Создание блога на YII2

В начале нужно спланировать структуру сайта, а именно как будет выглядеть его база данных.

1. **Описание**. Это сайт, на котором пользователи могут читать статьи. Пользователи оставляют комментарии. Возможность авторизации и регистрации. Авторизоваться через VK. Есть разделение статей по категориям и тегам. Теги вывести самому в верстке сайта.
2. **Сущности.** User, Comment, Article, Category, Tag.
3. **Связи.** У user может быть множество comments, однако у comment может быть только один пользователь. Article принадлежит одной category, имеет множество tag и принадлежит только одному user. Category имеет множество article, как и tag.
4. **Описание сущностей.** Свойства и действия объектов.

User

|  |  |
| --- | --- |
| Свойства | Действия |
| id | signup() |
| name | login() |
| email | signupFromVK() |
| password |  |
| photo |  |
| isAdmin |  |

Category

|  |  |
| --- | --- |
| id | getArticles() |
| title | getArticlesCount() |

Article

|  |  |
| --- | --- |
| id, | getTags() |
| title | getContents() |
| description | getCategory() |
| content | getAuthor() |
| image | работа с картинкой (загрузить) |
| viewed | saveCategory() |
| user\_id | getCategoryTitle() |
| category\_id | getDate() |
|  | getPopularArticles() |
|  | getRecentArtiles() |
|  | viewedCounter() |
|  | saveTags() |
|  | getSavedTags() |

Comment

|  |  |
| --- | --- |
| id | getArticcle() |
| text | getAuthor() |
| user\_id | allow() |
| article\_id | dissalow() |
| status | isAllowed() |
|  | getDate() |

1. **Описание конкретных действий.**

1. Установить Фреймворк.

2. Создать бд и таблицы.

3. Модуль, GII модели, CRUD для категорий, статей и тегов.

4. Работа со статьями, создание статьи.

5. Работа со статьями. Загрузка картинки, установка категорий и тегов.

6. Обзор и разбор верстки.

7. Создание и подключение шаблона

8. Регистрация

9. Сокрытие админки

10. Вывод информации для главной страницы

11. Вывод страницы категорий

12. Вывод статьи

13. Добавление комментариев

14. Тестирование и завершение

Затем установить YII2 на Open Server с помощью Composer

Настроить подключение к бд в db.php

composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-basic basic

Настроить отображение ссылок в .htaccess и web/.htaccess

Затем, с помощью миграций, создать таблицы командами

php yii migrate/create create\_article\_table

php yii migrate/create create\_category \_table

php yii migrate/create create\_tag\_table

php yii migrate/create create\_user\_table

php yii migrate/create create\_comment\_table

php yii migrate/create create\_article\_tag\_table

В папке migrations появятся файлы для миграции, которые нужно отредактировать, добавив нужные поля. Затем в консоли нужно выполнить команду:

php yii migrate

Работа с CRUD

Перейти по адресу <http://exaple.com/gii>

Перейти в **Module generator**

Module Class: app\modules\admin\Module

Module id: admin

Скопировать в web.php:

‘modules’ => [  
 ‘admin’ => [

‘class’ => ‘app/modules/admin/Module’,

],

],

Перейти в **Nodule generator**

В **Table Name** поставить \*, убрать галку напротив models\Migration.php

В **CRUD** Generator пишем:

**Model Class**: app\models\Article

**Search Model Class**: app\models\ArticleSearch

**Controller Class**: app\models\ArticleController

**View Path**: @app/modules/admin/views/article

В **Table Name** поставить \*, убрать галку напротив models\Migration.php

В **CRUD** Generator пишем:

**Model Class**: app\models\Category

**Search Model Class**: app\models\CategorySearch

**Controller Class**: app\models\CategoryController

**View Path**: @app/modules/admin/views/сategory

В **Table Name** поставить \*, убрать галку напротив models\Migration.php

В **CRUD** Generator пишем:

**Model Class**: app\models\Tag

**Search Model Class**: app\models\TagSearch

**Controller Class**: app\models\TagController

**View Path**: @app/modules/admin/views/tag

Далее, нужно сделать шаблон в modules\admin\Module.php и переписываем свойство layout

public $layout = '/admin'

Перейти в файл views\layout\main.php, скопировать его содержимое в views\layout\admin.php

И прописать ссылки для меню

['label' => 'Home', 'url' => ['/admin/default/index']],  
['label' => 'Article', 'url' => ['/admin/article/index']],  
['label' => 'Categories', 'url' => ['/admin/category/index']],  
['label' => 'Tag', 'url' => ['/admin/tag/index']]

Теперь нужно отредактировать вывод статьи в админке.

Переходим в modules/admin/views/article/\_form.php

В этом файле расположена верстка страницы статей. В ней можно убрать все ненужные поля, сменить внешний вид и так далее.

<?php  
  
use yii\helpers\Html;  
use yii\widgets\ActiveForm;  
  
*/\** ***@var*** *$this yii\web\View \*/  
/\** ***@var*** *$model app\models\Article \*/  
/\** ***@var*** *$form yii\widgets\ActiveForm \*/*?>  
  
<div class="article-form">  
  
 <?php $form = ActiveForm::*begin*(); ?>  
  
 <?= $form->field($model, 'title')->textInput(['maxlength' => true]) ?>  
  
 <?= $form->field($model, 'description')->textarea(['rows' => 6]) ?>  
  
 <?= $form->field($model, 'content')->textarea(['rows' => 6]) ?>  
  
 <?= $form->field($model, 'date')->textInput() ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Save', ['class' => 'btn btn-success']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

Далее прописывается валидация полей при создании статьи в файле models/Arcticle.php

public function rules()  
{  
 return [  
 [['title'], 'required'],  
 [['title', 'description', 'content'], 'string'],  
 [['date'], 'date', 'format' => 'php:Y-m-d'],  
 [['date'], 'default', 'value' => date('Y-m-d')],  
 [['title'], 'string', 'max' => 255],  
 [['description'], 'required'],  
 [['content'], 'required']  
 ];  
}

Из формы информация поступает в экшен modules/admin/controllers/ArcticleController.php

Теперь о создании картинки.

Первое, что нужно сделать, это создать кнопку для загрузки картинки в modules/admin/article/view.php . Затем нужно создать форму для кнопки, которая загружает картинку в modules/admin/article/image.php, форму можно скопировать из \_form.php.

<!--Форма загрузки картинки-->  
  
<div class="article-form">  
  
 <?php $form = ActiveForm::*begin*(); ?>  
  
 <?= $form->field($model, 'image')->fileInput(['maxlength' => true]) ?>  
  
 <div class="form-group">  
 <?= Html::*submitButton*('Submit', ['class' => 'btn btn-success']) ?>  
 </div>  
  
 <?php ActiveForm::*end*(); ?>  
  
</div>

Затем создать отдельную модель models/ImageUpload. Данная модель отвечает за валидацию картинки по расширению и формирование ее названия, а также за загрузку файла в папку и внесение его названия в базу данных и его удаление. Короче за логику работы загрузки картинки.

В экшене modules/admin/controllers/ArcticleController.php прописываем – что будет происходить после нажатия кнопки submit (отправить), в какие части фреймворка отправлять команды.

В модели картинки models/ImageUpload пишем логику обработки картинки и процесс ее загрузки на сервер.

В модели статьи modules/Article.php создаются команды для базы данных CRUD – так название картинки попадает в базу и удаляется из нее. А также, если картинка удалится из базы данных, то она удалится и из папки uploads.

В modules/admin/article/index.php добавляем изображение по умолчанию, когда картинка еще не загружена.

/\*выводится картинка "image not found"\*/  
[  
 'format' => 'html',  
 'label' => 'image',  
 'value' => function($data) {  
 return Html::*img*($data->getImage(), ['width' => 200]);  
 }  
],