**Базова инфорамция за YII2**

YII2 Работи на принципа на MVC. MVC (model, view, controller).

Model - Моделът това са данните с който да оперира системата и това как те се предават на view-тата.

View – Това са страниците който се визуализират с данните в браузъра. Те са изградени от html,css,js и php в тази система. Като PHP рендира html с данните който получава от модела.

Controller – Контролерите са като гара разпределител. Те казват кой данни да се заредят(model) и към коя страница да се предадът(view). Тоест те ръководят цялата система.

Как точно става магията :

Например в браузъра имаме URL:

www.mysite.com/index.php?r=site%2Fhome

r=site в тази система означава, че ще бъде зареден контролер който е с името SiteController а Fhome казва, че трябва да се изпълни методът actionHome в контролера който ще извика някакъв модел(ако страницата има динамични данни) и най – накрая ще рендира view-to.

R- идва от route – Той служи за да извика контролера който се казва SiteController . Разбира, че щом задаваме r=’нещо‘ трябва да намери в папката за контролерите ‚нещоController‘ и да изпълни метода който му е подаден. Ако няма след това например Fhome, той ще зареди винаги метода actionIndex().

Създаваме база данни в phpmyadmin . След това конфигурираме връзката с базата от common/confg/main-local.php Масива components -> host,dbname,username,password

За да създадем таблицата users ние използваме миграциите като го пускаме с конзолната команда ‘yii migrate’ . Автоматично генерира таблица в базата данни, която ни е необходима за да може да използваме входа и регистрацията на системата, която ни предоставя. Създават се две таблици едната е с миграциите другата е с юзърите.

Създаваме ръчно таблици за договорите и кандидатурите(contract и applications). Защото искаме да използваме системата за генериране на CRUD (create,read,update,delete) операции, която ни предоставя YII2, а това е gii tool. Достъпа до него е като се влезе през админа. Се пише в URL-a index.php?r=gii - r идва от route(раут) така системата разбира, към кой контролер да се обърне.

За да може да ползваме CRUD генератора първо трябва да създадем моделите ( също така познати като entities ). Метода е много лесен . В полето за таблица оказваме името на таблицата в базата данни и на името на класа същото. Това е лесен начин за създаване на модели. Създава всички връзки каквито сме дефинирали в базата ( едно към много, много към много, едно към едно). В този случай сме дефинирали много към много ( един потребител може да кандидатства по – много договори и за един договор могат да кандидатстват много потребители). След като генерираме един модел се създават два класа за него в папката models. Единия съдържа всички полета от базата(обектите). Другия основно служи за филтриране на данните(филтрацията).

След като създадем моделите вече може да ползваме CRUD генератора. Той изисква имената на двата класа, който сме създали (обекта, филтрацията). Като генерира контролер автоматично. Както и views.

Какво точно се е генерирало. Генерират се всички филтри и възможности за създаване на обекти, който се записват в базата(договори, кандидатури). Това обаче не е точно, което целим( не трябва да може да се създават нови кандидатури от администатора) затова влизаме в контролера и изтриваме метода който е за създаване на кандидатури. След това изтриваме файла който се е създал в view-то за create и премахваме бутона който води към него в страницата където се показват всички кандидатури.

Администратора има пълни права(изтрива, редактира,преглежда) кандидатури и (създава,изтрива,преглежда,редактира) договори.

Администраторския панел е защитен по следния начин:

За страницата с договорите е използвана системата за потребители и роли, която ни предоставя YII2, а за страницата с управление на кандидатурите е използван много опростен метод с който се показва, че не е задължително за по - малки проекти да използваме по – сложни технологий за достъп.

Управление на договорите защита ( система за управление на роли )

Yii2 ни предоставя система за управление на роли. Но за да я използваме ние трябва да създадем няколко допълнителни таблици в базата данни.

auth\_item – Тя ни позволява да създаваме нова роля. Тук сме създали роля която е ‘create-contracts’ . Тя позволява на потребителя да има право за създаване на договори.

auth\_assignment – Тази таблица служи за да свържем ролята към даден потребител ( не е задължително 1 потребител да има само една роля, възможно е да има и много повече)

auth\_rule и auth\_item\_child служи за да „разбиваме“ ролята на отдели (примерно администратор 1 може да редактира администратор 2 може да редактира и да изтрива, но и 2 мата са администратори) тук не го използваме това.

Проверяваме дали потребителя има роля така

if(Yii::$app->user->can('create-contracts'))

Някъде използвам и отрицанието тоест, ако потребителя няма тази роля

if(!Yii::$app->user->can('create-contracts'))

И добавяме тези ограничение преди да се зареди страницата. Ако потребителя няма достъп той просто ще бъде пренасочен(redirect) към началната страница.

Как е осъществен другия начин за защита. В таблицата с потребителите се добавя нова колона, която е isAdmin. След това когато потребителя влезе в системата и опита да направи нещо с кандидатурите се изпраща заявка към базата, която проверява той дали е админ. Ако не е достъпа му се спира. За целта е направен private(достъпен само в текущия клас CheckAccess) метод в контролера който се извиква преди всяко деиствие.

private function CheckAccess(){

$connection = new \yii\db\Connection([

'dsn' => 'mysql:host=localhost;dbname=framework\_yii2',

'username' => 'root',

'password' => '',

]);

$user = $connection->createCommand('SELECT \* FROM user WHERE id ='.Yii::$app->user->id)->queryOne();

if($user['isAdmin'] != 1){

die('NO ACCESS');

}

}

Това е по – лесен вариянт за управление на достъп.

**Потребители**

Потребителите могат да се регистрират с формата за вписване като тя отново е от yii2, също така има възтановяване на парола. По подразбиране потребителя, който се регистрира не е админ и няма никаква роля.

За UI(юзър интерфейса) на потребителите отново използваме CRUD генератора който създава страниците с договорите и кандидатурите. Проблемът който трябва да оправим тук е обикновенния потребител да има право само да вижда договорите и да вижда собствените си кандидатури. Като е с генератора потребителя има пълен достъп.

Решение при договорите(потребителя трябва да вижда всички договори, но да не може да създава нови и да не може да ги променя както и изтрива) – Изтриваме методите в контролера за създаване,изтриване, редактиране на договори и променяме view-то.

Решение при кандидатурите(тук е малко по – различно потребителя трябва да може да изтрива, редактира и създава кандидатури, но трябва да може да го прави само със собствените си). За целта променяме метода на класа ApplicationSearch така че, ако потребителя не е админ да може да вижда само неговите кандидатури.

if(Yii::$app->user->identity->isAdmin == 0)

{

$query->andFilterWhere(['=','owner\_id',Yii::$app->user->identity->id]);

}

Тук използваме нещо подобно на метода CheckAccess, но в по съкратен вариянт. Всеки логнат потребител има всички полета от базата и тук сравняваме, ако потребителя не е админ isAdmin ==0 (тоест ако е 1 ще е админ). И ако не е админ заявката към базата се филтрира като се добавя

WHERE owner\_id = ‚ид-то на потребителя‘ с което ние правим филтрацията.

Потребителите не могат да виждат и да кандидатстват за договори който са с изтекъл срок. Автоматизиран процес за промяна на договора от активен към неактивен с използването на CRON задачи( достъпни когато се качи на хостинга)

**„Cron Jobs** (автоматизирани задачи) е автоматизиран процес за стартиране на определени задачи и процеси. Чрез него можете да задавате време за изпълнение на даден процес, както и да настройвате период за повторение на изпълнението на процеса. **“**

**Скрипт проверява кога е крайния срок на обявата и ако той е като текущата дата или по - малък обявата става неактивна.( С cron може да се настройва колко често да се изпълнява този скрипт ).**