**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 3

по дисциплине: Технологии Web-программирования

тема: «**Серверное программирование. Yii2**»

Выполнил: ст. группы ПВ-41

Таран Алексей Николаевич

Проверил: Картамышев С.В.

Белгород 2020

**Задание к лабораторной работе:**

1. Развернуть базовое приложение Yii2 App Basic.
2. Настроить конфигурацию работы приложения с docker.
3. Добавить модуль для работы с API.
4. Добавить несколько контроллеров со статическими данными.
5. Продемонстрировать работу API в Postman.

Ход работы

Было развернуто базовое приложение PHP YII 2 при помощи composer.

Table

Description automatically generated

Затем, был собран docker-контейнер при помощи docker-compose

Конфигурация docker\_compose.yml:

version: '3.1'

services:

php:

image: yiisoftware/yii2-php:7.3-apache

volumes:

- ~/.composer-docker/cache:/root/.composer/cache:delegated

- ./:/app:delegated

ports:

- '1199:80'

extra\_hosts:

- "visiticelandbeauty.com:127.0.0.1"

mysql:

image: mysql

command: --default-authentication-plugin=mysql\_native\_password --character-set-server=utf8 --collation-server=utf8\_general\_ci

environment:

- MYSQL\_ROOT\_PASSWORD=verysecret

- MYSQL\_DATABASE=bstu

- MYSQL\_USER=bstu

- MYSQL\_PASSWORD=bstu321!

Запуск докер-контейнера:

docker-compose up -d

В конфигурации config/web.php добавлен модуль для поддержки версий будущего API

<?php

$params = require \_\_DIR\_\_ . '/params.php';

$db = require \_\_DIR\_\_ . '/db.php';

$config = [

'id' => 'basic',

'language' => 'ru-RU',

'basePath' => dirname(\_\_DIR\_\_),

'bootstrap' => ['log'],

'aliases' => [

'@bower' => '@vendor/bower-asset',

'@npm' => '@vendor/npm-asset',

],

'components' => [

'request' => [

// !!! insert a secret key in the following (if it is empty) - this is required by cookie validation

'cookieValidationKey' => 'pg96W4\_4-kjLBw4ZcpX2jBAej8YyfJp7',

'parsers' => [

'application/json' => 'yii\web\JsonParser'

]

],

'cache' => [

'class' => 'yii\caching\FileCache',

],

'user' => [

'identityClass' => 'app\models\User',

'enableAutoLogin' => true,

],

'errorHandler' => [

'errorAction' => 'site/error',

],

'mailer' => [

'class' => 'yii\swiftmailer\Mailer',

// send all mails to a file by default. You have to set

// 'useFileTransport' to false and configure a transport

// for the mailer to send real emails.

'useFileTransport' => true,

],

'log' => [

'traceLevel' => YII\_DEBUG ? 3 : 0,

'targets' => [

[

'class' => 'yii\log\FileTarget',

'levels' => ['error', 'warning'],

],

],

],

'db' => $db,

'urlManager' => [

'enablePrettyUrl' => true,

'showScriptName' => false,

'rules' => [

'api/<module:\w+>/<controller:\w+>/<action:\w+>' => '<module>/<controller>/<action>'

],

],

],

'params' => $params,

'modules' => [

'v1' => [

'class' => 'app\modules\v1\v1'

]

]

];

if (YII\_ENV\_DEV) {

// configuration adjustments for 'dev' environment

$config['bootstrap'][] = 'debug';

$config['modules']['debug'] = [

'class' => 'yii\debug\Module',

// uncomment the following to add your IP if you are not connecting from localhost.

'allowedIPs' => ['127.0.0.1', '::1', '\*'],

];

$config['bootstrap'][] = 'gii';

$config['modules']['gii'] = [

'class' => 'yii\gii\Module',

// uncomment the following to add your IP if you are not connecting from localhost.

'allowedIPs' => ['127.0.0.1', '::1', '\*'],

];

}

return $config;

Добавлен API\_Controller:

<?php

namespace app\modules\v1\controllers;

use yii\filters\auth\CompositeAuth;

use yii\filters\ContentNegotiator;

use yii\filters\Cors;

use yii\filters\RateLimiter;

use yii\filters\VerbFilter;

use yii\rest\Controller;

use yii\web\Response;

class ApiController extends Controller

{

public function behaviors()

{

return [

'contentNegotiator' => [

'class' => ContentNegotiator::className(),

'formats' => [

'application/json' => Response::FORMAT\_JSON,

],

],

'verbFilter' => [

'class' => VerbFilter::className(),

'actions' => $this->verbs(),

],

'authenticator' => [

'class' => CompositeAuth::className(),

],

'rateLimiter' => [

'class' => RateLimiter::className(),

],

'corsFilter' => [

'class' => Cors::class

]

];

}

}

Наследуясь от данного контроллера создаем остальные контроллеры проекта:

Например, в качестве примера контроллера приведем контроллер главной страницы:

<?php

namespace app\modules\v1\controllers;

use app\modules\v1\models\MainPage;

use app\modules\v1\models\Chapter;

use app\modules\v1\models\Note;

use app\modules\v1\models\PagePhoto;

use app\modules\v1\models\Video;

class MainpageController extends ApiController{

/\*\*

\* Пример запроса:

\* http://127.0.0.1:1199/api/v1/mainpage/names

\*/

public function actionNames(){

$mainPageObject = new MainPage();

return $mainPageObject -> dataTable();

}

/\*\*

\* Пример запроса:

\* http://127.0.0.1:1199/api/v1/mainpage/chapter?id=0

\*/

public function actionChapter($id){

/\*

Замена на выборку из БД

(Раздел главной страницы)

Фильтр по названию раздела страницы из id

\*/

$chapterObject = new Chapter();

return $chapterObject->dataTable()[10];

}

/\*\*

\* Пример запроса:

\* http://127.0.0.1:1199/api/v1/mainpage/note?id=0

\*/

public function actionNote($id){

/\*

Замена на выборку из БД

(Заметка для главной страницы)

Фильтр по названию заметки из id

\*/

$noteObject = new Note();

return $noteObject->dataTable()[8];

}

/\*\*

\* Пример запроса:

\* http://127.0.0.1:1199/api/v1/mainpage/video

\*/

public function actionVideo(){

/\*

Замена на выборку из БД

(Видео для главной страницы)

\*/

$videoObject = new Video();

return $videoObject->dataTable();

}

}

И в качестве модели, из которой он получает данные приведем mainPage.php

<?php

namespace app\modules\v1\models;

use yii\base\Model;

class MainPage extends Model {

public function rules(){

return [

[['id'], 'required', 'number'],

[['name'], 'required', 'string'],

];

}

public function dataTable(){

return [

['id' => 0, 'name' => "Столица"],

['id' => 1, 'name' => "Что посмотреть"],

['id' => 2, 'name' => "Кухня"],

['id' => 3, 'name' => "Как добраться"],

['id' => 4, 'name' => "Где жить"],

['id' => 5, 'name' => "На заметку"],

];

}

}

Аналогично создаются модели и контроллеры для остальных сущностей будущей БД.

Тестирование работы API в POSTMAN:

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email, Teams

Description automatically generated