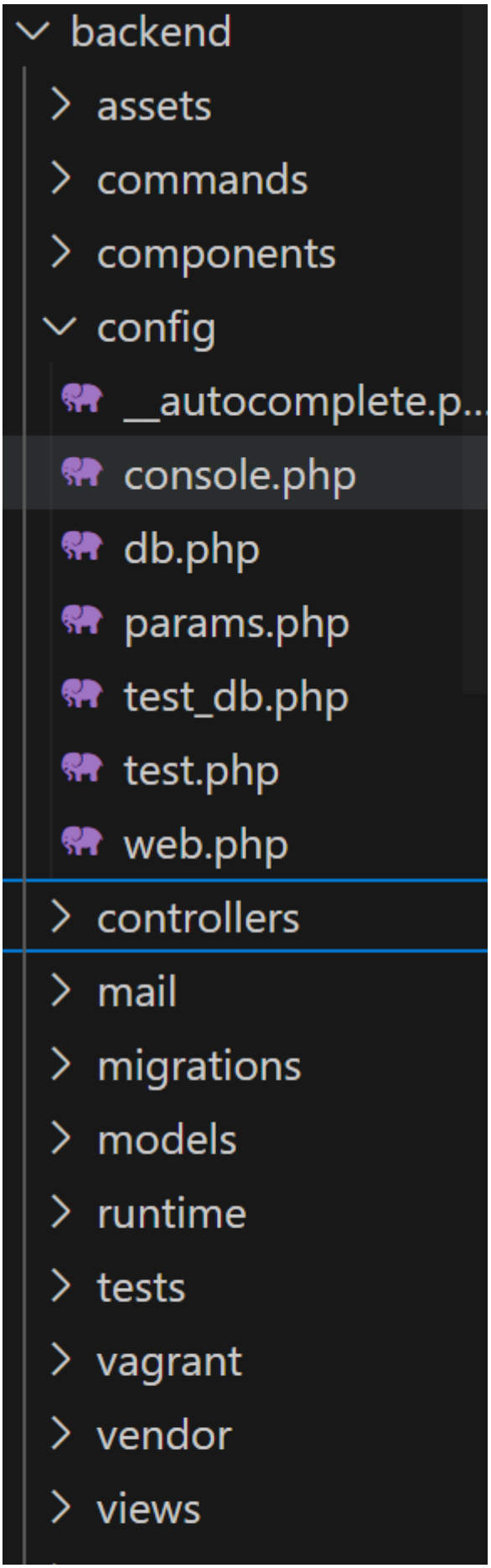
前端目录

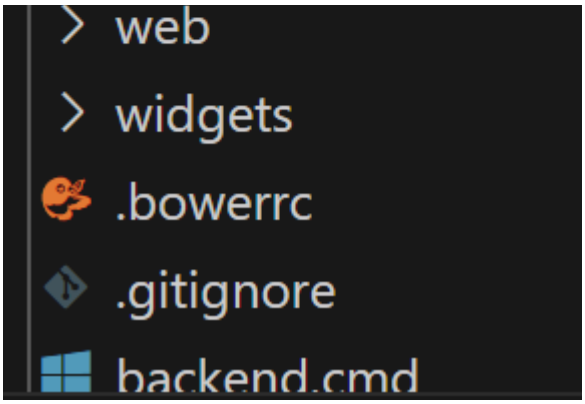



src
assets
components
router
views
adminLogin.vue
ArticlePlay.vue
ArticleView.vue
Home.vue
Login.vue
Profile.vue
ProjectIntro.vue
teammember.vu
VideoPlay.vue
VideoView.vue
App.vue
main.js
.gitignore
babel.config.js
frontend.cmd
jsconfig.json
package-lock.json
package.json

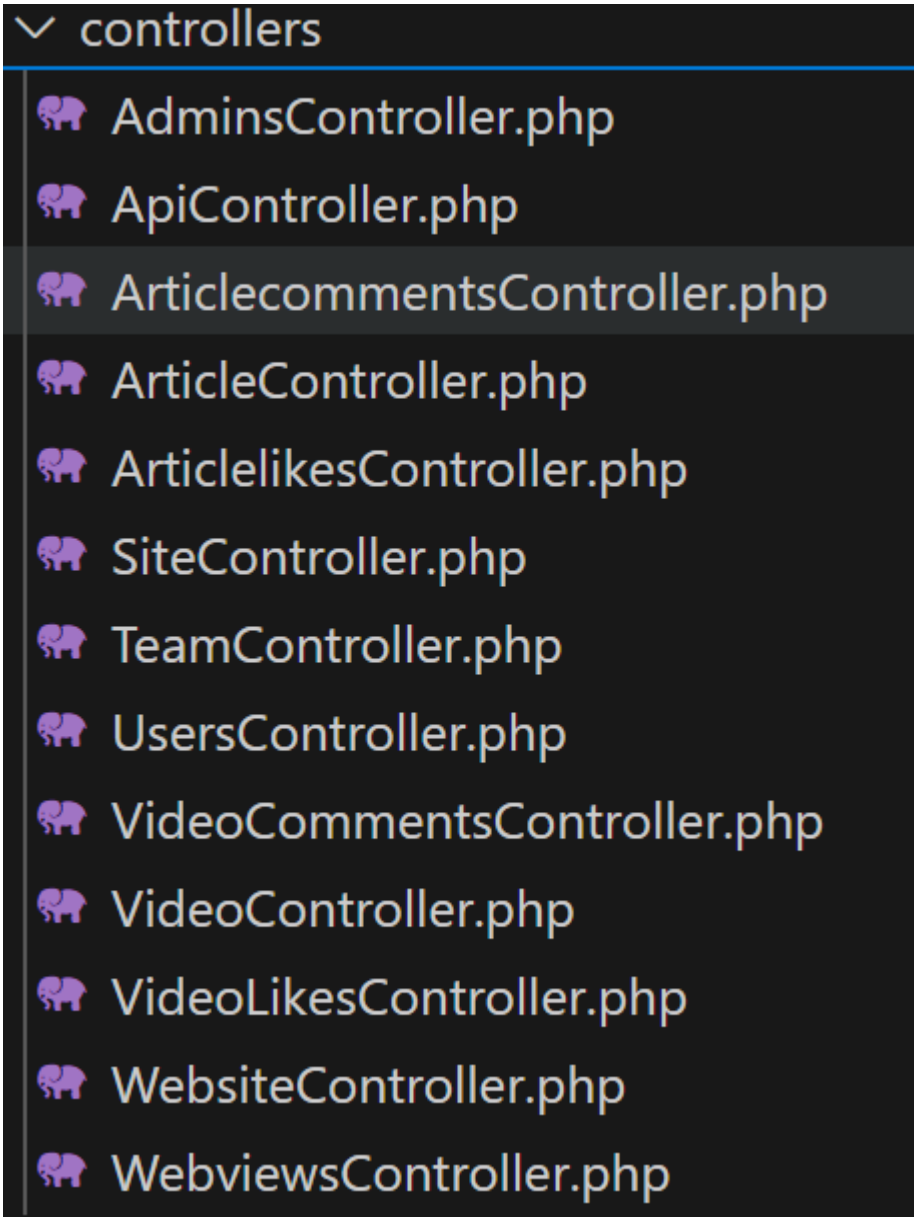
其中src目录是项目的核心，包含了项目的主要代码。其中views是每个页面的代码，components是网页代码可以嵌入的模块，App.vue是项目的主入口。

后端目录

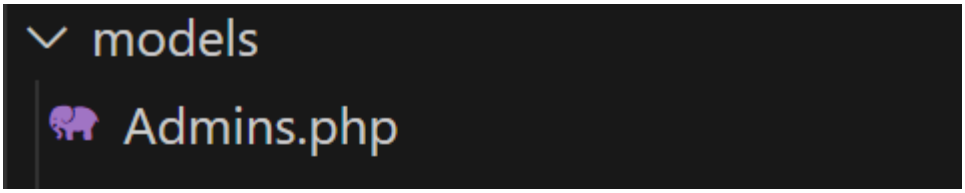


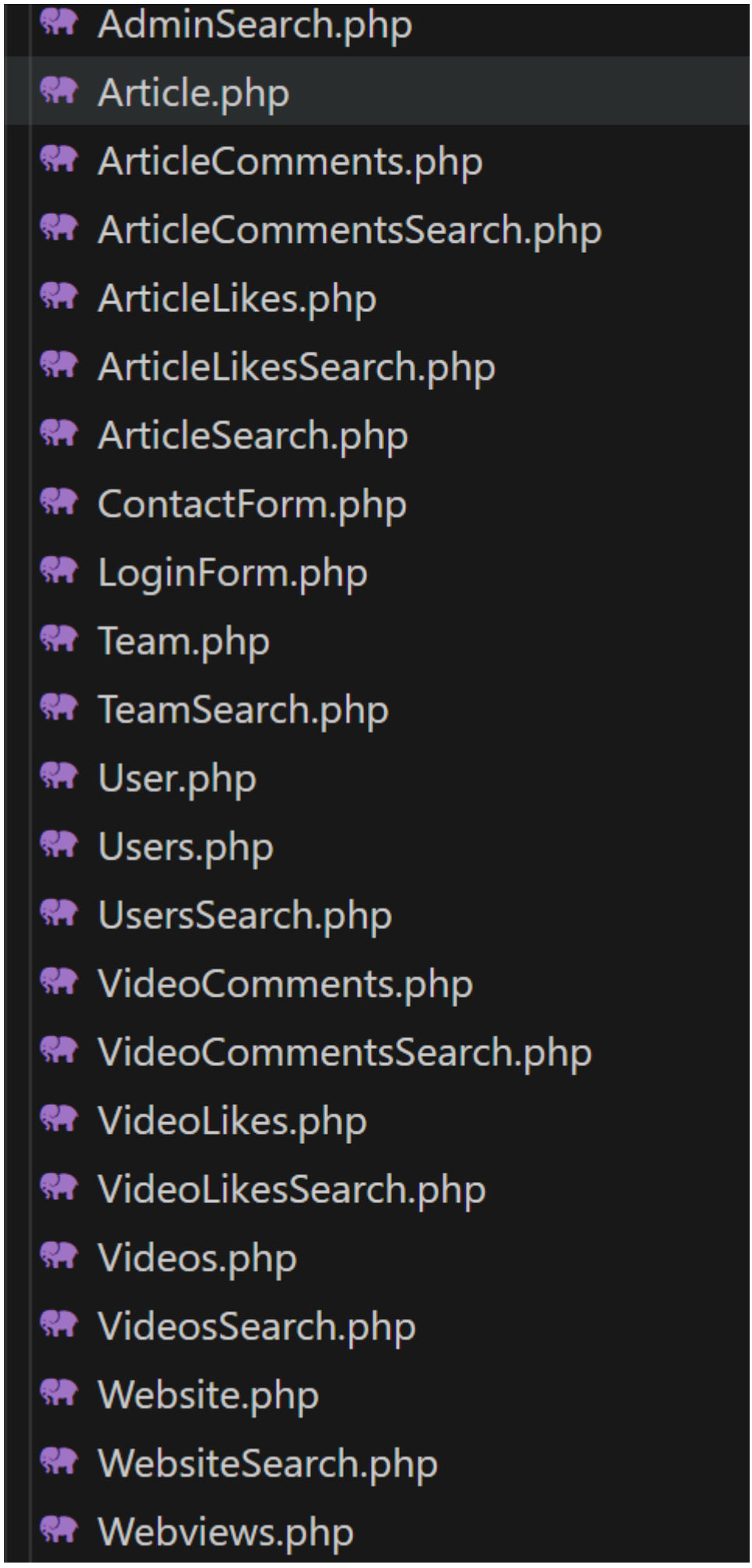


config 文件夹包含应用程序的配置文件，用于设置不同环境下的运行参数，如数据库用户名、密码、跨域设置等。

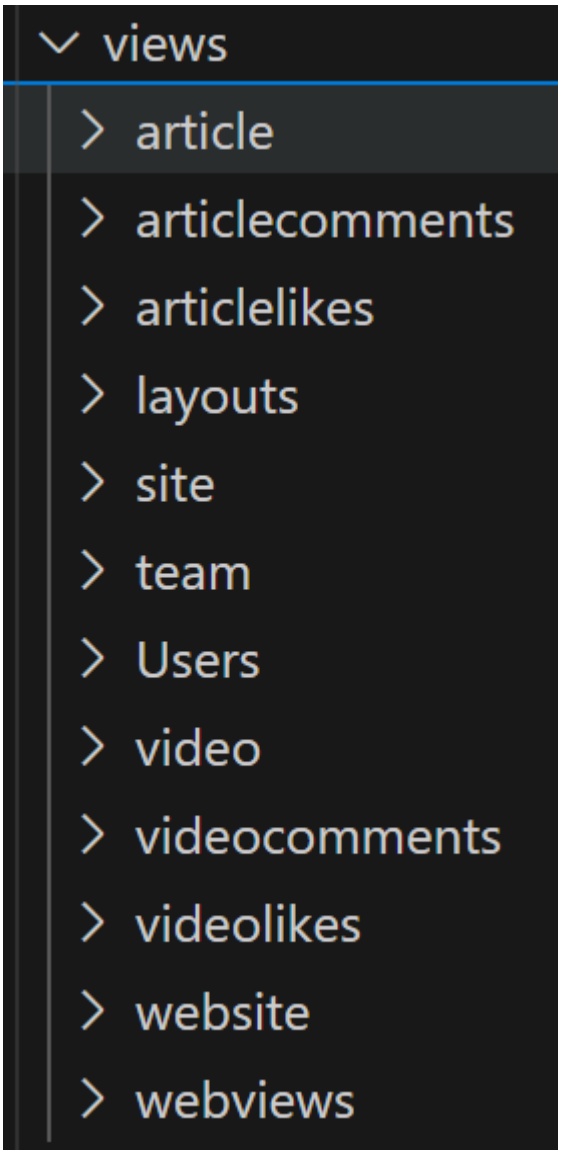


controllers 文件夹用于存放控制器类文件，负责接收用户请求、处理逻辑并返回相应的视图或数据。



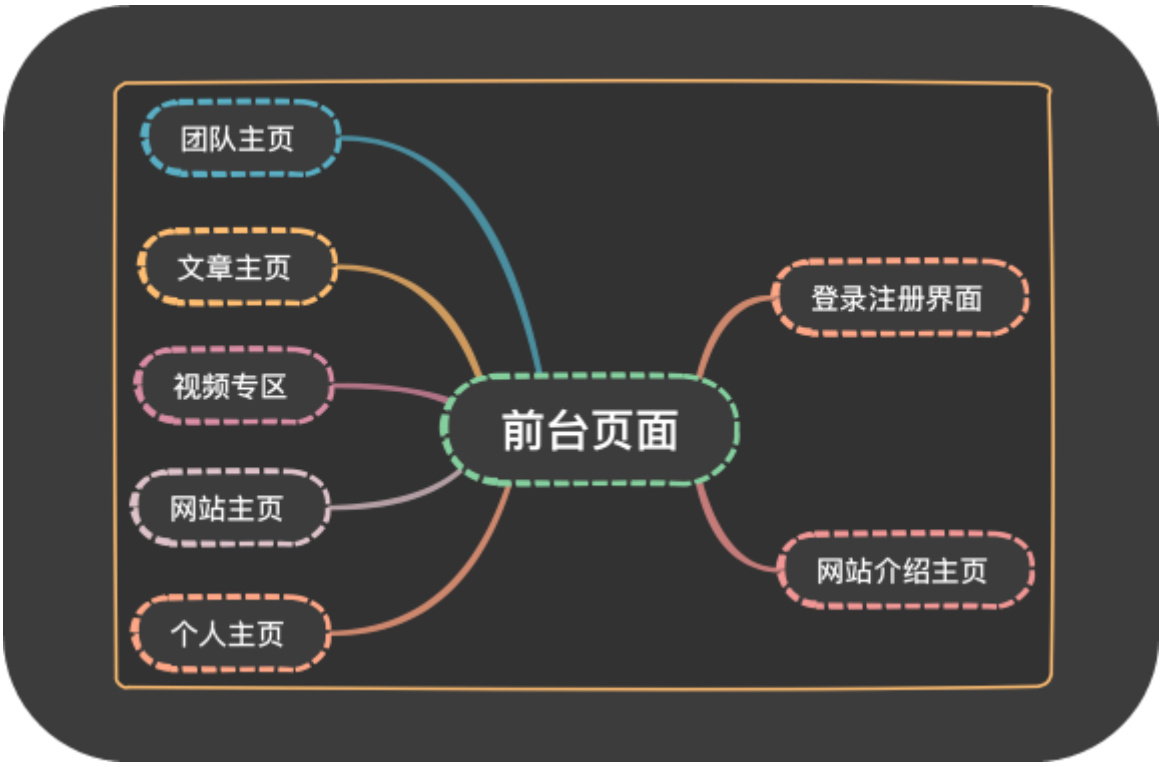

AdminSearch.php
Article.php
ArticleComments.php
ArticleCommentsSearch.php
ArticleLikes.php
ArticleLikesSearch.php
ArticleSearch.php
ContactForm.php
LoginForm.php
Team.php
TeamSearch.php
User.php
Users.php
UsersSearch.php
VideoComments.php
VideoCommentsSearch.php
VideoLikes.php
VideoLikesSearch.php
Videos.php
VideosSearch.php
Website.php
WebsiteSearch.php
Webviews.php

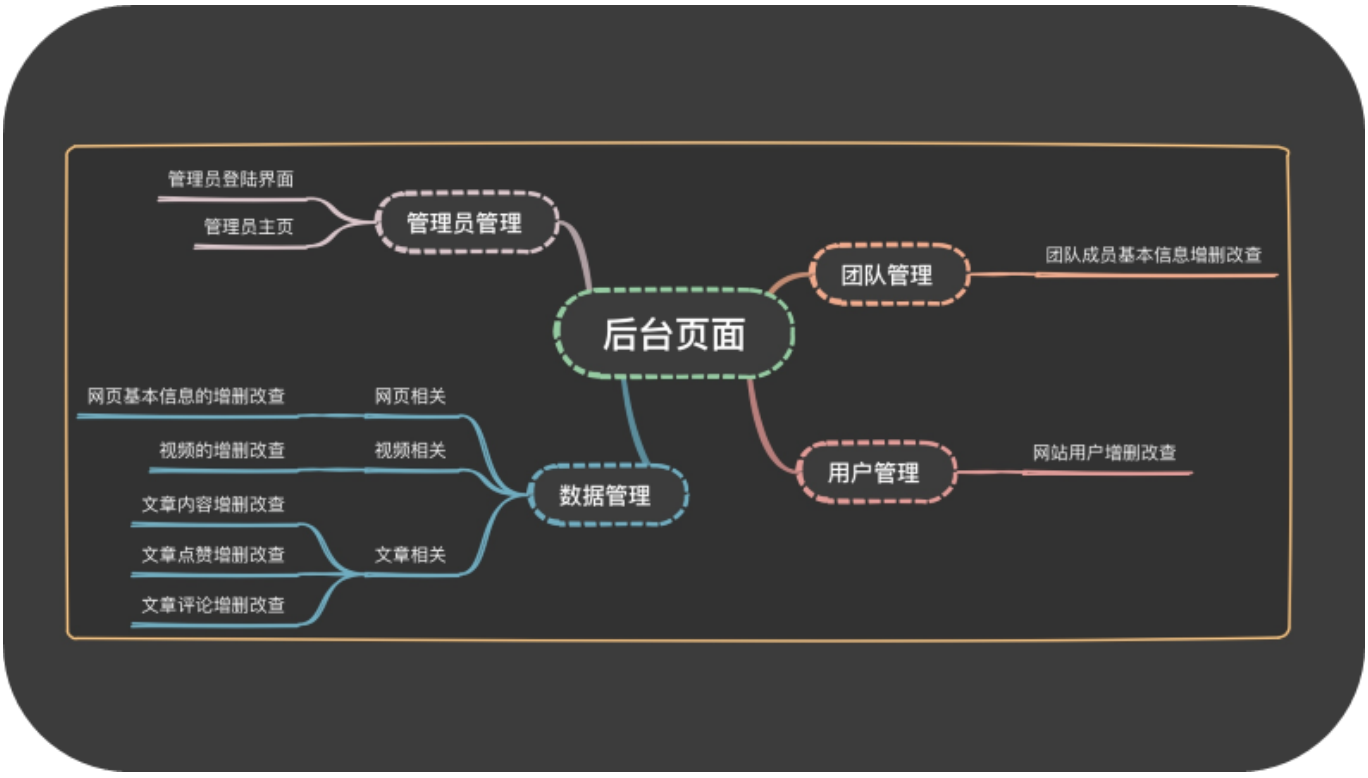
models 文件夹存放模型类文件，负责定义与数据库表的映射，处理数据的增删改查操作。



views 文件夹存放视图文件，负责展示数据和页面，通常与用户交互界面相关，接收控制器传递的数据并渲染成 HTML 页面。

网站页面设计





关键代码展示

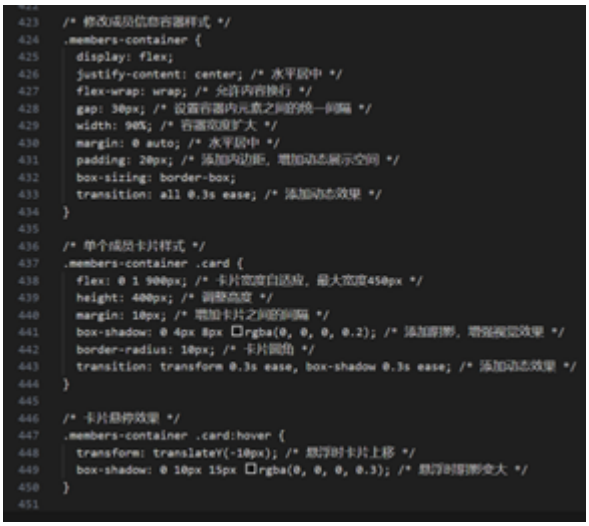
页面样式设计

主页设计包括动态图片轮播、特殊字体和图案，通过精心设计的布局和动画效果，提升用户视觉体验。

```vue
<!-- 图片轮播 -->
<div class="carousel-container">
  <el-carousel :interval="5000" type="card" height="60vh" trigger="click">
    <el-carousel-item v-for="(imagePath, index) in imagePaths" :key="index">
      <img :src="imagePath" alt="carousel image" />
    </el-carousel-item>
  </el-carousel>
</div>
```

```css
.election-text {
  font-size: 34px;
  color: white;
  font-family: 'Special Elite', cursive; /* 使用特别字体 */
  font-weight: bold;
}
/* 倾斜且加长的渐变分隔符 */
.separator {
  display: inline-block;
  width: 6px;
  height: 80px;
  background: linear-gradient(to bottom, #ff7e5f, #feb47b, #6a11cb, #2575fc);
  border-radius: 2px;  /* 圆角效果 */
  margin: 0 15px;
  transform: rotate(8deg);  /* 倾斜角度 */
  transform-origin: center;
}
.subtitle {
  font-size: 24px;
  color: rgba(255, 255, 255, 0.7);  /* 字体颜色为白色并带有透明度，0.7为微透明 */
  margin-left: 5px;
  font-family: 'Arial', sans-serif;
  font-weight: lighter;
  font-style: italic;
}
```

团队主页通过合理的元素布局与动效呈现，增强页面互动性。



```css
/* 修改成员信息容器样式 */
.members-container {
  display: flex;
  justify-content: center; /* 水平居中 */
  flex-wrap: wrap; /* 允许内容换行 */
  gap: 30px; /* 设置容器内元素之间的统一间隔 */
  width: 90%; /* 容器宽度扩大 */
  margin: 0 auto; /* 水平居中 */
  padding: 20px; /* 添加内边距，增加动态展示空间 */
  box-sizing: border-box;
  transition: all 0.3s ease; /* 添加动态效果 */
}

/* 单个成员卡片样式 */
.members-container .card {
  flex: 0 1 900px; /* 卡片宽度自适应，最大宽度450px */
  height: 400px; /* 调整高度 */
  margin: 10px; /* 增加卡片之间的间隔 */
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* 添加阴影，增强视觉效果 */
  border-radius: 10px; /* 卡片圆角 */
  transition: transform 0.3s ease, box-shadow 0.3s ease; /* 添加动态效果 */
}

/* 卡片悬停效果 */
.members-container .card:hover {
  transform: translateY(-10px); /* 悬浮时卡片上移 */
  box-shadow: 0 10px 15px rgba(0, 0, 0, 0.3); /* 悬浮时阴影变大 */
}
```

前后端交互

这里以网页浏览量为例，使用axios实现异步请求管理以及支持并发请求，并通过 SessionStorage 防止页面刷新或跳转时浏览量不必要地增加。

首先，函数获取当前路由，将其作为参数，通过 POST 请求发送给后端 API。后端处理请求后返回一个 JSON 格式的响应，前端读取响应中的数据并更新页面中相应的变量，从而完成前后端的数据交互。

```
import axios from 'axios';
export default {
  name: 'App',
  mounted() {
    // 检查 sessionStorage 是否已经发送过浏览量请求
    if (!sessionStorage.getItem('webviewSent')) {
      this.AddWebView();
      sessionStorage.setItem('webviewSent', 'true');
    } else {
      console.log("浏览量已增加，跳过");
    }
  },
  methods: {
    AddWebView() {
      axios
        .post('http://localhost:8080/api/addwebviews')
        .catch((error) => {
          console.error('浏览量增加失败', error);
        });
    },
  },
};
```

后端实现

后端为前端获取数据提供api接口如下：

```
'urlManager' => [
            'enablePrettyUrl' => true,
            'showScriptName' => false,
            'rules' => [
                'api/adminlogin' => 'api/adminlogin',
                'api/login' => 'api/authenticate',
                'api/signup' => 'api/register',
                'api/getarticle' => 'api/getarticle',
                'api/getarticlepagecount' => 'api/getarticlepagecount',
                'api/getarticlecomment' => 'api/getarticlecomment',
                'api/addarticlecomment' => 'api/addarticlecomment',
                'api/getarticlelikes' => 'api/getarticlelikes',
                'api/addarticlelikes' => 'api/addarticlelikes',
                'api/addwebviews' => 'api/addwebviews',
                'api/checkwebviews' => 'api/checkwebviews',
                'api/getmembers' => 'api/getmembers',
                'api/getdescription' => 'api/getdescription',
                'api/getvideo' => 'api/getvideo',
                'api/getvideocomments' => 'api/getvideocomments',
                'api/addvideocomments' => 'api/addvideocomments',
                'api/getvideolikes' => 'api/getvideolikes',
                'api/addvideolikes' => 'api/addvideolikes',
                'api/getclick' => 'api/getclick',
                'api/addclick' => 'api/addclick',
                'api/getpersonalinfo' => 'api/getpersonalinfo',
                'Users/index' => 'users/index',
                'article/index' => 'article/index',
                'articlecomments/index' => 'articlecomments/index',
                'articlelikes/index' => 'articlelikes/index',
```

```
                'videos/index' => 'video/index',
                'videocomments/index' => 'videocomments/index',
                'videolikes/index' => 'videolikes/index',
                'team/index' => 'team/index',
                'website/index' => 'website/index'
            ],
        ],
```

这里以文章点赞为例，前端通过axios请求http://localhost:8080/api/getarticlelikes，向后端发送包含文章ID的请求。Yii2后端通过actionGetarticlelikes()方法接收请求，查找或创建对应文章的点赞记录，并返回点赞数。对于增加文章点赞数，后端通过actionAddarticlelikes()方法处理，更新点赞数并返回更新结果。

```php
//获取文章点赞数
public function actionGetarticlelikes()
{
    \Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
    $articleID = \Yii::$app->request->get('article_id');
    $likes = ArticleLikes::find()
        ->where(['article_id' => $articleID])
        ->one();
    if ($likes === null) {
        // 如果没有找到点赞记录，则创建一条
        $likes = new ArticleLikes();
        $likes->article_id = $articleID;
        $likes->likes = 0;
        $likes->save();
    }
    return $likes->likes;
}

//增加文章点赞数
public function actionAddarticlelikes()
{
    \Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;
    $articleID = \Yii::$app->request->get('article_id');
    $num = \Yii::$app->request->get('num');
    $likes = ArticleLikes::find()
        ->where(['article_id' => $articleID])
        ->one();
    if ($likes === null) {
        // 如果没有找到点赞记录，则创建一条
        $likes = new ArticleLikes();
        $likes->article_id = $articleID;
        $likes->likes = 0;
        $likes->save();
    }
    // 增加点赞数
    $likes->likes += $num;
    if ($likes->save()) {
        return ['status' => 1, 'message' => 'Likes updated successfully'];
    } else {
        return ['status' => -1, 'message' => 'Failed to update likes'];
    }
}
```

数据爬取

这里以文章为例，爬取网站内容并保存到文件，然后将数据存储到数据库中。关键代码如下：

```
def scrape_articles():
    url = "https://search.cctv.com/search.php?qtext=%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD&sort=relevance&type=web&vtime=&datepid=1&ch
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.
    }
    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print("Failed to fetch the webpage")
        return []

    soup = BeautifulSoup(response.text, "html.parser")
    articles = []

    for li in soup.find_all("li", class_="image"):
        # 提取标题
        title_tag = li.find("a", id=lambda x: x and x.startswith("web_content_"))
        title = title_tag.text.strip() if title_tag else "No Title"

        # 提取文章链接
        link = title_tag["href"] if title_tag else None
        if link and link.startswith("link_p.php"):
            link = link.split("targetpage=")[1].split("&")[0]
            link = requests.utils.unquote(link)  # 解码链接

        # 提取摘要
        summary_tag = li.find("p", class_="bre")
        summary = summary_tag.text.strip() if summary_tag else "No Summary"

        # 提取发布时间
        time_tag = li.find("span", class_="tim")
        publish_time = time_tag.text.strip().replace("发布时间：", "") if time_tag else "Unknown"
```

```python
def upload_to_database(articles):
    connection = None
    try:
        connection = pymysql.connect(**DB_CONFIG)
        cursor = connection.cursor()
        # 插入文章数据
        insert_query = """
        INSERT INTO articles (title, content, summary, author, publication_date)
        VALUES (%s, %s, %s, %s, %s)
        """
        for article in articles:
            try:
                cursor.execute(
                    insert_query,
                    (
                        article["title"],
                        article["content"],
                        article.get("summary", ""),
                        article["author"],
                        article.get("publish_time", None),
                    ),
                )
            except Exception as e:
                print(f"Failed to insert article '{article['title']}': {e}")
        # 提交事务
        connection.commit()
        print(f"Successfully uploaded {cursor.rowcount} articles.")
    except Exception as e:
        print(f"Error connecting to database: {e}")
    finally:
        if connection:
            connection.close()
```