

实现文档

你说的都队

一、团队实施

1.分工

吴传杰：homepage&about&publish&后台

谢志颖：blogs 前后台&登录注册界面

田佳业：video&服务器部署&一键部署

2.项目管理

我们把项目放在服务器（IP 地址为 8.130.17.136）内，然后通过远程 SSH 连接修改服务器上的代码，这样的话大家同改一份代码，测试也通过访问此 IP 地址来进行测试。因此，我们的 git commit 记录很少。

本项目的 github 地址为：

https://github.com/Metreacs/Internet_database_development_R-U_War

二、个人模版实现

吴传杰

Article 模块

1、 功能介绍

对于浏览型用户：

- ① 浏览由其他用户发表的与俄乌战争有关的博客。
- ② 根据关键词搜索相关的文章。
- ③ 对某篇文章进行评论（需要注册登录网站）。

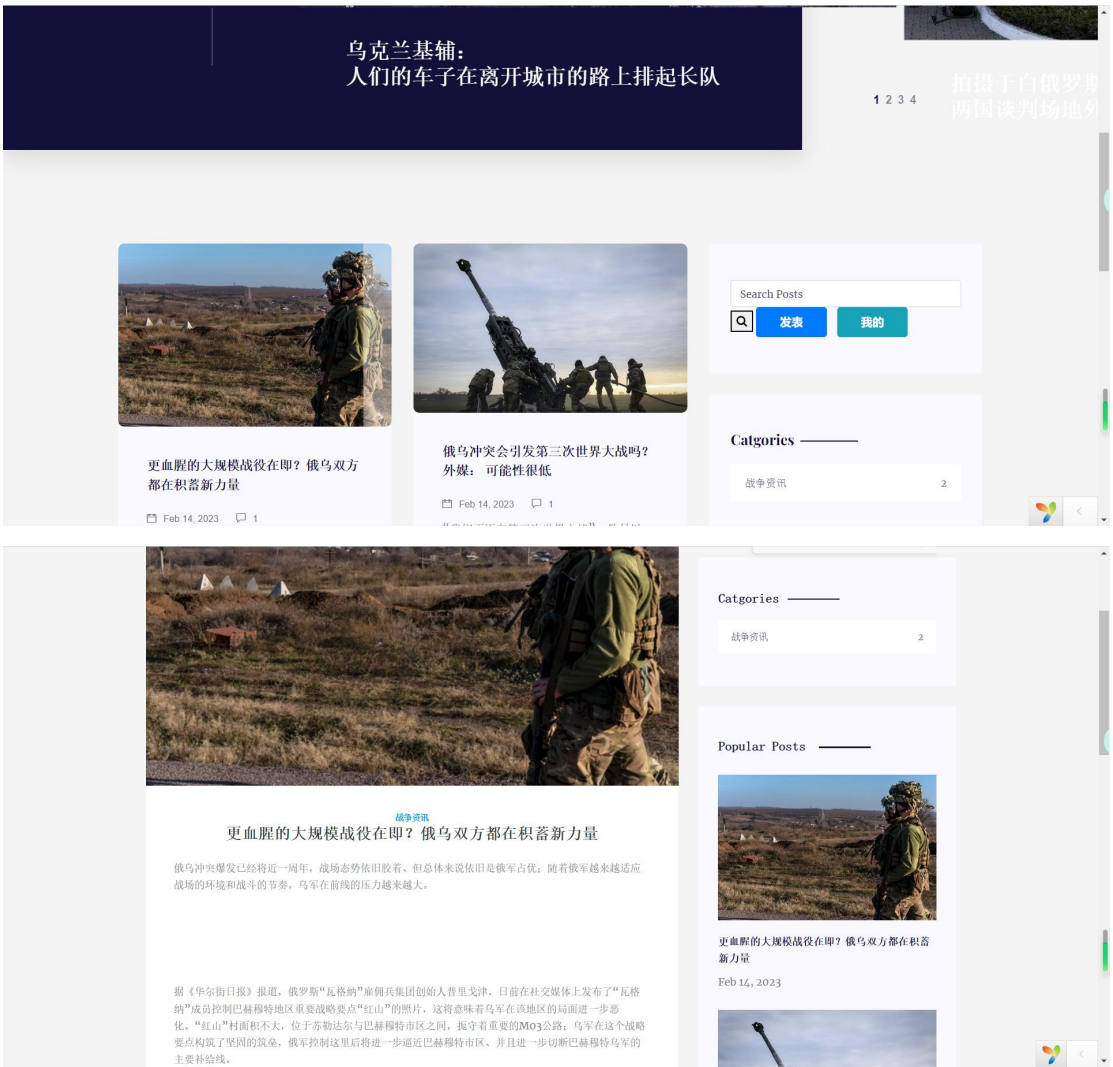
对于创作型用户：

- ① 发表与俄乌战争相关的文章（需要注册登录网站）。
- ② 上传对应的封面图，选择相应的类别。
- ③ 管理用户自己的文章，可以进行更新、删除。

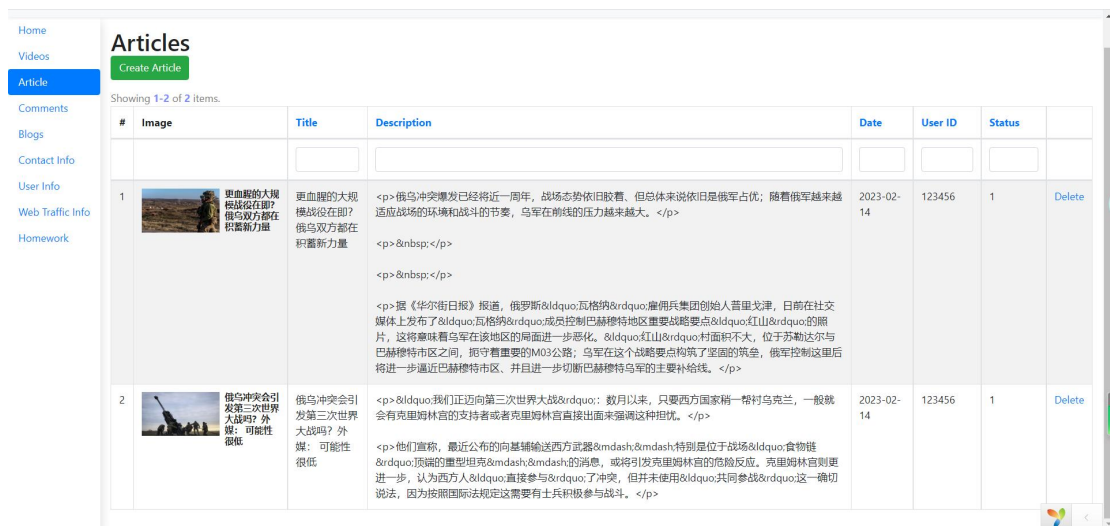
对于网站管理者：

- ① 审核用户上传的文章内容，决定是否发表。
- ② 对所有用户上传的文章有更新、删除的权限。
- ③ 审核用户发表的评论。

I. 前台：



后台：



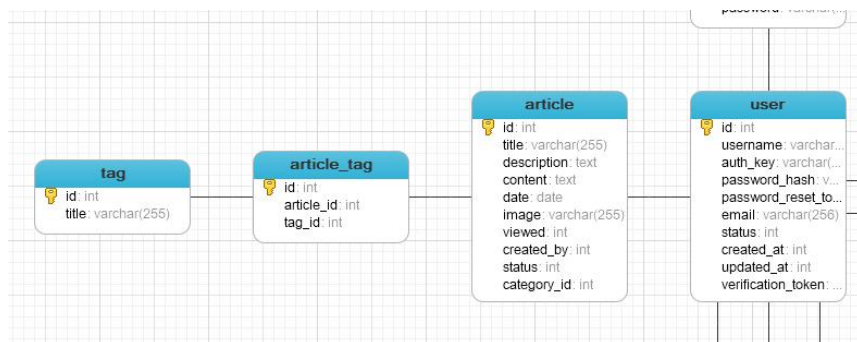
代码实现

1. 前台：

Model:

Model 主要由 gii 生成，几个重要的 model:

- ◎ common\models\Article.php => 对应表 article
- ◎ common\models\Category.php => 对应表 category
- ◎ common\models\Comment.php => 对应表 comment



Controller:

frontend\controllers\BlogController.php

几个 function 示例:

```
77 public function actionView($id)
78 {
79     $article = Article::findOne($id);
80
81     $popular = Article::Popular();
82
83     $recent = Article::Recent();
84
85     $categories = Category::find()->all();
86
87     $comments = $article->ArticleComments();
88     $commentForm = new CommentFormB();
89
90     $article->viewedCounter();
91
92     return $this->render('view', [
93         'article' => $article,
94         'popular' => $popular,
95         'recent' => $recent,
96         'categories' => $categories,
97         'comments' => $comments,
98         'commentForm' => $commentForm
99     ]);
100 }
```

```
public function actionMyPost()
{
    if (!Yii::$app->user->isGuest) {
        $this->layout = 'blog';
        $searchModel = new ArticleSearch();
        $dataProvider = new ActiveDataProvider([
            'query' => Article::find()->creator(Yii::$app->user->id)->latest(),
        ]);
        return $this->render('mypost', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }
    return $this->redirect(['site/login']);
}
```

```
public function actionInfo($id)
{
    $this->layout = "blog";
    return $this->render('info', [
        'model' => $this->findModel($id),
    ]);
}

public function actionUpdate($id)
{
    $this->layout = "blog";
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }

    return $this->render('update', [
        'model' => $model,
    ]);
}
```

```

public function actionBlog()
{
    $this->layout = 'blog';
    $data = Article::getAll(5);

    for ($i = 0, $j = 0; $i < sizeof($data); $i += 2, $j++) {
        $data1[$j] = $data['articles'][$i];
    }
    for ($i = 1, $k = 0; $i < sizeof($data); $i += 2) {
        $data2[$k++] = $data['articles'][$i];
    }

    $popular = Article::Popular();

    $recent = Article::Recent();

    $categories = Category::find()->all();

    return $this->render("blog", [
        'articles' => $data['articles'],
        'articles1' => $data1,
        'articles2' => $data2,
        'pagination' => $data['pagination'],
        'popular' => $popular,
        'recent' => $recent,
        'categories' => $categories
    ]);
}

```

```

public function actionCategory($id)
{
    $data = Category::getArticlesByCategory($id);

    $popular = Article::Popular();

    $recent = Article::Recent();

    $categories = Category::find()->all();

    return $this->render("category", [
        'articles' => $data['articles'],
        'pagination' => $data['pagination'],
        'popular' => $popular,
        'recent' => $recent,
        'categories' => $categories
    ]);
}

public function actionCreate()
{
    if (!Yii::$app->user->isGuest) {
        $this->layout = 'blog';
        $model = new Article();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['info', 'id' => $model->id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }
    return $this->redirect(['site/login']);
}

```

View:

几个比较重要的 view:

- ◎ frontend\views\blog\blog.php => Blog 模块首页 (模版来自 WordPress)
- ◎ frontend\views\blog\create.php => Blog 模块创建文章页面

- ◎ frontend\views\blog\view.php => 文章详情页
- ◎ frontend\views\blog\mypost.php => 管理我的文章页面

II. 后台:

Model: 与前台共用 common 里的 model, 不再介绍。

Controller:

- ◎ backend\controllers\ArticleController.php
- ◎ backend\controllers\CommentController.php

几个 function 示例:

```
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}
public function actionCreate()
{
    $model = new Article();

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }

    return $this->render('create', [
        'model' => $model,
    ]);
}
```

```
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }

    return $this->render('update', [
        'model' => $model,
    ]);
}

public function actionDelete($id)
{
    $this->findModel($id)->delete();

    return $this->redirect(['index']);
}
```

View:

主要由 gii 生成, 并进行了修改, 一些主要的 view:

- ◎ backend\views\article*
- ◎ backend\views\category*
- ◎ backend\views\comment*

首页界面

1.功能介绍

在首页我们提供了数据库中最新博客的概览, 并为用户提供直接跳转到博客详情页的选项。



点击新闻标题跳转到对应新闻详情页,点击更多新闻跳转到我们网站的新闻页面。

2、 代码实现

涉及表: CovNews

Controller: 位于 SiteController.php

```
public function actionIndex()  
{  
    $data=RU_Blog::getAll(6);  
    return $this->render('index',[  
        'news'=>$data['blogs'],  
    ]);  
}
```

Model:

由谢志颖同学实现,这里不再赘述。

View: 位于 index.php

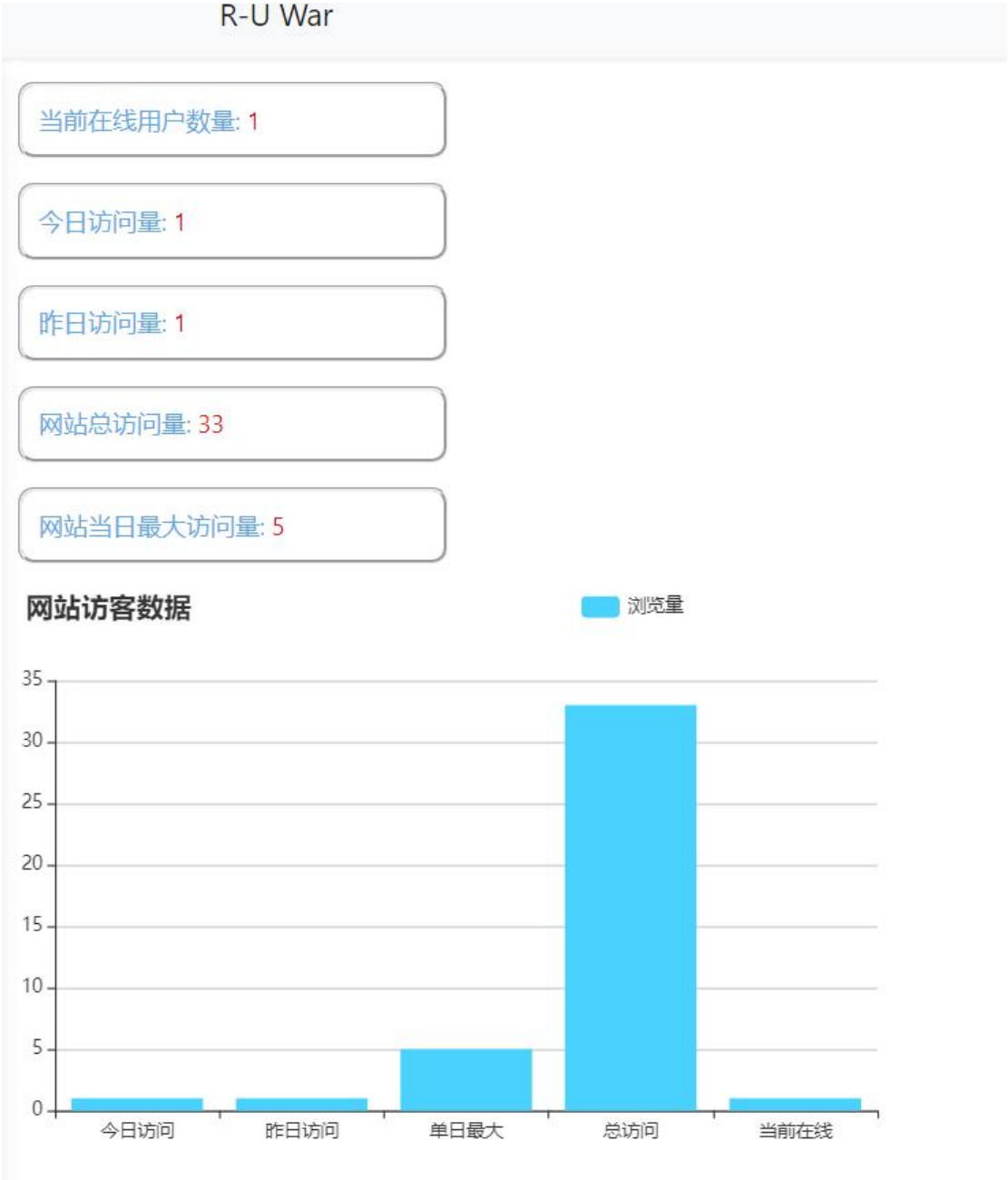
```
<div class="card-carousel">  
    <div class="card" id="1">  
        <div class="sliderThumb" style="background:url(<?= $url ?>kewadi.png) 50% 50% no-repeat; background-size:cover;height:60px;width:100%">  
        </div>  
        <div class="sliderCaption" style="text-align:center;padding:20px 20px 0px">  
            <a href="<?= $news[0]->sourceUrl ?>" style="font-size: 22px;font-weight:600;color:#4a0d66"><?= $news[0]->title?></a>  
            <p style="font-size: 0.8rem;padding-top: 10px;"><?= $news[0]->summary?></p>  
            <a href="<?=Url::toRoute(['news/index']);?>" class="btn btn-secondary">更多新闻</a>  
        </div>  
    </div>  
</div>
```

全站流量统计

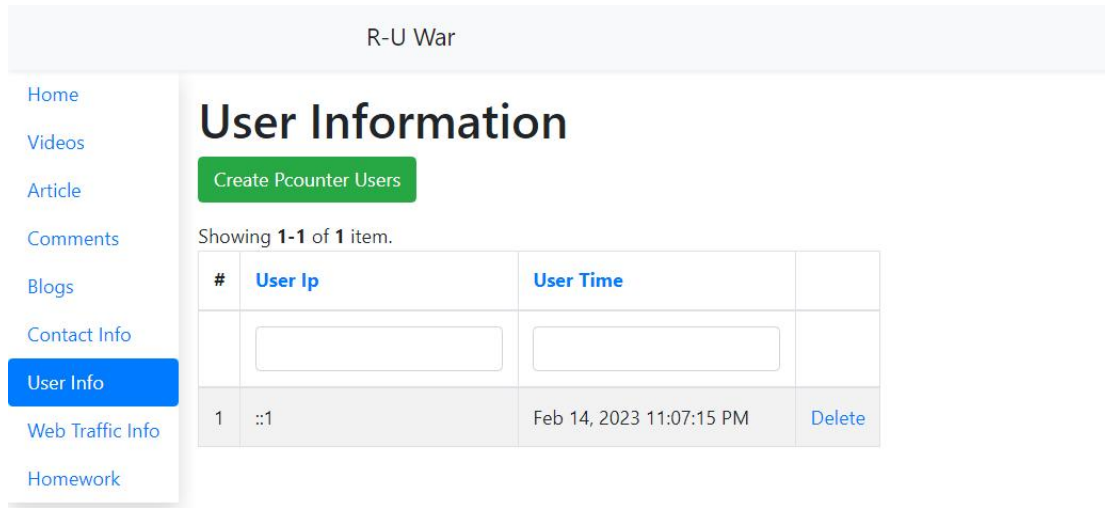
1. 功能介绍

根据网站的用户访问情况,实时更新数据库中相应表信息,并以数据表和动态柱状图的形式显示在后端主页中。管理员同时可以通过对后台中相应模块进行管理和检测,查看当日用户的访问信息和网站总流量的信息,并进行相应的增删改查操作。

后台可视化：



后台管理：


















2. 代码介绍

Models 由 gii 生成，路径：

common\models\PcounterUser.php

common\models\PcounterSave.php

<div><div><div></div><div></div><div></div></div><div>↔ T ↔</div></div>						save_name	save_value	
<input type="checkbox"/>		编辑		复制		删除	counter	25
<input type="checkbox"/>		编辑		复制		删除	day_time	2459012
<input type="checkbox"/>		编辑		复制		删除	max_count	5
<input type="checkbox"/>		编辑		复制		删除	max_time	1591437600
<input type="checkbox"/>		编辑		复制		删除	yesterday	1

+ 选项						
<div><div>←</div><div>T</div><div>→</div></div>				<div>▼</div>	user_ip	user_time
<input type="checkbox"/>		编辑	 复制	 删除	127.0.0.1	1591873523
<input type="checkbox"/>		编辑	 复制	 删除	::1	1591878771

Components:

路径：backend\components\UserCounter.php

此模块的实现利用了 yii 中的组件，主要是实现一些重要的统计函数和输出函数，得以进行用户访问时信息的实时创建和更新数据库相应信息，这是配置文件中增加的组件信息：

```
'userCounter' => [
    'class' => 'backend\components\UserCounter',
    'tableUsers' => 'pcounter_users',
    'tableSave' => 'pcounter_save',
    'autoInstallTables' => true,
    'onlineTime' => 10, // min
],
```

组件中的部分重要函数:

```
protected function checkTables()
{
    if (Yii::$app->db->schema->getTableSchema($this->tableUsers, true) === null) {
        if ($this->autoInstallTables) {
            Yii::$app->db->createCommand()
                ->createTable(
                    $this->tableUsers,
                    array(
                        'user_ip' => 'VARCHAR(255) NOT NULL PRIMARY KEY',
                        'user_time' => 'int(10) unsigned NOT NULL',
                    ))
                ->execute();
        }
    }
    if (Yii::$app->db->schema->getTableSchema($this->tableSave, true) === null) {
        if ($this->autoInstallTables) {
            Yii::$app->db->createCommand()
                ->createTable(
                    $this->tableSave,
                    array(
                        'save_name' => 'VARCHAR(10) NOT NULL PRIMARY KEY',
                        'save_value' => 'int(10) unsigned NOT NULL',
                    ))
                ->execute();

            Yii::$app->db->createCommand()
                ->batchInsert(
                    $this->tableSave,
                    array('save_name', 'save_value'),
                    array(
                        array('day_time', 0),
                        array('counter', 0),
                        array('yesterday', 0),
                        array('max_count', 0),
                        array('max_time', 0),
                    ))
                ->execute();
        }
    }
}
```

```
public function refresh($force = false)
{
    if ($this->alreadyUpdated && !$force) {
        return;
    }
    $this->getCurrentData();
    $today = GregorianCalendar::toJD(date('m'), date('j'), date('Y'));
    $daysSinceLastUpdate = $today - $this->dayTime;
    if ($this->isNewDay()) {
        $lastUpdateTotalUsers = $this->getLastLoggedUsers();
        $this->yesterday = ($daysSinceLastUpdate == 1 ? $lastUpdateTotalUsers : 0);
        $this->update($this->tableSave, array('save_value' => $this->yesterday, 'save_name' => "yesterday"));
        if ($this->isNewMaximum($lastUpdateTotalUsers)) {
            $this->maxCount = $lastUpdateTotalUsers;
            $this->maxDate = mktime(12, 0, 0, date('n'), date('j'), date('Y')) - 86400;
            $this->update($this->tableSave, array('save_value' => $this->maxCount, 'save_name' => "max_count"));
            $this->update($this->tableSave, array('save_value' => $this->maxDate, 'save_name' => "max_time"));
        }
        $this->update($this->tableSave, array('save_value' => $this->total + $lastUpdateTotalUsers, 'save_name' => "counter"));
        $this->update($this->tableSave, array('save_value' => $today, 'save_name' => "day_time"));
        $this->truncate($this->tableUsers);
        $this->total += $lastUpdateTotalUsers;
    }
    $this->insertOrUpdateIpAddress();
    $this->today = $this->getLastLoggedUsers();
    $this->online = $this->getLastLoggedUsers(true);
    $this->total += $this->today;
    if ($this->isNewMaximum($this->today)) {
        $this->maxCount = $this->today;
        $this->maxDate = time();
    }
    $this->alreadyUpdated = true;
}
```

```

public function getTotal()
{
    return $this->total;
}

/**
 * Getter for number of users which are online at the moment.
 * @return int
 */
public function getOnline()
{
    return $this->online;
}

/**
 * Getter for number of users which were online today.
 * @return int
 */
public function getToday()
{
    return $this->today;
}

/**
 * Getter for number of users which were online yesterday.
 * @return int
 */
public function getYesterday()
{
    return $this->yesterday;
}

```

Views: 路径: backend\views\site\index.php

将全站的流量统计数据分别以表格和柱状图的形式展现在后台。表格中数据利用组件中提到的函数实现前端页面的输出。

```

<p>当前在线用户数量: <span><?php echo Yii::$app->userCounter->getOnline(); ?></span></p>
<p>今日访问量: <span><?php echo Yii::$app->userCounter->getToday(); ?></span></p>
<p>昨日访问量: <span><?php echo Yii::$app->userCounter->getYesterday(); ?></span></p>
<p>网站总访问量: <span><?php echo Yii::$app->userCounter->getTotal(); ?></span></p>
<p>网站当日最大访问量: <span><?php echo Yii::$app->userCounter->getMaximal(); ?></span></p>
<p>网站最大访问量日期: <span><?php echo date('Y.m.d', Yii::$app->userCounter->getMaximalTime()); ?></span></p>

```

为了使后台管理者对网站的流量获得及时且直观的反馈,利用 E-charts 的 js 模板进行可视化。动态图表的实现总体思路为:从数据库读取相应数据,经组件中的函数 php 语句进行输出,再把结果以 js 变量的方式导入到 E-charts 模板中,以便实现与数据库中数据的实时更新,借此,每个产生的访问量都会同时显示在动态的柱状图中。

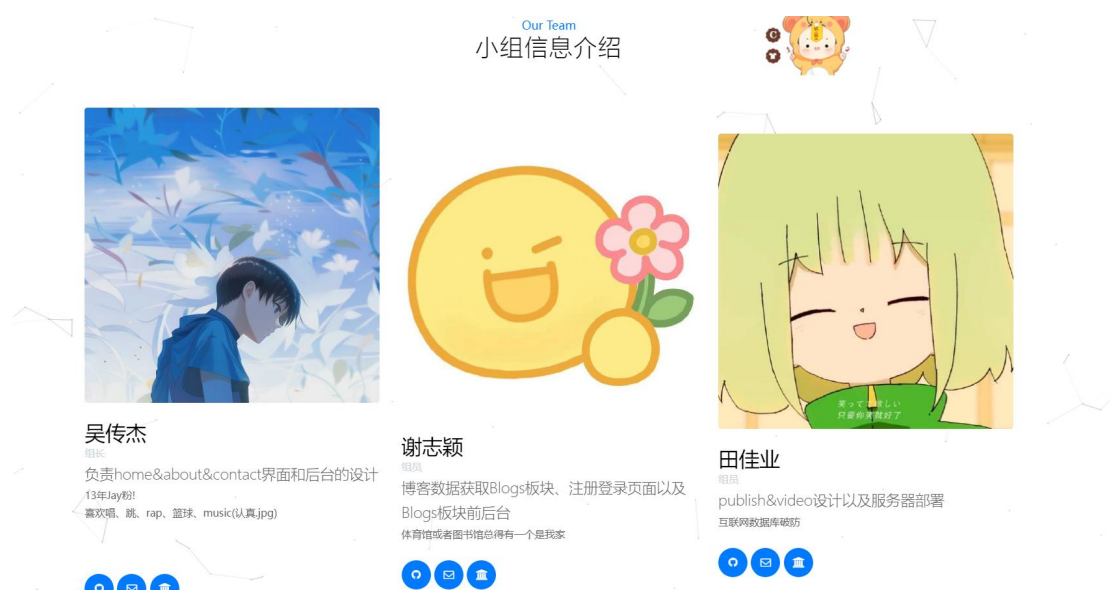
网站介绍及联系页面

1. 功能介绍

本模块所实现的功能是让网站的访客对网站的内容主旨和创作团队有全方位的了解，同时访问者可以对网页中感兴趣的部分通过本页找到其对应的创作者，并以多种方式（Github，邮箱，个人主页）了解创作者。同时，最重要的功能是收集和统计用户对于网站内容及疫情相关的反馈和信息，将数据采集到数据库并访问到后台，以便管理者对收到的信息进行反馈。

contact 页面效果：





代码介绍

Models: 由 gii 生成，路径：
common\models\ContactForm.php
主要记录了用户发送消息的各类信息

testdb contact_form
id : int(20)
firstname : varchar(50)
lastname : varchar(50)
sex : tinyint(1)
wechatid : varchar(50)
phone : varchar(50)
message : varchar(255)

controllers: 由 gii 生成，路径：

backend\controllers\ContactFormController.php

views: 使用了 colorlib 的 scene 模板，并且在各个细节上进行了优化。

路径: frontend\views\site\contact.php

团队作业页面

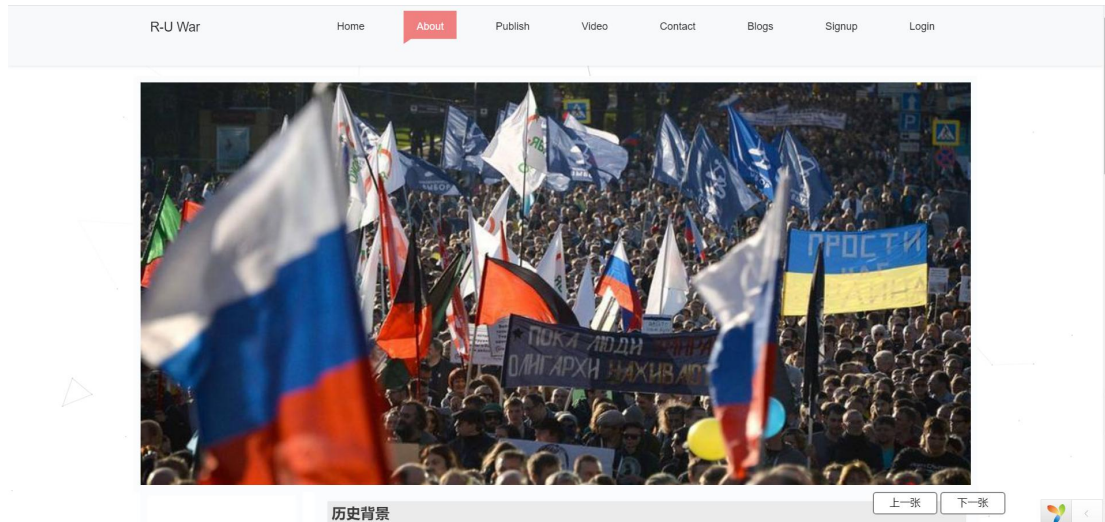
该页面通过对原有模板的精心编写，实现了本团队作业的提交下载页面。为了使得页面更加美观和直观，进行了多次的页面布局修改，同时为了实现分门分类的精细化下载，把下载文件分成全部和单独下载，方便按需取阅。

网页效果：



about 页面的实现

首先在页面的上方添加了一个照片的自动轮播图，点击【上一张】【下一张】可以实现指定跳转。



具体实现代码在 `about.php` 中，以下是部分实现代码（`views/site/about.php`）

```
<body>
  <div class="box">
    
  </div>
  <div>
    <input type="button" class="btn1" value="上一张" />
    <input type="button" class="btn1" value="下一张" />
  </div>
  <script>
    var pic = document.getElementById("pic");
    var preBtn = document.getElementsByClassName("btn1")[0];
    var nextBtn = document.getElementsByClassName("btn1")[1];
    //
    preBtn.onclick = function() {
      n--;
      if (n == 0) {
        n = 6;
      }
      pic.src = "../../frontend/web/img/" + n + ".jpg"
    }
    nextBtn.onclick = function() {
      picLunH();
    }
    var n = 1;

    function picLunH() {
      n++;
      if (n == 7) {
        n = 1;
      }
      pic.src = "../../frontend/web/img/" + n + ".jpg"
    }
    setInterval(picLunH, 3000);
  </script>
```

我的页面有一个随处飘动的线段背景，在鼠标停留的地方会不断累积。

以下是在 `about.php` 中调用的实现代码（源代码在 `web/js/canvas-nest.min.js` 中）

谢志颖

Blogs 模块功能与实现:

通过 python 实现爬虫爬取俄乌战争相关博客存取到数据库表 RU_Blog 中再渲染到页面。

1.爬虫的实现:

“spider\spiderForBlog.py”

通过打开 common/config 下的 main-local 配置文件读取到信息

```
project_dir = os.getcwd()
db_setting_file = open(project_dir + "/common/config/main-local.php")
main_local = db_setting_file.readlines()
db_setting_file.close()
```

再通过对读取到文本的解析得到数据库对应信息

```
for line in main_local:
    if line.find('dsn') >= 0:
        host = line[line.find("mysql:host=") + 11 : line.find(";dbname=")]
        database = line[line.find(";dbname=") + 8 : line.find(",")]
    elif line.find("username") >= 0:
        username = line[line.find("'username' => '") + 15 : line.find(",")]
    elif line.find("password") >= 0:
        password = line[line.find("'password' => '") + 15 : line.find(",")]
```

通过 pymysql 库连接到数据库

```
connect = sql.connect(host=theHost,
                        user=username,passwd=password,db=database)
cur = connect.cursor()
```

通过对指定博客网站的 html 分析对页面进行解析爬取,并执行 sql 语句存到数据库中

```
def crawlBlog(webUrl:str):
    """通过爬虫获取文章内容存储在数据库中
    Args:
        webUrl (str): 网页链接
    """
    global cur
    global counter
    getRequest=requests.get(webUrl,headers=header)
    getRequest.raise_for_status()
    bsobj=bs(getRequest.text,'html.parser',from_encoding=getRequest.encoding)
    content= bsobj.find("div",{"class":"searchListOne"})
    for blog in content.find_all("li"):
        content = blog.find("div")
        source = blog.find("p",{"class":"source"})
        if not content : continue
        title = content.find("a",{"target": "_blank"})
        theSQL="""insert into ru_blog(pubDate,title,infoSource,summary,sourceUrl,image) values(%s,%s,%s,%s,%s,%s)"""
        theData=[source.find("span").get_text(),title.get_text(),source.find("a").get_text(),content.find("p").get_text(),title.get_text(),content.find("img").get_text()]
        cur.execute(theSQL,theData)
        counter+=1
```

存储博客的发布日期、标题、消息模块来源、内容概要、源头网址、博客相关图片等信息。

2.Controller

frontend\controllers\BlogsController.php

frontend\controllers\RU_BlogController.php

```
public function actionIndex()
{
    $this->layout = 'blog';
    $data = RU_Blog::getAll(16);

    $data1 = $data['blogs'][0];
    for ($i = 1, $j = 0; $i < 5;) {
        $data2[$j++] = $data['blogs'][$i++];
    }
    for ($i = 5, $j = 0; $i < 10;) {
        $data3[$j++] = $data['blogs'][$i++];
    }
    for ($i = 10, $j = 0; $i < 16;) {
        $data4[$j++] = $data['blogs'][$i++];
    }
    $videos = Video::getAll(5);
    return $this->render("index", [
```

通过对数据的获取渲染

关键数据获取数据库中对应 ru_blog 表中的 blogs

3.Model

common\models\RU_Blog.php

common\models\RU_BlogQuery.php

common\models\RU_BlogSearch.php

在 RU_Blog.php 中 类 RU_Blog 继承 yii\db\ActiveRecord 类
定义的 rules 方法、attribute 方法对数据进行解析限制

```
public function rules()
{
    return [
        [['image'], 'string'],
        [['pubDate', 'title', 'infoSource', 'sourceUrl'], 'string', 'max' => 255],
        [['summary'], 'string', 'max' => 2000],
    ];
}
```

getAll 方法根据页面限制对最新的博客进行获取

```

public static function getAll($pageSize = 5)
{
    $query = RU_Blog::find()->latest();

    $count = $query->count();

    $pagination = new Pagination(['totalCount' => $count, 'pageSize'=>$pageSize])

    $blogs = $query->offset($pagination->offset)
        ->limit($pagination->limit)
        ->all();

    $data['blogs'] = $blogs;
    $data['pagination'] = $pagination;

    return $data;
}

```

在 RU_BlogQuery.php 中 类 RU_Blog 继承 yii\db\ActiveQuery 类
定义 all 方法和 one 方法 以及 latest 方法，指定由发布时间排序

```

public function latest(){
    return $this->orderBy(['pubDate'=>SORT_DESC]);
}

```

4.View

frontend\views\blogs\index.php

通过 controller 传递 的数据 进行 页面 渲染

```

<div class="col-md-6">
    <div class="row">
        <?php foreach ($news2 as $news) : ?>
            <div class="col-md-6 col-6 paddding animate-box" data-animat
                <div class="fh5co_suceefh5co_height_2"><?php echo 'img
                    <div class="fh5co_suceefh5co_height_position_absolut
                        <div class="fh5co_suceefh5co_height_position_absolut
                            <div class=""><a href="#" class="color_fff"> <i
                                <div class=""><a href="?= $news->sourceUrl ?>"
                                    </div>
                                </div>
                            </div>
                        </div>
                    </div>
                <?php endforeach; ?>
            </div>
        </div>
    </div>
</div>

```


登录、注册页面的实现

1 页面设置

frontend\views\site\signup.php

frontend\views\site\login.php

在原 yii2 模版的基础上将登陆表格移动到屏幕中间，并且修改背景颜色、按钮、添加标题等操作。

部分实现：

```
<div class="site-signup">
    <style>
        .site-signup {
            height: 450px;
            width: 100%;
            text-align: center;
            background-position: center;
            background-size: 50%;
            position: relative;
        }
    </style>

    <p>Signup</p>
    <div class="row">
        <div class="col-lg-12">
            <?php $form = ActiveForm::begin(['id' => 'form-signup']); ?>

            <?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>

            <?= $form->field($model, 'email') ?>
        </div>
    </div>
</div>
```

```

use yii\helpers\Html;
use yii\bootstrap4\ActiveForm;

$this->title = 'Login';
$this->params['breadcrumbs'][] = $this->title;
?>
<?php echo Html::cssFile('@web/css/style.css'); ?>

<div class="site-login">
    <style>
        .site-login {
            height: 450px;
            width: 100%;
            background-position: center;
            background-size: 50%;
            text-align: center;
            position: center;
        }
    </style>

    <p>Login</p>

```

2 背景实现

以文件中俄乌战争相关图片作为背景设置样式存放于
frontend\web\css\background.css

田佳业

2 Video 模块

1、功能介绍

对于浏览型用户：

- ① 浏览由其他用户发表的与俄乌战争有关的视频。
- ② 根据关键词搜索相关的视频。
- ③ 对某个视频进行评论（需要注册登录网站）。
- ④ 点赞\不喜欢 某个视频。
- ⑤ 订阅某个用户。

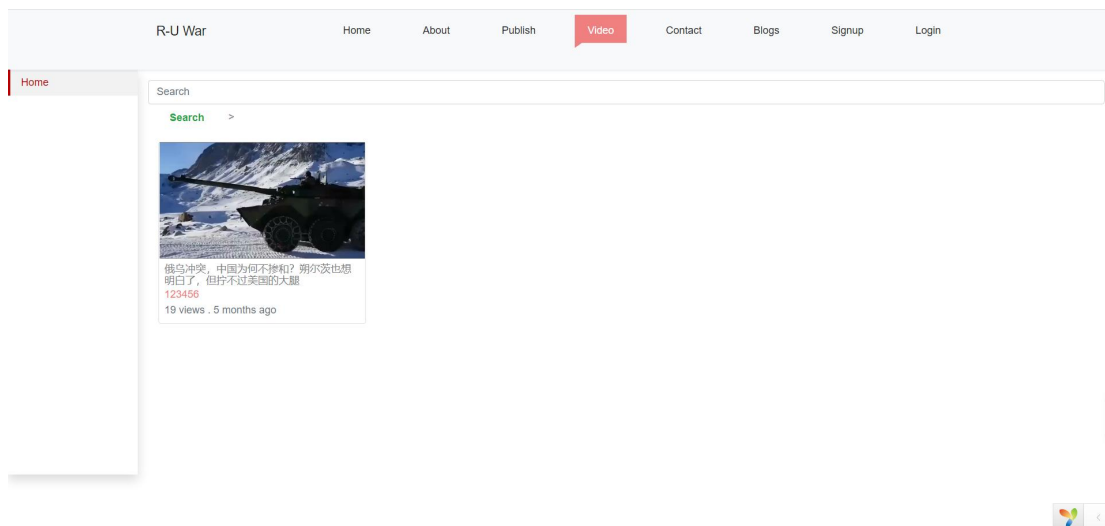
对于创作型用户：

- ④ 发表与俄乌战争相关的视频（需要注册登录网站）。
- ⑤ 上传对应的封面图，选择相应的类别。
- ⑥ 管理用户自己的视频，可以进行更新、删除。

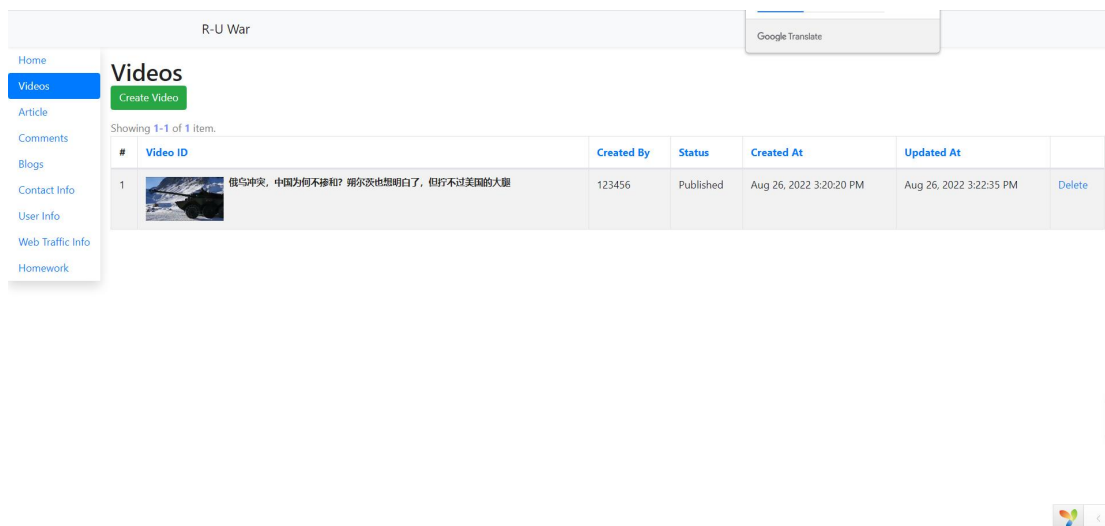
对于网站管理者：

- ④ 审核用户上传的视频内容，决定是否发表。
- ⑤ 对所有用户上传的视频有更新、删除的权限。
- ⑥ 审核用户发表的评论。

前台：



后台：



2、代码实现

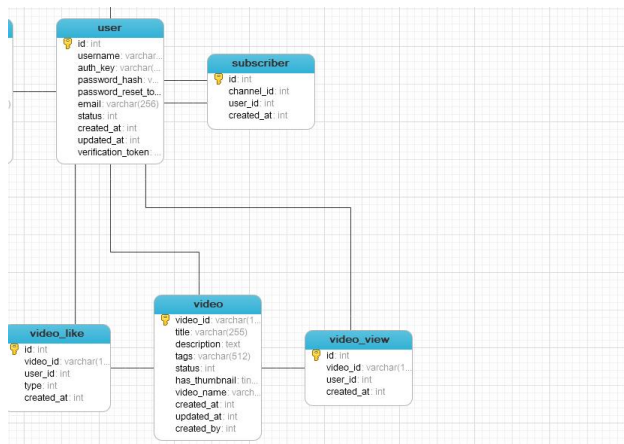
◎ 注：视频的 **comment** 与 **blog** 使用同样的模块，故不再介绍。

前台：

Model:

Model 主要由 gii 生成，几个重要的 model:

- ◎ common\models\Video.php => 对应表 video
- ◎ common\models\VideoLike.php => 对应表 video_like
- ◎ common\models\VideoView.php => 对应表 video_view
- ◎ common\models\Subscriber.php => 对应表 subscriber



Controller:

© frontend\controllers\VideoController.php

几个 function 示例:

```

public function actionCreate()
{
    $this->layout='video';
    $model = new Video();

    $model->video = UploadedFile::getInstanceByName('video');

    if (Yii::$app->request->isPost && $model->save()) {
        return $this->redirect(['update', 'id' => $model->video_id]);
    }

    return $this->render('create', [
        'model' => $model,
    ]);
}
  
```

```

public function actionSearch($keyword){
    $this->layout = 'video';
    $query = Video::find()
        ->with('createdBy')
        ->published()
        ->latest();
    if ($keyword) {
        $query->byKeyword($keyword);
    }
    $dataProvider = new ActiveDataProvider([
        'query' => $query
    ]);

    return $this->render('search', [
        'dataProvider' => $dataProvider
    ]);
}

public function actionIndex(){
    $this->layout='video';
    $dataProvider=new ActiveDataProvider([
        'query'=>Video::find()->published()->latest(),
        'pagination'=>[
            'pageSize'=>6
        ]
    ]);
}
  
```

View:

几个重要的 view:

© frontend\views\video\create.php => 上传视频页面

© frontend\views\video\view.php => 视频观看页面

© frontend\views\video\index.php => 视频模块首页

© frontend\views\video\myvideo.php => 我的视频页面

II. 后台:

Model: 与前台共用 common 里的 model, 故不再描述

Controller:

© backend\controllers\VideoController.php

几个 function 示例:

```
public function actionView($id)
{
    return $this->render('view', [
        'model' => $this->findModel($id),
    ]);
}

/**
 * @return mixed
 */
public function actionCreate()
{
    $model = new Video();

    $model->video = UploadedFile::getInstanceByName('video');

    if (Yii::$app->request->isPost && $model->save()) {
        return $this->redirect(['update', 'id' => $model->video_id]);
    }

    return $this->render('create', [
        'model' => $model,
    ]);
}

/**
 * @param string $id
 * @return mixed
 * @throws NotFoundHttpException
 */
public function actionUpdate($id)
{
    $model = $this->findModel($id);

    $model->thumbnail = UploadedFile::getInstanceByName('thumbnail');
    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['update', 'id' => $model->video_id]);
    }

    return $this->render('update', [
        'model' => $model,
    ]);
}
```

View:

主要由 gii 生成, 并进行了修改, 一些主要的 view:

© backend\views\video*

© backend\views>tag*

在本次实验中, 我购买了阿里云服务器, 并针对云端环境进行了代码修改, 编写一键部署脚本并进行部署。

下面将详细介绍实现的过程:

服务器环境搭建

服务器购买不再赘述。我们团队通过使用 **Vscode** 配置 **ssh** 访问的方式实现了服务器端的协作开发。

服务器环境的搭建可以详细查看部署文档。由于 **composer** 受网络环境影响不稳定，为保证部署的顺利进行，我们只好将 **compsoser** 依赖对应的 **vender** 文件夹添加到了 **git**。

云端环境修改

在云端需要对 **Apache** 环境的路由进行配置，可以在 `/etc/httpd/conf/httpd.conf` 中进行默认路由的配置，并通过 **htaccess** 配置路由规则。

对于云端，采用美化的 **URL** 并不适配 **Apache**，因此云端路径还原为默认形式，使用 `use yii\helpers\Url;` 中的 `Url::to` 可实现若干 **Url** 相关的页面和资源访问方法。

一键部署脚本

这一部分花费时间较多，也是通过各种尝试最终将所有环境配置和项目配置跑通，可以详细查看部署文档和一键部署演示视频。