АНОТАЦІЯ

Документ 1116130.00934-01 12 01 «Навчальна система для планових та факультативних занять кафедри іноземних мов. Текст програми» входить до складу документації програмного комплексу.

У даному документі представлений опис програми, написаної на мові PHP з використанням JavaScript, HTML, CSS та MySQL. Для функціонування клієнтської частини програмного продукту вимагається наявність ЕОМ, що задовольняє нормальну роботу ОС Windows Vista/7/8, Mac OS або Linux з підтримкою роботи браузерів Internet Explorer 8.0 та выще, Mozilla FireFox 3.6 та выще, Google Chrome, Opera, Safari з доступом в інтернет.

Зміст

[1 Інформація про файли програми 4](#_Toc453890331)

[2 Текст програми 6](#_Toc453890332)

[2.1 Файл index.php 6](#_Toc453890333)

[2.2 Файл Autoload.php 6](#_Toc453890335)

[2.3 Файл Db.php 6](#_Toc453890336)

[2.4 Файл Router.php 7](#_Toc453890337)

[2.6 Файл ElectiveController 10](#_Toc453890338)

[2.7 Файл ExerciseController 12](#_Toc453890339)

[2.9 Файл LibraryController.php 15](#_Toc453890340)

[2.10 Файл MainController.php 16](#_Toc453890341)

[2.11 Файл PuzzleController.php 16](#_Toc453890342)

[2.12 Файл TestController.php 17](#_Toc453890343)

[2.13 Файл UserController.php 18](#_Toc453890344)

[2.14 Файл Book.php 21](#_Toc453890345)

[2.15 Файл Comments.php 24](#_Toc453890346)

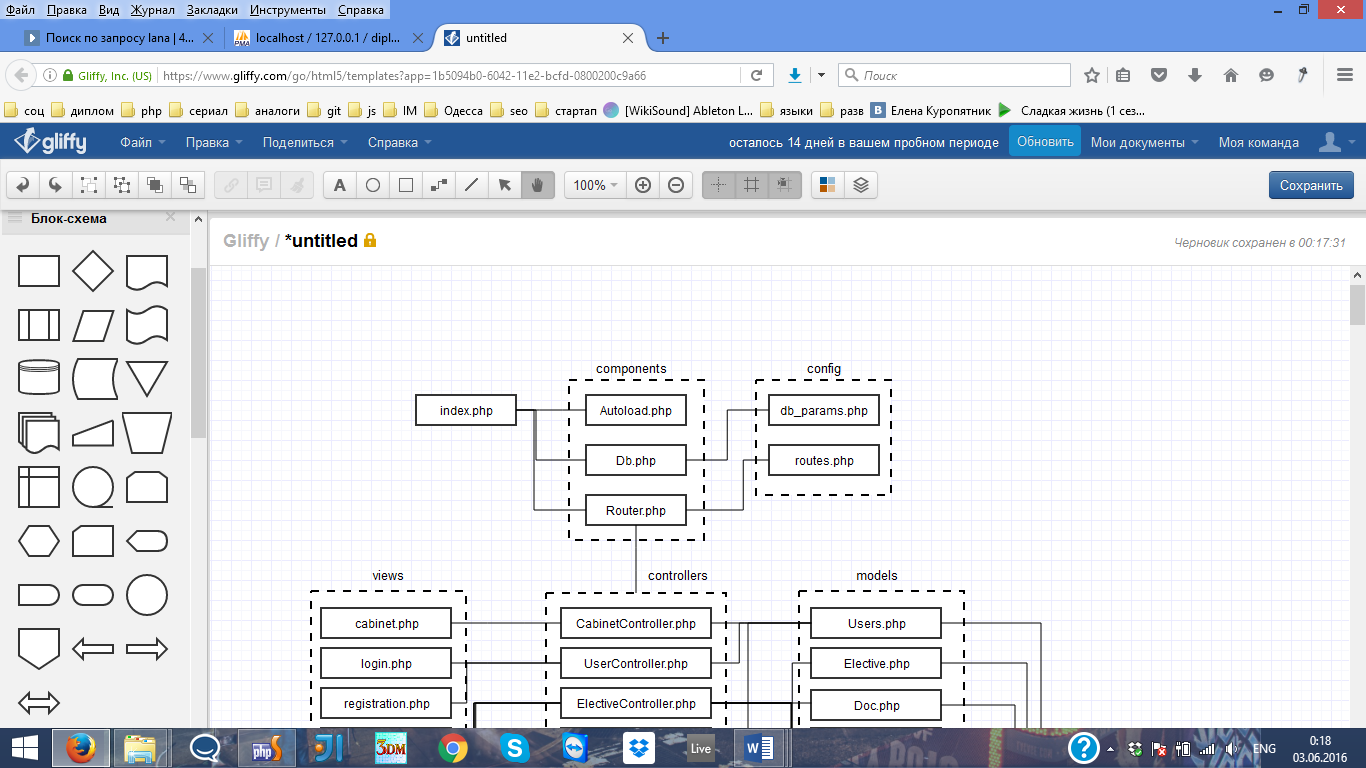
[2.16 Файл DescOfSiteSection.php 27](#_Toc453890347)

[2.17 Файл Elective.php 29](#_Toc453890348)

[2.18 Файл Folders.php 32](#_Toc453890349)

# 1 ІНФОРМАЦІЯ ПРО ФАЙЛИ ПРОГРАМИ

Основні файли, з яких складається програмний комплекс, приведено на діаграмі компонентів (рис. 1.1).



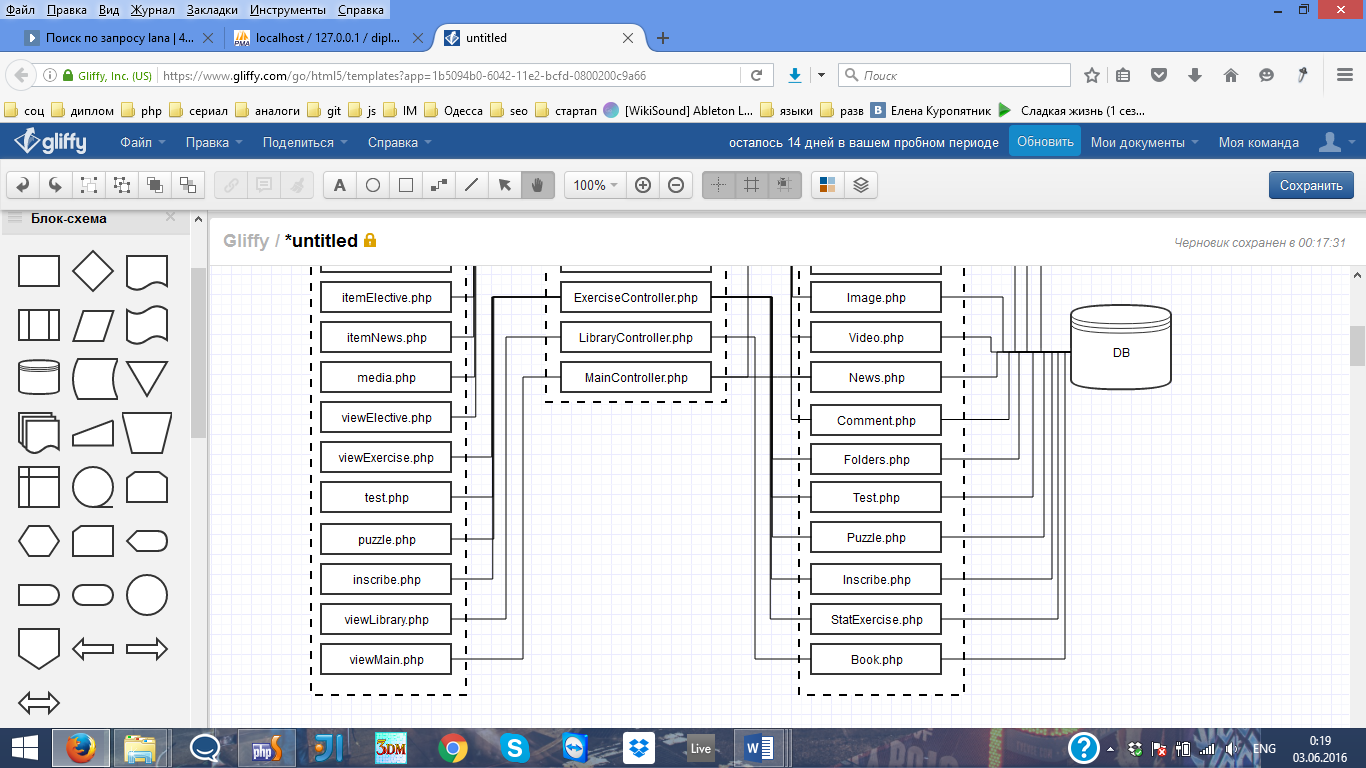


Рисунок 1.1 – Діаграма компонентів системи

Представлення сайту складається з наступних файлів:

* DB – база даних програмного комплексу;
* index.php – скрипт ініціалізації web-додатку;
* Db.php – файл ініцілізації бази даних;
* Router.php – файл маршрутизації запиту;
* db\_params.php – файл параметрів підключення до бази даних;
* routes.php – файл маршрутів запитів;
* Users.php – файл моделі даних користувача;
* Elective.php – файл моделі даних факультативів;
* Doc.php – файл моделі даних документів;
* Image.php – файл моделі даних зображень;
* Video.php – файл моделі даних відео;
* News.php – файл моделі даних новин;
* Comment.php – файл моделі даних коментарів;
* Folders.php – файл моделі даних папок завдань;
* Test.php – файл моделі даних завдання «тести»;
* Puzzle.php – файл моделі даних завдання «пазли»;
* Inscribe.php – файл моделі даних завдання «вставне слово»;
* StatExercise.php – файл моделі даних статистики завдань;
* Book.php – файл моделі даних книг;
* CabinetController.php – файл контролера кабінету;
* UserController.php – файл контролера користувача;
* ElectiveController.php – файл контролера факультативів;
* ExerciseController.php – файл контролера завдань;
* LibraryController.php – файл контролера бібліотеки;
* MainController.php – файл контролера головної сторінки;
* cabinet.php – файл зовнішнього виду сторінки кабінету;
* login.php – файл зовнішнього виду сторінки кабінету;
* registration.php – файл зовнішнього виду сторінки реєстрації;
* itemElective.php – файл зовнішнього виду сторінки факультативу;
* itemNews.php – файл зовнішнього виду сторінки новини факультативу;
* media.php – файл зовнішнього виду сторінки медіа галереї новини;
* viewElective.php – файл зовнішнього виду сторінки;
* viewExercise.php – файл зовнішнього виду сторінки факультативів;
* test.php – файл зовнішнього виду сторінки завдання «тест»;
* puzzle.php – файл зовнішнього виду сторінки завдання «пазл»;
* inscribe.php – файл зовнішнього виду сторінки завдання «вставне слово»;
* viewLibrary.php – файл зовнішнього виду сторінки бібліотеки.

# ТЕКСТ ПРОГРАМИ

## Файл index.php

<?php  
 //FRONT CONTROLLER  
 //1. Общие настройки  
 ini\_set('display\_errors', 1);  
 error\_reporting(E\_ALL);  
  
 session\_start();  
 //2. Подключение файлов системы  
 define('ROOT', dirname(\_\_FILE\_\_));  
 require\_once(ROOT.'/components/Autoload.php');

//3. Установка соединения с БД  
 //4. Вызов Router  
 $router = new Router();  
 $router->run();



## Файл Autoload.php

<?php  
function \_\_autoload($class\_name) {  
 $array\_paths = array(  
 '/models/',  
 '/components/'  
 );  
 foreach ($array\_paths as $path) {  
 $path = ROOT . $path . $class\_name . '.php';  
 if(is\_file($path)) {  
 include\_once $path;  
 }  
 }  
}

## Файл Db.php

<?php  
  
class Db  
{  
 public static function getConnection(){  
 $paramsPath = ROOT.'/config/db\_params.php';  
 $params = include($paramsPath);

$dsn = "mysql::host={$params['host']};dbname={$params['dbname']}";  
 $db = new PDO($dsn, $params['user'], $params['password']);  
 return $db;  
 }  
}

2.4 Файл Router.php  
class Router  
{  
 private $routes;  
 public function \_\_construct()  
 {  
 $routesPath = *ROOT*.'/config/routes.php';  
 $this->routes = include($routesPath);  
 }  
 private function getURI()  
 {  
 if(!empty($\_SERVER['REQUEST\_URI'])){  
 return *trim*($\_SERVER['REQUEST\_URI'],'/');  
 }  
 }  
 public function run()  
 {  
 *// получить строку запроса* $uri = $this->getURI();  
  
 *// проверить наличие такого запроса в routes.php* foreach ($this->routes as $uriPattern => $path) {  
 *// сравниваем $uriPattern и $uri* if(*preg\_match*("~$uriPattern~", $uri)) {  
  
 */\*  
 echo '<br>Где ищем запрос, который набрал пользователь): '.$uri;  
 echo '<br>Что ищем (совпадение из правил): '.$uriPattern;  
 echo '<br>Кто обрабатывает: '.$path;  
 \*/  
 // получаем внутренний путь из внешнего согласно правилу.* $internalRoute = *preg\_replace*("~$uriPattern~", $path, $uri);  
 *//echo '<br><br>Нужно сформировать: '.$internalRoute;  
 //определить какой контроллер  
 // и action обрабатывают запрос, и (параметры)* $segments = *explode*('/', $internalRoute);  
  
 $controllerName = *array\_shift*($segments).'Controller';  
 $controllerName = *ucfirst*($controllerName);  
  
 *//$actionName = 'action'.ucfirst(array\_shift($segments));* $className = *ucfirst*(*array\_shift*($segments)).'Controller';  
 */\*  
 echo '<br>controller name: '.$controllerName;  
 echo '<br>action name: '.$actionName;  
 \*/* $actionName = 'action'.*ucfirst*(*array\_shift*($segments));  
 $parameters = $segments;  
 */\*  
 echo '<pre>';  
 print\_r($parameters);  
 die;  
 \*/  
 // подключить файл класса-контроллера* $controllerFile = *ROOT*.'/controllers/'.$controllerName.'.php';  
 if(*file\_exists*($controllerFile)) {  
 include\_once($controllerFile);  
 }  
 *// создать объект, вызвать метод (т.е. action)* $controllerObject = new $className;  
 *//$result = $controllerObject->$actionName($parameters);* $result = *call\_user\_func\_array*(array($controllerObject, $actionName), $parameters);  
 *//$result = $controllerObject->$actionName();* if($result != null) {  
 break;  
 }  
 *//СВОЁ* break;  
 }  
 }

2.5 Файл CabinetController.php

<?php  
  
class CabinetController  
{  
 public function actionView() {  
 $User = new Users();  
 $StatEx = new StatisticsExercise();  
 $News = new News();  
 $Elective = new Elective();  
 $teachers = $User->getAllTeachers();  
 $dataElective = $Elective->getByTitle('departmen');  
 $idElective = $dataElective['id'];  
 $importanceNews = $News->getAll($idElective, 1, 4, false);  
  
 $idUser = $User->checkLogged();  
 if(!$idUser) {  
 *header*("Location: http://www.englishtest.ua/login");  
 }  
 $user = $User->getById($idUser);  
  
 $tests = $StatEx->getByType('test', $idUser);  
 $puzzles = $StatEx->getByType('puzzle', $idUser);  
 $inscribes = $StatEx->getByType('inscribe', $idUser);  
  
 require\_once(*ROOT*. '/views/cabinet/index.php');  
 return true;  
 }  
  
 public function actionAjaxChangeInfo() {  
 $User = new Users();  
  
 $idUser = $\_POST['idUser'];  
 $firstName = $\_POST['firstName'];  
 $middleName = $\_POST['middleName'];  
 $lastName = $\_POST['lastName'];  
 $oldPassword = $\_POST['oldPassword'];  
 $newPassword = $\_POST['newPassword'];  
  
 $arrEr = array();  
 $check = true;  
  
 if($User->checkName($firstName) && $User->checkName($middleName) && $User->checkName($lastName)) {  
 if($User->checkPassword($oldPassword) && $User->checkPassword($newPassword)){  
 $flag = $User->updateParameter('firstName', $firstName, $idUser);  
 if(!$flag) {  
 $arrEr['firstName'] = "Помилка (Ім'я)";  
 $check = false;  
 }  
 $flag = $User->updateParameter('middleName', $middleName, $idUser);  
 if(!$flag) {  
 $arrEr['middleName'] = "Помилка (По-батькові)";  
 $check = false;  
 }  
 $flag = $User->updateParameter('lastName', $lastName, $idUser);  
 if(!$flag) {  
 $arrEr['lastName'] = "Помилка (Призвище)";  
 $check = false;  
 }  
 $flag = $User->changePassword($idUser, $oldPassword, $newPassword);  
 if(!$flag) {  
 $arrEr['password'] = "Помилка (Пароль)";  
 $check = false;  
 }  
 echo json\_encode(array(  
 'check' => $check,  
 'arrEr' => $arrEr,  
 ));  
 }  
 else {  
 $arrEr['password'] = "Занадно короткий пароль";  
 }  
 }  
 else {  
 $arrEr['firstName'] = "Занадно короткі дані П.І.Б";  
 }  
 echo json\_encode(array(  
 'check' => false,  
 'arrEr' => $arrEr,  
 ));  
 }  
}

## 2.6 Файл ElectiveController

<?php  
  
class ElectiveController  
{  
 public function actionView() {  
 $Elective = new Elective();  
 $Teacher = new Users();  
 $DescOfSection = new DescOfSiteSection();  
  
 $aboutElectives = $DescOfSection->getByName('elective');  
 $electiveList = $Elective->getAll();  
 $teacherList = $Teacher->getAllTeachers();  
  
 require\_once(*ROOT*.'/views/elective/view.php');  
 return true;  
 }  
  
 public function actionViewMedia($idNews) {  
 $News = new News();  
 $Image = new Image();  
  
 $title = $News->getById($idNews);  
 $title = $title['title'];  
 $images = $Image->getByIdNews($idNews);  
  
 require\_once(*ROOT*.'/views/elective/media.php');  
 return true;  
 }  
  
 public function actionItemElective($idElective) {  
 $Elective = new Elective();  
 $Teacher = new Users();  
 $News = new News();  
  
 $news = $News->getAll($idElective, 0, false, false);  
  
 $elective = $Elective->getById($idElective);  
 $idAuthor = $elective['idAuthor'];  
 $teacher = $Teacher->getById($idAuthor);  
  
 require\_once(*ROOT*.'/views/elective/itemElective.php');  
 return true;  
 }  
  
 public function actionItemNews($idNews) {  
 $User = new Users();  
 $News = new News();  
 $Comments = new Comments();  
  
 $news = $News->getById($idNews);  
  
 $comments = $Comments->getAllByIdNews($idNews, false, false);  
  
 $idUser = $User->checkLogged();  
  
 $idTeacher = $news['idAuthor'];  
 $teacher = $User->getById($idTeacher);  
  
 require\_once(*ROOT*.'/views/elective/itemNews.php');  
 return true;  
 }  
  
 public function actionAjaxAddComment() {  
 $User = new Users();  
 $Comments = new Comments();  
  
 $idNews = $\_POST['idNews'];  
 $text = $\_POST['text'];  
 $idAuthor = $User->checkLogged();  
  
 $arr = array();  
 $arr['idAuthor'] = $idAuthor;  
 $arr['idNews'] = $idNews;  
 $arr['text'] = $text;  
  
 $res = $Comments->add($arr);  
 if($res) {  
 echo true;  
 }  
 else {  
 echo false;  
 }  
 }  
}

## 2.7 Файл ExerciseController

<?php  
  
class ExerciseController  
{  
 public function actionView() {  
 $DescOfSection = new DescOfSiteSection();  
 $Folder = new Folders();  
 $User = new Users();  
  
 $description = $DescOfSection->getByName('exercise');  
 $description = $description['description'];  
 $folders = $Folder->getAll('test');  
 $idUser = $User->checkLogged();  
  
 require\_once(*ROOT* . '/views/exercise/view.php');  
 return true;  
 }  
  
 public function actionAjaxGetFolder() {  
 $typeFolder = $\_POST['typeExercise'];  
 $Folder = new Folders();  
 $folders = $Folder->getAll($typeFolder);  
 echo json\_encode($folders);  
 }  
  
 public function actionViewTest($idFolder) {  
 $Folder = new Folders();  
 $User = new Users();  
 $Test = new Test();  
 $StatEx = new StatisticsExercise();  
  
 $idUser = $User->checkLogged();  
 if(!$idUser) {  
 *header*("Location: http://www.englishtest.ua/login");  
 }  
  
 $infoFolder = $Folder->getById($idFolder);  
 $defTests = $Test->getAllByIdFolder($idFolder);  
  
 *//output = [i]{'id', 'idFolder', 'text', 'ans1', 'ans2', 'ans3', 'ans4'}* $tests = $Test->formatTests($defTests);  
  
 $answerStr = '';  
  
 */\*Так и нужно, ничего не менять\*/* $len = *count*($defTests);  
 for($i = 0; $i < $len-1; $i++) {  
 $answerStr.=$defTests[$i]['answerRight'].'/';  
 }  
 $answerStr.=$defTests[$len-1]['answerRight'];  
  
 $lastExer = $StatEx->getLastForStat($idUser);  
  
 require\_once(*ROOT* . '/views/exercise/test.php');  
 return true;  
 }  
  
 public function actionAjaxSaveRes() {  
 $User = new Users();  
 $StatEx = new StatisticsExercise();  
  
 $idUser = $User->checkLogged();  
 $idFolder = $\_POST['idFolder'];  
 $mark = $\_POST['mark'];  
 $countQuestion = $\_POST['countQuestion'];  
 $countRight = $\_POST['countRight'];  
  
 $StatEx->add($idFolder, $idUser, $countQuestion, $countRight, $mark);  
 }  
  
 public function actionAjaxShowInfoModal() {  
 $idUser = $\_POST['idUser'];  
 $idFolder = $\_POST['idFolder'];  
  
 $User = new Users();  
 $StatEx = new StatisticsExercise();  
  
 $bestRes = $StatEx->getBestRes($idUser, $idFolder);  
 $user = $User->getById($idUser);  
 echo json\_encode(array(  
 'best' => $bestRes,  
 'user' => $user  
 ));  
 }  
  
 public function actionViewInscribe($idFolder) {  
 $Folder = new Folders();  
 $User = new Users();  
 $Inscribe = new Inscribe();  
 $StatEx = new StatisticsExercise();  
  
 $idUser = $User->checkLogged();  
 if(!$idUser) {  
 *header*("Location: http://www.englishtest.ua/login");  
 }  
  
 $infoFolder = $Folder->getById($idFolder);  
 $defInscribe = $Inscribe->getAllByIdFolder($idFolder);  
  
 $arrStr = array();  
  
 $i = 0;  
 foreach($defInscribe as $item) {  
 $arrStr[$i]['words'] = *explode*(' ', $item['text']);  
 $arrStr[$i]['skipWord'] = $item['skipWord'];  
 $arrStr[$i]['id'] = $item['id'];  
 $arrStr[$i]['idFolder'] = $item['idFolder'];  
 $i++;  
 }  
  
 $lastExer = $StatEx->getLastForStat($idUser);  
  
 require\_once(*ROOT* . '/views/exercise/inscribe.php');  
 return true;  
 }  
  
 public function actionViewPuzzle($idFolder) {  
 $Folder = new Folders();  
 $User = new Users();  
 $Puzzle = new Puzzles();  
 $StatEx = new StatisticsExercise();  
  
 $idUser = $User->checkLogged();  
 if(!$idUser) {  
 *header*("Location: http://www.englishtest.ua/login");  
 }  
  
 $infoFolder = $Folder->getById($idFolder);  
 $defPuzzle = $Puzzle->getAllByIdFolder($idFolder);  
  
 $arrStr = array();  
 $i = 0;  
 foreach($defPuzzle as $item) {  
 $tempText = *explode*(' ', $item['textEnglish']);  
 *shuffle*($tempText);  
 $arrStr[$i]['words'] = $tempText;  
 $arrStr[$i]['textEnglish'] = *explode*(' ', $item['textEnglish']);  
 $arrStr[$i]['textPuzzle'] = $item['textPuzzle'];  
 $arrStr[$i]['textInfo'] = $item['textEnglish'];  
 $arrStr[$i]['id'] = $item['id'];  
 $arrStr[$i]['idFolder'] = $item['idFolder'];  
 $i++;  
 }  
  
 $lastExer = $StatEx->getLastForStat($idUser);  
  
 require\_once(*ROOT* . '/views/exercise/puzzle.php');  
 return true;  
 }  
}

2.8 Файл InscribeController.php

<?php  
  
class InscribeController {  
 public function actionViewInscribe ($idFolder) {  
 $Folder = new Folders();  
 $Inscribe = new Inscribe();  
 $dataFolder = $Folder->getById($idFolder);  
 $dataInscribe = $Inscribe->getAllByIdFolder($idFolder);  
 $transText = array();  
 foreach($dataInscribe as $inscribe) {  
 $transText[] = $Inscribe->transformText($inscribe['text'], $inscribe['skipWord']);  
 }  
 require\_once(*ROOT* . '/views/training/inscribe.php');  
 return true;  
 }  
  
 public function actionAjaxGetDataByFolder() {  
 $Inscribe = new Inscribe();  
 $idFolder = $\_POST['idFolder'];  
 $data = $Inscribe->getAllByIdFolder($idFolder);  
 echo json\_encode($data);  
 }  
}

## 2.9 Файл LibraryController.php

<?php  
  
class LibraryController  
{  
 public function actionView() {  
 $DescOfSiteSection = new DescOfSiteSection();  
 $descOfSection = $DescOfSiteSection->getByName('library');  
  
 $Book = new Book();  
 $books = $Book->getAll(false, false);  
  
 require\_once(*ROOT*.'/views/library/view.php');  
 return true;  
 }  
 public function actionAjaxGetById() {  
 $idBook = $\_POST['idBook'];  
 $Book = new Book();  
 $res = $Book->getById($idBook);  
 echo json\_encode($res);  
 }  
 public function actionAjaxGetBySearch(){  
 $Book = new Book();  
 $res = $Book->getBySearch($\_POST['type'], $\_POST['text']);  
 echo json\_encode($res);  
 }  
 public function actionAjaxAddDownload() {  
 $Book = new Book();  
 return $Book->plusDownloadBook($\_POST['idBook']);  
 }  
}

## 2.10 Файл MainController.php

<?php  
  
class MainController  
{  
 public function actionView()  
 {  
 $DescriptionSection = new DescOfSiteSection();  
 $News = new News();  
 $Elective = new Elective();  
 $User = new Users();  
 $Book = new Book();  
  
 $dataElective = $Elective->getByTitle('departmen');  
 $idElective = $dataElective['id'];  
  
 $importanceNews = $News->getAll($idElective, 1, 4, false);  
  
 $otherNews = $News->getAll($idElective, 0, 6, false);  
  
 $aboutDepartmen = $DescriptionSection->getByName('mainAbout');  
  
 $teachers = $User->getAllTeachers();  
  
 $dataBooks = $Book->getAll(3, false);  
  
 require\_once(*ROOT* . '/views/main.php');  
 return true;  
 }  
}

## 2.11 Файл PuzzleController.php

<?php  
  
class PuzzleController {  
 public function actionViewPuzzle ($idFolder) {  
 $Folder = new Folders();  
 $Puzzle = new Puzzles();  
 $dataFolder = $Folder->getById($idFolder);  
 $dataPuzzle = $Puzzle->getAllByIdFolder($idFolder);  
 $transText = array();  
 foreach($dataPuzzle as $puzzle) {  
 $transText[] = $Puzzle->cutText($puzzle['textPuzzle']);  
 }  
 require\_once(*ROOT* . '/views/training/puzzle.php');  
 return true;  
 }  
  
 public function actionAjaxGetDataByFolder() {  
 $Inscribe = new Inscribe();  
 $idFolder = $\_POST['idFolder'];  
 $data = $Inscribe->getAllByIdFolder($idFolder);  
 echo json\_encode($data);  
 }  
}

## 2.12 Файл TestController.php

<?php  
  
class TestController {  
 public function actionViewTest($idFolder) {  
 $Folder = new Folders();  
 $Test = new Test();  
 $dataFolder = $Folder->getById($idFolder);  
 $dataTests = $Test->getAllByIdFolder($idFolder);  
 require\_once(*ROOT* . '/views/training/test.php');  
 return true;  
 }  
  
 public function actionAjaxCheckTest() {  
 $Test = new Test();  
 $User = new Users();  
 $StatExercise = new StatisticsExercise();  
  
 $idUser = $User->checkLogged();  
 $idFolder = $\_POST['idFolder'];  
 $inputData = json\_decode($\_POST['inputData']);  
 $result = $Test->checkUserAnswer($idFolder, $inputData);  
  
 $arr = array();  
 $arr['idFolder'] = $idFolder;  
 $arr['idUser'] = $idUser;  
 $arr['allItem'] = $result['countAll'];  
 $arr['sucItem'] = $result['countRight'];  
 $arr['mark'] = $result['res'];  
 $StatExercise->add($arr);  
  
 echo json\_encode($result);  
 }  
}

## 2.13 Файл UserController.php

<?php  
  
include\_once *ROOT* . '/models/Users.php';  
  
class UserController {  
  
 public function actionAjaxGetById() {  
 $idTeacher = $\_POST['idTeacher'];  
 $User = new Users();  
 echo json\_encode($User->getById($idTeacher));  
 }  
  
 public function actionRegister() {  
  
 $User = new Users();  
  
 $firstName = '';  
 $middleName = '';  
 $lastName = '';  
 $email = '';  
 $password = '';  
 $passwordR = '';  
  
 $errors = array();  
 $errors['firstName'] = '';  
 $errors['middleName'] = '';  
 $errors['lastName'] = '';  
 $errors['email'] = '';  
 $errors['password'] = '';  
 $errors['passwordR'] = '';  
 $errors['all'] = '';  
  
 if(isset($\_POST['submit'])) {  
 $firstName = $\_POST['firstName'];  
 $middleName = $\_POST['middleName'];  
 $lastName = $\_POST['lastName'];  
 $email = $\_POST['email'];  
 $password = $\_POST['password'];  
 $passwordR = $\_POST['passwordR'];  
  
  
 $ok = true;  
  
 if(!$User->checkName($firstName)) {  
 $errors['firstName'] = "Занадто коротке ім'я або присутні цифри";  
 }  
 if(!$User->checkName($middleName)) {  
 $errors['middleName'] = "Занадто коротке призвище або присутні цифри";  
 }  
 if(!$User->checkName($lastName)) {  
 $errors['lastName'] = "Занадто коротке по-батькові або присутні цифри";  
 }  
 if($User->checkMail($email)) {  
 $errors['email'] = "Користувач із такою поштою вже існує";  
 }  
 if(!$User->checkPassword($password)) {  
 $errors['password'] = "Занадто короткий пароль";  
 }  
 if($password != $passwordR) {  
 $errors['passwordR'] = 'Різні паролі';  
 }  
  
 foreach ($errors as $error) {  
 if($error != '') {  
 $ok = false;  
 }  
 }  
  
 if($ok) {  
 $arr = array();  
 $arr['firstName'] = $firstName;  
 $arr['middleName'] = $middleName;  
 $arr['lastName'] = $lastName;  
 $arr['mail'] = $email;  
 $arr['password'] = $password;  
 $arr['idPic'] = NULL;  
  
 $temp = $User->add($arr);  
 if($temp) {  
 *header*("Location: /");  
 }  
 else {  
 $errors['all'] = "Помилка під час відправки";  
 }  
 }  
 }  
  
 require\_once(*ROOT* . '/views/users/register.php');  
 return true;  
 }  
  
 public function actionLogin() {  
 $urlReq = *getenv*("HTTP\_REFERER");  
  
 if($urlReq[7] == 'w') {  
 */\* обрезаем http://www.englishtest.ua \*/* $urlReq = *substr*($urlReq, 25);  
 }  
 else {  
 */\* обрезаем http://englishtest.ua \*/* $urlReq = *substr*($urlReq, 21);  
 }  
  
 if($urlReq != '/login' && $urlReq != '/login#') {  
 $\_SESSION['urlReq'] = $urlReq;  
 }  
  
 $User = new Users();  
 $email = '';  
 $password = '';  
  
 $errors = array();  
 $errors['email'] = '';  
 $errors['password'] = '';  
 $errors['all'] = '';  
  
 $ok = true;  
  
 if(isset($\_POST['submit'])) {  
 $email = $\_POST['email'];  
 $password = $\_POST['password'];  
  
 if(!$User->checkPassword($password)) {  
 $errors['password'] = 'Слишком короткий пароль';  
 }  
  
 foreach ($errors as $error) {  
 if($error != '') {  
 $ok = false;  
 }  
 }  
  
 $userId = $User->checkLogin($email, $password);  
  
 if($userId == false) {  
 $errors['all'] = 'Неправильная почта или пароль';  
 }  
 else {  
 *//User - Auth* $User->auth($userId);  
  
 if(isset($\_SESSION['urlReq'])) {  
 $goUrl = $\_SESSION['urlReq'];  
 unset($\_SESSION['urlReq']);  
 *header*("Location: ".$goUrl);  
 }  
 else {  
 *header*("Location: /cabinet");  
 }  
 }  
 }  
  
 require\_once(*ROOT* . '/views/users/auth.php');  
 return true;  
 }  
  
 public function actionLogout() {  
 unset($\_SESSION['user']);  
 *header*("Location: /");  
 }  
}

## 2.14 Файл Book.php

<?php  
  
class Book {  
 public function add($array) {  
 if(isset($array)) {  
 $title = $array['title'];  
 $author = $array['author'];  
 $yearBegin = $array['yearBegin'];  
 $description = $array['description'];  
 $url = $array['url'];  
 $db = Db::*getConnection*();  
 $result = $db->query("INSERT INTO `book` (title, author, yearBegin, description, url) VALUES ('$title', '$author', '$yearBegin', '$description', '$url')");  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function getAll($limit, $begin) {  
 $resQuery = false;  
 if($limit == false) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` ORDER BY uploadDate DESC");  
 }  
 else if($begin == false && *intval*($limit) > 0) {  
 $limit = (string) $limit;  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` ORDER BY uploadDate DESC LIMIT 0, $limit");  
 }  
 else if(*intval*($limit) > 0 && *intval*($begin) > 0) {  
 $begin = (string) $begin;  
 $limit = (string) $limit;  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` ORDER BY uploadDate DESC LIMIT $begin, $limit");  
 }  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['title'] = $row['title'];  
 $result[$i]['author'] = $row['author'];  
 $result[$i]['yearBegin'] = $row['yearBegin'];  
 $result[$i]['description'] = $row['description'];  
 $result[$i]['url'] = $row['url'];  
 $result[$i]['countDownload'] = $row['countDownload'];  
 $result[$i]['uploadDate'] = $row['uploadDate'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
  
 public function getById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `book` WHERE id='$id'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function getBySearch($type, $val) {  
 $valLC = *mb\_strtolower*($val);  
 $valFUC = *ucfirst*($val);  
 if(isset($type) && isset($val)) {  
 switch($type) {  
 case 'title':  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` WHERE title LIKE '%$val%' OR title LIKE '%$valFUC%' OR title LIKE '%$valLC%'");  
 break;  
 case 'author':  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` WHERE author LIKE '%$val%'");  
 break;  
 case 'year':  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` WHERE yearBegin LIKE '%$val%'");  
 break;  
 case 'description':  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `book` WHERE description LIKE '%$val%'");  
 break;  
 }  
  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['title'] = $row['title'];  
 $result[$i]['author'] = $row['author'];  
 $result[$i]['yearBegin'] = $row['yearBegin'];  
 $result[$i]['description'] = $row['description'];  
 $result[$i]['url'] = $row['url'];  
 $result[$i]['countDownload'] = $row['countDownload'];  
 $result[$i]['uploadDate'] = $row['uploadDate'];  
 $i++;  
 }  
 return $result;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function deleteById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `book` WHERE id='.$id);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function updateParameter($parameterName, $newValue, $id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 if ($parameterName == 'id') {  
 return false;  
 }  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("UPDATE `book` SET $parameterName='$newValue' WHERE id='$id'");  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function plusDownloadBook($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("UPDATE `book` SET countDownload = countDownload+1 WHERE id='$id'");  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
}

## 2.15 Файл Comments.php

<?php  
  
class Comments  
{  
 public function add($array) {  
 if(isset($array)) {  
 $idAuthor = $array['idAuthor'];  
 $idNews = $array['idNews'];  
 $text = $array['text'];  
  
 $db = Db::*getConnection*();  
 $result = $db->query("INSERT INTO `commentar` (idAuthor, idNews, text) VALUES ('$idAuthor', '$idNews', '$text')");  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result->fetch();  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function getAllByIdNews($idNews ,$limit, $begin) {  
 $resQuery = false;  
 if($limit == false) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT `users`.`firstName`, `users`.`middleName`, `users`.`lastName`, `users`.`urlImage`, `commentar`.`text`, `commentar`.`thisDate` FROM `commentar` INNER JOIN `users` ON `commentar`.`idAuthor` = `users`.`id` WHERE `commentar`.`idNews` = '$idNews' ORDER BY `commentar`.`thisDate` ASC");  
 }  
 else if($begin == false && *intval*($limit) > 0) {  
 $limit = (string) $limit;  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT `users`.`firstName`, `users`.`middleName`, `users`.`lastName`, `users`.`urlImage`, `commentar`.`text`, `commentar`.`thisDate` FROM `commentar` INNER JOIN `users` ON `commentar`.`idAuthor` = `users`.`id` WHERE `commentar`.`idNews` = '$idNews' ORDER BY `commentar`.`thisDate` ASC LIMIT 0, $limit");  
 }  
 else if(*intval*($limit) > 0 && *intval*($begin) > 0) {  
 $begin = (string) $begin;  
 $limit = (string) $limit;  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT `users`.`firstName`, `users`.`middleName`, `users`.`lastName`, `users`.`urlImage`, `commentar`.`text`, `commentar`.`thisDate` FROM `commentar` INNER JOIN `users` ON `commentar`.`idAuthor` = `users`.`id` WHERE `commentar`.`idNews` = '$idNews' ORDER BY `commentar`.`thisDate` ASC LIMIT $begin, $limit");  
 }  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['firstName'] = $row['firstName'];  
 $result[$i]['middleName'] = $row['middleName'];  
 $result[$i]['lastName'] = $row['lastName'];  
 $result[$i]['urlImage'] = $row['urlImage'];  
 $result[$i]['text'] = $row['text'];  
 $result[$i]['thisDate'] = $row['thisDate'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
  
 public function getById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `commentar` WHERE id='$id'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function deleteById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `commentar` WHERE id='.$id);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function updateParameter($parameterName, $newValue, $id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 if ($parameterName == 'id') {  
 return false;  
 }  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("UPDATE `commentar` SET $parameterName='$newValue' WHERE id='$id'");  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

## 2.16 Файл DescOfSiteSection.php

<?php  
  
class DescOfSiteSection {  
 public function add($array) {  
 if(isset($array)) {  
  
 $nameSection = $array['nameSection'];  
 $description = $array['description'];  
  
 if (!$this->checkNameSection($nameSection)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("INSERT INTO `descofsitesection` (nameSection, description) VALUES ('$nameSection', '$description')");  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 if ($result) {  
 return true;  
 } else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
 }  
  
 public function getAll() {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `descofsitesection`");  
  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['nameSection'] = $row['nameSection'];  
 $result[$i]['description'] = $row['description'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
  
 public function getById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `descofsitesection` WHERE id='$id'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function getByName($name) {  
 if(isset($name)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `descofsitesection` WHERE nameSection='$name'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function deleteById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `descofsitesection` WHERE id='.$id);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function deleteByName($name) {  
 if(isset($name)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `descofsitesection` WHERE nameSection='.$name);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function updateParameter($parameterName, $newValue, $id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 if ($parameterName == 'id') {  
 return false;  
 }  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("UPDATE `descofsitesection` SET $parameterName='$newValue' WHERE id='$id'");  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private function checkNameSection($name) {  
 if(isset($name)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM 'descofsitesection' WHERE nameSection='$name'");  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
}

## 2.17 Файл Elective.php

<?php  
  
class Elective {  
  
 public function getByTitle($title) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `elective` WHERE title = '$title'");  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 if($result) {  
 return $result;  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function add($array) {  
 if(isset($array)) {  
 $title = $array['title'];  
 $description = $array['description'];  
 $idAuthor = $array['idAuthor'];  
 $idAuthor = *intval*($idAuthor);  
 if(isset($title) && isset($description) && isset($idAuthor)) {  
  
 echo 'In isset Title Desc'.'<br>';  
  
 if(!$this->getByTitle($title)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("INSERT INTO `elective` (title, description, idAuthor) VALUES ('$title', '$description', '$idAuthor')");  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function getAll() {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `elective` WHERE idAuthor != 1");  
  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['title'] = $row['title'];  
 $result[$i]['description'] = $row['description'];  
 $result[$i]['idAuthor'] = $row['idAuthor'];  
 $result[$i]['createDate'] = $row['createDate'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
  
 public function getById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `elective` WHERE id='$id'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function getByIdAuthor($idAuthor) {  
 $idAuthor = *intval*($idAuthor);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `elective` WHERE idAuthor='$idAuthor'");  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['title'] = $row['title'];  
 $result[$i]['description'] = $row['description'];  
 $result[$i]['idAuthor'] = $row['idAuthor'];  
 $result[$i]['createDate'] = $row['createDate'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
 return false;  
 }  
  
 public function deleteById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `elective` WHERE id='.$id);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function updateParameter($parameterName, $newValue, $id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 if ($parameterName == 'id') {  
 return false;  
 }  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("UPDATE `elective` SET $parameterName='$newValue' WHERE id='$id'");  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
}

## 2.18 Файл Folders.php

<?php  
  
class Folders {  
 public function add($array) {  
 if(isset($array)) {  
 $title = $array['title'];  
 $description = $array['description'];  
 if($array['typeExercise'] == 'puzzle' || $array['typeExercise'] == 'inscribe' || $array['typeExercise'] == 'test') {  
 $typeExercise = $array['typeExercise'];  
 $db = Db::*getConnection*();  
 $result = $db->query("INSERT INTO `folders` (title, description, typeExercise) VALUES ('$title', '$description', '$typeExercise')");  
 if($result) {  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
 else {  
 return false;  
 }  
 }  
  
 public function getAll($type) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("SELECT *\** FROM `folders` WHERE typeExercise='$type'");  
  
 $result = array();  
 if($resQuery) {  
 $resQuery->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $i = 0;  
 while($row = $resQuery->fetch()) {  
 $result[$i]['id'] = $row['id'];  
 $result[$i]['title'] = $row['title'];  
 $result[$i]['description'] = $row['description'];  
 $result[$i]['typeExercise'] = $row['typeExercise'];  
 $i++;  
 }  
 return $result;  
 }  
 return false;  
 }  
  
 public function getById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $result = $db->query("SELECT *\** FROM `folders` WHERE id='$id'");  
 if($result) {  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $result = $result->fetch();  
 return $result;  
 }  
 }  
 return false;  
 }  
  
 public function deleteById($id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 $db = Db::*getConnection*();  
 $resQuery = $db->query('DELETE FROM `folders` WHERE id='.$id);  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 public function updateParameter($parameterName, $newValue, $id) {  
 $id = *intval*($id);  
 if(isset($id)) {  
 if ($parameterName == 'id') {  
 return false;  
 }  
 $db = Db::*getConnection*();  
 $resQuery = $db->query("UPDATE `folders` SET $parameterName='$newValue' WHERE id='$id'");  
 if($resQuery) {  
 return true;  
 }  
 }  
 return false;  
 }  
}