Конфиги

1. в файле db.php прописываем

'dsn' => 'mysql:host=127.0.0.1:1403;dbname=название БД,

1. в файле web.php раскомментируем urlManager и прописываем

'cookieValidationKey' => '123',

'baseUrl' => '',

1. создаем в корне файл .htaccess и прописываем в нем

RewriteEngine on

RewriteRule ^(.\*)$ web/$1

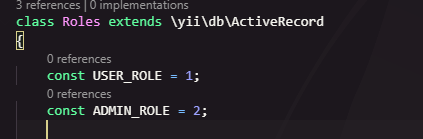
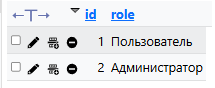
* если класс User называется по-другому в файле web.php меняем

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

Регистрация

1. переименовываем файл User.php в UserOld.php
2. генерируем модель User в gii
3. генерируем CRUD для User
4. если у пользователей есть роли, то:

* генерируем класс для ролей( например Roles)
* прописываем в нем константы в зависимости от того, что у нас в БД



* в контроллере для User находим метод actionCreate и убираем из условия $model->save(). В теле этого условия добавляем выдачу роли $model->role\_id = Roles::USER\_ROLE;. Помещаем return внутрь условия $model->save() и меняем redirect с view на нужную нам страницу( например login).

ДО 

ПОСЛЕ 

* во вьюшке \_form.php убираем поле <?= $form->field($model, 'role\_id')->textInput() ?>
* проверяем что у нас все работает( ссылка типа название\_*проекта/сгенерированная*\_папка\_вьюшек/create)

1. переносим метод actionCreate в SiteController и переименовываем в actionRegister
2. переносим файлы create.php и \_form.php в папку site( теперь регистрация доступна по ссылке название\_*проекта/site*/register)

Валидация

Если ФИО разбито на разные поля, то для каждого поля можно использовать следующую валидацию:

[[name], 'match', 'pattern' => '/^[а-яА-ЯёЁ]+$/u', 'message' => 'что-то'],

* ^ - начало строки
* $ - конец строки
* [а-яА-ЯёЁ] - любая буква кириллицы
* + - любое количество повторений
* /……………/u - юникод

Если ФИО находится в одном поле, то можно использовать следующую валидацию:

[['fio'], 'match', 'pattern' => '/^[а-яА-ЯёЁ]+\s[а-яА-ЯёЁ]+\s[а-яА-ЯёЁ]+$/u', 'message' => 'что-то'],

* \s - пробел

Для валидации почты можно использовать следующий код:  
[['email'], 'email', 'message' => 'Неверный формат почты'],

[['email'], 'unique', 'message' => 'Эта почта уже используется'],

Если нужно сделать проверку на количество символов, то можно использовать следующую валидацию:

[['password'], 'string', 'min' => 6, 'tooShort' => 'Пароль должен содержать как минимиум 6 символов'],

Если пароль нужно вводить дважды, то:

* Объявляем переменную в модели User:

var $password\_confirm = "";

* Делаем поле обязательным:

[['password\_confirm'], 'required', 'message' => 'Это обязательное поле'],

* Делаем проверку на совпадение паролей:

[['password\_confirm'], 'compare', 'compareAttribute' => 'password', 'message' => 'Пароли не совпадают'],

* Добавляем перевод:

public function attributeLabels()

{

return [

'password\_confirm' => 'Подтверждение пароля',

];

}

* Добавляем поле во вьюшке:

<?= $form->field($model, 'password\_confirm')->passwordInput(['maxlength' => true]) ?>

Для создания маски для телефона меняем поле ввода телефона на:

<?= $form->field($model, 'tel')->textInput(['maxlength' => true])->widget(\yii\widgets\MaskedInput::class, [

'mask' => '+7(999)999-99-99',

]) ?>

Для ограничения даты и времени поменять поле ввода начала бронирования на: <?= $form->field($model, 'time\_start')->input('datetime-local', [

'max' => date('Y-m-d\TH:i:s', strtotime("+1 year")),

'min' => (new DateTime())->setTime(0, 0)->modify('+1 day')->format("Y-m-d\TH:i:s"),

]) ?>

Авторизация

1. из UserOld берем implements \yii\web\IdentityInterface и добавляем к нашему классу User
2. копируем все методы из того же класса и вставляем в класс User
3. У методов validateAuthKey getAuthKey findIdentityByAccessToken заменяем тело на return null;

* В ранее сгенерированной модели UserSearch можно найти метод find( $query = Users::find(); ) ctrl+ЛКМ по нему потом х2 ЛКМ. в этом файле нажимаем ctrl+f и ищем where. Первая найденная строка будет $user = User::find()->where(['name' => 'CeBe'])->one(); . Копируем ее.

1. В методе findIdentity заменяем return на return static::find()->where(['id' => $id])->one();
2. В методе findByUsername удаляем все в теле метода и заменяем на return static::find()->where(['email' => $email])->one(); . Также меняем аргумент функции на $email и переименовываем функцию ( public static function findByEmail($email) )
3. В LoginForm.php меняем все username на email
4. Меняем название метода в LoginForm.php: $this->\_user = User::findByEmail($this->email);

* Если класс User назван по-другому нужно не забыть изменить эту строку $this->\_user = User::findByEmail($this->email);

1. Во вьюшке делаем тоже самое
2. В main.php делаем тоже самое

Панель навигации

В main.php заменяем эту панель навигации:

<?php

NavBar::begin([

'brandLabel' => Yii::$app->name,

'brandUrl' => Yii::$app->homeUrl,

'options' => ['class' => 'navbar-expand-md navbar-dark bg-dark fixed-top']

]);

echo Nav::widget([

'options' => ['class' => 'navbar-nav'],

'items' => [

['label' => 'Home', 'url' => ['/site/index']],

['label' => 'About', 'url' => ['/site/about']],

['label' => 'Contact', 'url' => ['/site/contact']],

Yii::$app->user->isGuest

? ['label' => 'Login', 'url' => ['/site/login']]

: '<li class="nav-item">'

. Html::beginForm(['/site/logout'])

. Html::submitButton(

'Logout (' . Yii::$app->user->identity->email . ')',

['class' => 'nav-link btn btn-link logout']

)

. Html::endForm()

. '</li>'

]

]);

NavBar::end();

?>

На эту:

<?php

NavBar::begin([

'brandLabel' => Yii::$app->name,

'brandUrl' => Yii::$app->homeUrl,

'options' => ['class' => 'navbar-expand-md navbar-dark bg-dark fixed-top']

]);

$items = [

['label' => 'Home', 'url' => ['/']],

];

if(Yii::$app->user->isGuest){

$items[] = ['label' => 'Login', 'url' => ['/site/login']];

$items[] = ['label' => 'Register', 'url' => ['/site/register']];

}

else{

тут страницы, созданные по заданию

$items[] = '<li class="nav-item">'

. Html::beginForm(['/site/logout'])

. Html::submitButton(

'Logout (' . Yii::$app->user->identity->login . ')',

['class' => 'nav-link btn btn-link logout']

)

. Html::endForm()

. '</li>';

}

echo Nav::widget([

'options' => ['class' => 'navbar-nav'],

'items' => $items,

]);

NavBar::end();

?>

Смена статуса заявки

Для смены статуса вставляем следующий код в GridView:

[

'attribute' => 'Сменить статус',

'visible' => (Yii::$app->user->identity->role\_id == Role::ADMIN\_ID ? true : false),

'format' => 'raw',

'value' => function($model){

if($model->status\_id == Status::NEW\_STATUS){

$html = Html::beginForm(Url::to(['update', 'id' => $model->id]));

$html .= Html::activeDropDownList($model, 'status\_id',[

'2' => 'Принять',

'3' => 'Отклонить',

]);

$html .= Html::submitButton('Save', ['class' => 'btn btn-success']);

return $html;

$html .= Html::endForm();

}

return "";

}

],

Заявки пользователя(контроллер)

$ID = null;

if ($user->role\_id == Role::USER\_ID){

$ID = $user->id;

}

$dataProvider = $searchModel->search($this->request->queryParams, $ID);

Нормальные названия(индекс)

['attribute' => 'status\_id',

'value' => function ($model) {

return $model->status;

},

],

toString(Модель)

$public function \_\_toString(){

return $this->category;

}

Или(контроллер)

$query= Yii::$app->user->identity->role\_id == 1 ? Report::find(): Report::find()->where(['user\_id'=>Yii::$app->user->identity->id]);