Код сайта:

CompanyesController.php:

<?php  
  
namespace app\controllers;  
  
use Yii;  
use app\models\Companyes;  
use app\models\CompanyesSearch;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;  
use yii\filters\VerbFilter;  
  
*/\*\*  
 \* CompanyesController implements the CRUD actions for Companyes model.  
 \*/*class CompanyesController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all Companyes models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $searchModel = new CompanyesSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 return $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Companyes model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Companyes model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new Companyes();  
  
 if ($model->load(Yii::*$app*->request->post()))  
 {  
 $model->user\_id = $\_SESSION['\_\_id'];  
  
  
  
 if ($model->save()) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 };  
 }  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
  
 }  
  
 */\*\*  
 \* Updates an existing Companyes model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing Companyes model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
  
 return $this->redirect(['index']);  
 }  
  
 */\*\*  
 \* Finds the Companyes model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *Companyes the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = Companyes::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

MessagesController.php:

<?php  
  
namespace app\controllers;  
  
use app\modules\admin\models\Journalizations;  
use Yii;  
use app\models\User;  
use app\models\Messages;  
use yii\data\ActiveDataProvider;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;  
use yii\filters\VerbFilter;  
  
*/\*\*  
 \* MessagesController implements the CRUD actions for Messages model.  
 \*/*class MessagesController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all Messages models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $dataProvider = new ActiveDataProvider([  
 'query' => Messages::*find*(),  
 ]);  
  
  
  
  
  
 return $this->render('index', [  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Messages model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Messages model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new Messages();  
  
 if ($model->load(Yii::*$app*->request->post())){  
 $model->created\_at = gmdate("Y-m-d H:i:s");  
 $model->id\_users\_sender = $\_SESSION['\_\_id'];  
 if ($model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 1,'13')) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
 }  
 }  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing Messages model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 3,'13')) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing Messages model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 2,'13')) {  
  
 return $this->redirect(['index']);  
 }  
 }  
  
 */\*\*  
 \* Finds the Messages model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *Messages the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = Messages::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

ResumesController.php:

<?php  
  
namespace app\controllers;  
  
use app\modules\admin\models\Journalizations;  
use Yii;  
use app\models\Resumes;  
use app\models\ResumesSearch;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;  
use yii\filters\VerbFilter;  
use \yii\db\Expression;  
  
*/\*\*  
 \* ResumesController implements the CRUD actions for Resumes model.  
 \*/*class ResumesController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all Resumes models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $searchModel = new ResumesSearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 return $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Resumes model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Resumes model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new Resumes();  
  
  
  
 if ($model->load(Yii::*$app*->request->post())){  
  
 $model->id\_user = $\_SESSION['\_\_id'];  
  
  
 if ($model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 1,'17')) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
 }  
 }  
  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing Resumes model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 3,'17')){  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing Resumes model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
 $journalizations = new Journalizations();  
 $journalizations->Oparations($\_SESSION['\_\_id'], 2,'17');  
  
 return $this->redirect(['index']);  
 }  
  
 */\*\*  
 \* Finds the Resumes model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *Resumes the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = Resumes::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

SiteController.php:

<?php  
  
namespace app\controllers;  
  
use app\models\User;  
use app\models\Usersi;  
use app\models\Vacancy;  
use app\modules\admin\models\Journalizations;  
use Yii;  
use yii\filters\AccessControl;  
use yii\web\Controller;  
use yii\web\Response;  
use yii\filters\VerbFilter;  
use app\models\LoginForm;  
use app\models\ContactForm;  
use app\models\SignupForm;  
  
class SiteController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'only' => ['logout'],  
 'rules' => [  
 [  
 'actions' => ['logout'],  
 'allow' => true,  
 'roles' => ['@'],  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'logout' => ['post'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function actions()  
 {  
 return [  
 'error' => [  
 'class' => 'yii\web\ErrorAction',  
 ],  
 'captcha' => [  
 'class' => 'yii\captcha\CaptchaAction',  
 'fixedVerifyCode' => *YII\_ENV\_TEST* ? 'testme' : null,  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Displays homepage.  
 \*  
 \** ***@return*** *string  
 \*/* public function actionIndex()  
 {  
 $model = new Vacancy();  
 return $this->render('index', ['model' => $model]);  
 }  
  
 */\*\*  
 \* Login action.  
 \*  
 \** ***@return*** *Response|string  
 \*/* public function actionLogin()  
 {  
 if (!Yii::*$app*->user->isGuest) {  
 return $this->goHome();  
 }  
  
 $model = new LoginForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->login()) {  
 if ($\_SESSION['\_\_id'] <> ''){  
 $user = new User();  
 $\_SESSION['\_\_role\_name'] = $user->getRolename($\_SESSION['\_\_id']);  
 }  
 return $this->goBack();  
 }  
  
 $model->password = '';  
 return $this->render('login', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Logout action.  
 \*  
 \** ***@return*** *Response  
 \*/* public function actionLogout()  
 {  
 Yii::*$app*->user->logout();  
  
 return $this->goHome();  
 }  
  
 */\*\*  
 \* Displays contact page.  
 \*  
 \** ***@return*** *Response|string  
 \*/* public function actionContact()  
 {  
 $model = new ContactForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->contact(Yii::*$app*->params['adminEmail'])) {  
 Yii::*$app*->session->setFlash('contactFormSubmitted');  
  
 return $this->refresh();  
 }  
 return $this->render('contact', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Displays about page.  
 \*  
 \** ***@return*** *string  
 \*/* public function actionCreateUser(){  
  
 $user = new Usersi();  
 $user->username = 'admin';  
 $user->setPassword('admin');  
 $user->e\_mail = 'admin@mail.ru';  
  
 if ($user->save()) {  
 echo 'good';  
 }  
 var\_dump($user->save());  
  
 }  
 public function actionAbout()  
 {  
 return $this->render('about');  
 }  
  
  
  
 public function actionSignup()  
 {  
 $model = new SignupForm();  
  
 if ($model->load(Yii::*$app*->request->post())) {  
 if ($user = $model->signup()) {  
 if (Yii::*$app*->getUser()->login($user)) {  
 $user = new User();  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($user->lastUser()[0], 1,'21')){  
 return $this->goHome();  
 }  
  
  
 }  
 }  
 }  
  
 return $this->render('signup', [  
 'model' => $model,  
 ]);  
 }  
  
 public function actionPersonalAccount(){  
  
 return $this->render('personalaccount');  
 }  
  
}

VacancyController.php:

<?php  
  
namespace app\controllers;  
  
use app\modules\admin\models\Journalizations;  
use Yii;  
use app\models\Vacancy;  
use app\models\VacancySearch;  
use yii\web\Controller;  
use yii\web\NotFoundHttpException;  
use yii\filters\VerbFilter;  
  
*/\*\*  
 \* VacancyController implements the CRUD actions for Vacancy model.  
 \*/*class VacancyController extends Controller  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 return [  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'delete' => ['POST'],  
 ],  
 ],  
 ];  
 }  
  
 */\*\*  
 \* Lists all Vacancy models.  
 \** ***@return*** *mixed  
 \*/* public function actionIndex()  
 {  
 $searchModel = new VacancySearch();  
 $dataProvider = $searchModel->search(Yii::*$app*->request->queryParams);  
  
 return $this->render('index', [  
 'searchModel' => $searchModel,  
 'dataProvider' => $dataProvider,  
 ]);  
 }  
  
 */\*\*  
 \* Displays a single Vacancy model.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionView($id)  
 {  
 return $this->render('view', [  
 'model' => $this->findModel($id),  
 ]);  
 }  
  
 */\*\*  
 \* Creates a new Vacancy model.  
 \* If creation is successful, the browser will be redirected to the 'view' page.  
 \** ***@return*** *mixed  
 \*/* public function actionCreate()  
 {  
 $model = new Vacancy();  
  
 if ($model->load(Yii::*$app*->request->post())){  
 $model->created\_at = gmdate("Y-m-d H:i:s");  
 if ($model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 1,'22')){  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
  
  
 }  
 }  
  
  
 return $this->render('create', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Updates an existing Vacancy model.  
 \* If update is successful, the browser will be redirected to the 'view' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionUpdate($id)  
 {  
 $model = $this->findModel($id);  
  
 if ($model->load(Yii::*$app*->request->post()) && $model->save()) {  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 3, '22')) {  
 return $this->redirect(['view', 'id' => $model->id]);  
 }  
 }  
  
 return $this->render('update', [  
 'model' => $model,  
 ]);  
 }  
  
 */\*\*  
 \* Deletes an existing Vacancy model.  
 \* If deletion is successful, the browser will be redirected to the 'index' page.  
 \** ***@param*** *integer $id  
 \** ***@return*** *mixed  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* public function actionDelete($id)  
 {  
 $this->findModel($id)->delete();  
 $journalizations = new Journalizations();  
 if ($journalizations->Oparations($\_SESSION['\_\_id'], 2,'22')) {  
  
 return $this->redirect(['index']);  
 }  
 }  
  
 */\*\*  
 \* Finds the Vacancy model based on its primary key value.  
 \* If the model is not found, a 404 HTTP exception will be thrown.  
 \** ***@param*** *integer $id  
 \** ***@return*** *Vacancy the loaded model  
 \** ***@throws*** *NotFoundHttpException if the model cannot be found  
 \*/* protected function findModel($id)  
 {  
 if (($model = Vacancy::*findOne*($id)) !== null) {  
 return $model;  
 }  
  
 throw new NotFoundHttpException('The requested page does not exist.');  
 }  
}

Companyes.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use app\models\Users;  
use app\models\Vacancy;  
  
*/\*\*  
 \* This is the model class for table "companyes".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *int|null $user\_id  
 \** ***@property*** *string $name  
 \** ***@property*** *string|null $phone  
 \** ***@property*** *string|null $e\_mail  
 \** ***@property*** *string|null $created\_at  
 \*  
 \** ***@property*** *User $user  
 \** ***@property*** *Vacancy[] $vacancies  
 \*/*class Companyes extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'companyes';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['user\_id'], 'integer'],  
 [['name'], 'required', 'message' => 'Пожалуйста заполните нименование компании!'],  
 [['phone'], 'required', 'message' => 'Пожалуйста заполните телефон!'],  
 [['e\_mail'], 'required', 'message' => 'Пожалуйста заполните e-mail!'],  
 ['e\_mail', 'email'],  
 [['created\_at'], 'safe'],  
 [['name', 'e\_mail'], 'string', 'max' => 255],  
 [['phone'], 'string', 'max' => 20],  
 [['user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['user\_id' => 'id']],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'user\_id' => 'User ID',  
 'name' => 'Name',  
 'phone' => 'Phone',  
 'e\_mail' => 'E Mail',  
 'created\_at' => 'Created At',  
 ];  
 }  
  
 */\*\*  
 \* Gets query for [[User]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getUser()  
 {  
 return $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
 }  
  
 */\*\*  
 \* Gets query for [[Vacancies]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getVacancies()  
 {  
 return $this->hasMany(Vacancy::*className*(), ['id\_companyes' => 'id']);  
 }  
}

CompanyesSearch.php:

<?php  
  
namespace app\models;  
  
use yii\base\Model;  
use yii\data\ActiveDataProvider;  
use app\models\Companyes;  
  
*/\*\*  
 \* CompanyesSearch represents the model behind the search form of `app\models\Companyes`.  
 \*/*class CompanyesSearch extends Companyes  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id', 'user\_id'], 'integer'],  
 [['name', 'phone', 'e\_mail', 'created\_at'], 'safe'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function scenarios()  
 {  
 // bypass scenarios() implementation in the parent class  
 return Model::*scenarios*();  
 }  
  
 */\*\*  
 \* Creates data provider instance with search query applied  
 \*  
 \** ***@param*** *array $params  
 \*  
 \** ***@return*** *ActiveDataProvider  
 \*/* public function search($params)  
 {  
 $query = Companyes::*find*();  
  
 // add conditions that should always apply here  
  
 $dataProvider = new ActiveDataProvider([  
 'query' => $query,  
 ]);  
  
 $this->load($params);  
  
 if (!$this->validate()) {  
 // uncomment the following line if you do not want to return any records when validation fails  
 // $query->where('0=1');  
 return $dataProvider;  
 }  
  
 // grid filtering conditions  
 $query->andFilterWhere([  
 'id' => $this->id,  
 'user\_id' => $this->user\_id,  
 'created\_at' => $this->created\_at,  
 ]);  
  
 $query->andFilterWhere(['like', 'name', $this->name])  
 ->andFilterWhere(['like', 'phone', $this->phone])  
 ->andFilterWhere(['like', 'e\_mail', $this->e\_mail]);  
  
 return $dataProvider;  
 }  
}

ContactForm.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\base\Model;  
  
*/\*\*  
 \* ContactForm is the model behind the contact form.  
 \*/*class ContactForm extends Model  
{  
 public $name;  
 public $email;  
 public $subject;  
 public $body;  
 public $verifyCode;  
  
  
 */\*\*  
 \** ***@return*** *array the validation rules.  
 \*/* public function rules()  
 {  
 return [  
 // name, email, subject and body are required  
 [['name', 'email', 'subject', 'body'], 'required'],  
 // email has to be a valid email address  
 ['email', 'email'],  
 // verifyCode needs to be entered correctly  
 ['verifyCode', 'captcha'],  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *array customized attribute labels  
 \*/* public function attributeLabels()  
 {  
 return [  
 'verifyCode' => 'Verification Code',  
 ];  
 }  
  
 */\*\*  
 \* Sends an email to the specified email address using the information collected by this model.  
 \** ***@param*** *string $email the target email address  
 \** ***@return*** *bool whether the model passes validation  
 \*/* public function contact($email)  
 {  
 if ($this->validate()) {  
 Yii::*$app*->mailer->compose()  
 ->setTo($email)  
 ->setFrom([Yii::*$app*->params['senderEmail'] => Yii::*$app*->params['senderName']])  
 ->setReplyTo([$this->email => $this->name])  
 ->setSubject($this->subject)  
 ->setTextBody($this->body)  
 ->send();  
  
 return true;  
 }  
 return false;  
 }  
}

LoginForm.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\base\Model;  
use app\models\User;  
  
*/\*\*  
 \* LoginForm is the model behind the login form.  
 \*  
 \** ***@property*** *User|null $user This property is read-only.  
 \*  
 \*/*class LoginForm extends Model  
{  
 public $username;  
 public $password;  
 public $rememberMe = true;  
  
 private $\_user = false;  
  
  
 */\*\*  
 \** ***@return*** *array the validation rules.  
 \*/* public function rules()  
 {  
 return [  
 // username and password are both required  
 [['username', 'password'], 'required'],  
 // rememberMe must be a boolean value  
 ['rememberMe', 'boolean'],  
 // password is validated by validatePassword()  
 ['password', 'validatePassword'],  
 ];  
 }  
  
 */\*\*  
 \* Validates the password.  
 \* This method serves as the inline validation for password.  
 \*  
 \** ***@param*** *string $attribute the attribute currently being validated  
 \** ***@param*** *array $params the additional name-value pairs given in the rule  
 \*/* public function validatePassword($attribute, $params)  
 {  
 if (!$this->hasErrors()) {  
 $user = $this->getUser();  
  
 if (!$user || !$user->validatePassword($this->password)) {  
 $this->addError($attribute, 'Неверное имя пользователя или пароль');  
 }  
 }  
 }  
  
 */\*\*  
 \* Logs in a user using the provided username and password.  
 \** ***@return*** *bool whether the user is logged in successfully  
 \*/* public function login()  
 {  
 if ($this->validate()) {  
 return Yii::*$app*->user->login($this->getUser(), $this->rememberMe ? 3600\*24\*30 : 0);  
 }  
 return false;  
 }  
  
 */\*\*  
 \* Finds user by [[username]]  
 \*  
 \** ***@return*** *User|null  
 \*/* public function getUser()  
 {  
 if ($this->\_user === false) {  
 $this->\_user = User::*findByUsername*($this->username);  
 }  
  
 return $this->\_user;  
 }  
}

Messages.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\grid\Column;  
use app\models\User;  
  
*/\*\*  
 \* This is the model class for table "messages".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *int|null $id\_users\_sender  
 \** ***@property*** *int|null $id\_users\_recipient  
 \** ***@property*** *string|null $text  
 \** ***@property*** *string|null $created\_at  
 \*/*class Messages extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'messages';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id\_users\_sender', 'id\_users\_recipient'], 'integer'],  
 [['text'], 'string'],  
 [['created\_at'], 'safe'],  
 [['id\_users\_sender'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['id\_users\_sender' => 'id']],  
 [['id\_users\_recipient'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['id\_users\_recipient' => 'id']],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'id\_users\_sender' => 'Id Users Sender',  
 'id\_users\_recipient' => 'Id Users Recipient',  
 'text' => 'Text',  
 'created\_at' => 'Created At',  
 ];  
 }  
 public function getUsersSender()  
 {  
 return $this->hasOne(User::*className*(), ['id' => 'id\_users\_sender']);  
 }  
  
 */\*\*  
 \* Gets query for [[UsersRecipient]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getUsersRecipient()  
 {  
 return $this->hasOne(User::*className*(), ['id' => 'id\_users\_recipient']);  
 }  
 public function getMessages($user\_id){  
 $sql = 'Select messages.id\_users\_sender, messages.id\_users\_recipient, messages.text, messages.created\_at FROM messages JOIN users ON users.Id = messages.id\_users\_sender WHere messages.id\_users\_sender=:user\_id or messages.id\_users\_recipient=:user\_id ;';  
 return static::*findBySql*($sql, [':user\_id' => $user\_id])->all();  
  
 }  
 public function getUsers(){  
 return User::*find*()->select('username, id')->all();  
 }  
  
  
}

ResumesSearch.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use app\modules\admin\models\Cities;  
use app\modules\admin\models\Educations;  
use app\modules\admin\models\Status;  
use app\modules\admin\models\Citizenship;  
use yii\behaviors\TimestampBehavior;  
  
*/\*\*  
 \* This is the model class for table "resumes".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *int $id\_user  
 \** ***@property*** *string|null $family  
 \** ***@property*** *string|null $name  
 \** ***@property*** *string|null $patronomic  
 \** ***@property*** *int|null $id\_position  
 \** ***@property*** *int|null $salary  
 \** ***@property*** *string|null $phone  
 \** ***@property*** *string|null $e\_mail  
 \** ***@property*** *int|null $id\_citi  
 \** ***@property*** *int|null $id\_citizenship  
 \** ***@property*** *string|null $birthday  
 \** ***@property*** *int|null $id\_education  
 \** ***@property*** *string|null $experience  
 \** ***@property*** *string|null $education  
 \** ***@property*** *string|null $personal\_qualities  
 \** ***@property*** *int|null $id\_status  
 \** ***@property*** *int|null $created\_at  
 \** ***@property*** *int|null $updated\_at  
 \*  
 \** ***@property*** *Cities $citi  
 \** ***@property*** *Educations $education0  
 \** ***@property*** *Status $status  
 \** ***@property*** *Citizenship $citizenship  
 \** ***@property*** *Users $user  
 \*/*class Resumes extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'resumes';  
 }  
  
  
 public function behaviors()  
 {  
 return [  
 TimestampBehavior::*className*(),  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id\_user'], 'required'],  
 [['id\_user', 'id\_position', 'salary', 'id\_citi', 'id\_citizenship', 'id\_education', 'id\_status'], 'integer'],  
 [['birthday'], 'safe'],  
 [['experience', 'education', 'personal\_qualities'], 'string'],  
 [['family', 'name', 'patronomic', 'phone', 'e\_mail'], 'string', 'max' => 255],  
 [['id\_citi'], 'exist', 'skipOnError' => true, 'targetClass' => Cities::*className*(), 'targetAttribute' => ['id\_citi' => 'id']],  
 [['id\_education'], 'exist', 'skipOnError' => true, 'targetClass' => Educations::*className*(), 'targetAttribute' => ['id\_education' => 'id']],  
 [['id\_status'], 'exist', 'skipOnError' => true, 'targetClass' => Status::*className*(), 'targetAttribute' => ['id\_status' => 'id']],  
 [['id\_citizenship'], 'exist', 'skipOnError' => true, 'targetClass' => Citizenship::*className*(), 'targetAttribute' => ['id\_citizenship' => 'id']],  
 [['id\_user'], 'exist', 'skipOnError' => true, 'targetClass' => Users::*className*(), 'targetAttribute' => ['id\_user' => 'id']],  
 [['created\_at', 'updated\_at'], 'safe'],  
 ];  
 }  
  
  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'id\_user' => 'Id User',  
 'family' => 'Family',  
 'name' => 'Name',  
 'patronomic' => 'Patronomic',  
 'id\_position' => 'Id Position',  
 'salary' => 'Salary',  
 'phone' => 'Phone',  
 'e\_mail' => 'E Mail',  
 'id\_citi' => 'Id Citi',  
 'id\_citizenship' => 'Id Citizenship',  
 'birthday' => 'Birthday',  
 'id\_education' => 'Id Education',  
 'experience' => 'Experience',  
 'education' => 'Education',  
 'personal\_qualities' => 'Personal Qualities',  
 'id\_status' => 'Id Status',  
 'created\_at' => 'Created At',  
 'updated\_at' => 'Updated At',  
 ];  
 }  
  
 */\*\*  
 \* Gets query for [[Citi]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getCiti()  
 {  
 return $this->hasOne(Cities::*className*(), ['id' => 'id\_citi']);  
 }  
  
 */\*\*  
 \* Gets query for [[Education0]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getEducation0()  
 {  
 return $this->hasOne(Educations::*className*(), ['id' => 'id\_education']);  
 }  
  
 */\*\*  
 \* Gets query for [[Status]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getStatus()  
 {  
 return $this->hasOne(Status::*className*(), ['id' => 'id\_status']);  
 }  
  
 */\*\*  
 \* Gets query for [[Citizenship]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getCitizenship()  
 {  
 return $this->hasOne(Citizenship::*className*(), ['id' => 'id\_citizenship']);  
 }  
  
 */\*\*  
 \* Gets query for [[User]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getUser()  
 {  
 return $this->hasOne(Users::*className*(), ['id' => 'id\_user']);  
 }  
 public function getCountResume($id\_status){  
 $sql = 'SELECT *count*(id) from resumes where id\_status=:id\_status;';  
 return static::*findBySql*($sql,[':id\_status' => $id\_status] )->column();  
 }  
}

SignupForm.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\base\Model;  
use app\models\User;  
  
*/\*\*  
 \* Signup form  
 \*/*class SignupForm extends Model  
{  
  
 public $username;  
 public $email;  
 public $password;  
 public $password\_repeat;  
 public $rolename;  
  
 */\*\*  
 \** ***@inheritdoc*** *\*/* public function rules()  
 {  
 return [  
 ['username', 'trim'],  
 ['username', 'required','message' => 'Поле с логином не заполнено'],  
 ['username', 'unique', 'targetClass' => '\app\models\User', 'message' => 'Такой логин уже существвует'],  
 ['username', 'string', 'min' => 2, 'max' => 255],  
 ['email', 'trim'],  
 ['email', 'required','message' => 'Поле с e-mail не заполнено'],  
 ['email', 'email'],  
 ['email', 'string', 'max' => 255],  
 ['email', 'unique', 'targetClass' => '\app\models\User', 'message' => 'Такой e-mail уже существует'],  
 [['password','password\_repeat'], 'required','message' => 'Поле с паролем не заполнено'],  
 [['password','password\_repeat'], 'string'],  
 ['password\_repeat', 'compare', 'compareAttribute' => 'password', 'operator' => '==','message' => 'Пароли не совпадают!'],  
 [['rolename'], 'required', 'message' => 'Роль не выбрана'],  
 ['rolename', 'default', 'value' => 'vacant'],  
  
 ];  
 }  
  
 */\*\*  
 \* Signs user up.  
 \*  
 \** ***@return*** *User|null the saved model or null if saving fails  
 \*/* public function signup()  
 {  
  
 if (!$this->validate()) {  
 return null;  
 }  
  
 $user = new User();  
 $user->username = $this->username;  
 $user->email = $this->email;  
 $user->setPassword($this->password);  
 $user->generateAuthKey();  
 $user->role\_name = $this->rolename;  
 return $user->save() ? $user : null;  
 }  
  
}

User.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\base\NotSupportedException;  
use yii\web\IdentityInterface;  
use yii\behaviors\TimestampBehavior;  
use yii\db\Query;  
  
*/\*\*  
 \* This is the model class for table "users".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *string $role\_name  
 \** ***@property*** *string $username  
 \** ***@property*** *string|null $auth\_key  
 \** ***@property*** *string $password\_hash  
 \** ***@property*** *string $email  
 \** ***@property*** *int $status  
 \** ***@property*** *int|null $created\_at  
 \** ***@property*** *int|null $updated\_at  
 \*/*class User extends \yii\db\ActiveRecord implements IdentityInterface  
{  
 const *STATUS\_DELETED* = 0;  
 const *STATUS\_ACTIVE* = 10;  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'users';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function behaviors()  
 {  
 $this->created\_at = time();  
 $this->updated\_at = time();  
 return [  
 TimestampBehavior::*className*(),  
 ];  
 }  
  
 public function rules()  
 {  
 return [  
 ['status', 'default', 'value' => self::*STATUS\_ACTIVE*],  
 ['status', 'in', 'range' => [self::*STATUS\_ACTIVE*, self::*STATUS\_DELETED*]],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'role\_name' => 'Role Name',  
 'username' => 'Username',  
 'auth\_key' => 'Auth Key',  
 'password\_hash' => 'Password Hash',  
 'email' => 'Email',  
 'status' => 'Status',  
 'created\_at' => 'Created At',  
 'updated\_at' => 'Updated At',  
 ];  
 }  
 public static function findIdentity($id)  
 {  
 return static::*findOne*(['id' => $id, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 public static function findIdentityByAccessToken($token, $type = null)  
 {  
 throw new NotSupportedException('"findIdentityByAccessToken" is not implemented.');  
 }  
 public static function findByUsername($username)  
 {  
 return static::*findOne*(['username' => $username, 'status' => self::*STATUS\_ACTIVE*]);  
 }  
  
 public function getId()  
 {  
 return $this->getPrimaryKey();  
 }  
  
 public function getAuthKey()  
 {  
 return $this->auth\_key;  
 }  
  
 public function validateAuthKey($authKey)  
 {  
 return $this->getAuthKey() === $authKey;  
 }  
  
 public function validatePassword($password)  
 {  
 return Yii::*$app*->security->validatePassword($password, $this->password\_hash);  
 }  
  
 public function setPassword($password)  
 {  
 $this->password\_hash = Yii::*$app*->security->generatePasswordHash($password);  
 }  
  
 public function generateAuthKey()  
 {  
 $this->auth\_key = Yii::*$app*->security->generateRandomString();  
 }  
 public function getRoleName($user\_id){  
 return static::*find*()->select(['role\_name'])  
 ->where(['id' => $\_SESSION['\_\_id']])  
 ->column();  
 }  
 public function getUsername($user\_name){  
  
 $query = new Query();  
 return $query->select('username')  
 ->from('users')  
 ->where('id=:user\_id',['user\_id' => $user\_name])  
 ->column();  
 }  
  
 public function lastUser(){  
 $sql = 'SELECT id FROM users ORDER BY id DESC LIMIT 0,1;';  
 return User::*findBySql*($sql)->column();  
 }  
}

Vacancy.php:

<?php  
  
namespace app\models;  
  
use Yii;  
use app\modules\admin\models\Cities;  
use app\modules\admin\models\Educations;  
use app\modules\admin\models\Position;  
use app\modules\admin\models\Schedule;  
use app\modules\admin\models\Status;  
  
*/\*\*  
 \* This is the model class for table "vacancy".  
 \*  
 \** ***@property*** *int $id  
 \** ***@property*** *int $id\_companyes  
 \** ***@property*** *int $id\_status  
 \** ***@property*** *int $id\_cities  
 \** ***@property*** *int|null $id\_positions  
 \** ***@property*** *int|null $id\_educations  
 \** ***@property*** *int|null $salary  
 \** ***@property*** *int|null $id\_schedules  
 \** ***@property*** *string|null $text  
 \** ***@property*** *string|null $phone  
 \** ***@property*** *string|null $e\_mail  
 \** ***@property*** *string|null $created\_at  
 \*  
 \** ***@property*** *Cities $cities  
 \** ***@property*** *Companyes $companyes  
 \** ***@property*** *Educations $educations  
 \** ***@property*** *Position $positions  
 \** ***@property*** *Schedule $schedules  
 \** ***@property*** *Status $status  
 \*/*class Vacancy extends \yii\db\ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'vacancy';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id\_companyes', 'id\_status', 'id\_cities'], 'required'],  
 [['id\_companyes', 'id\_status', 'id\_cities', 'id\_positions', 'id\_educations', 'salary', 'id\_schedules'], 'integer'],  
 [['text'], 'string'],  
 [['created\_at'], 'safe'],  
 [['phone', 'e\_mail'], 'string', 'max' => 255],  
 [['id\_cities'], 'exist', 'skipOnError' => true, 'targetClass' => Cities::*className*(), 'targetAttribute' => ['id\_cities' => 'id']],  
 [['id\_companyes'], 'exist', 'skipOnError' => true, 'targetClass' => Companyes::*className*(), 'targetAttribute' => ['id\_companyes' => 'id']],  
 [['id\_educations'], 'exist', 'skipOnError' => true, 'targetClass' => Educations::*className*(), 'targetAttribute' => ['id\_educations' => 'id']],  
 [['id\_positions'], 'exist', 'skipOnError' => true, 'targetClass' => Position::*className*(), 'targetAttribute' => ['id\_positions' => 'id']],  
 [['id\_schedules'], 'exist', 'skipOnError' => true, 'targetClass' => Schedule::*className*(), 'targetAttribute' => ['id\_schedules' => 'id']],  
 [['id\_status'], 'exist', 'skipOnError' => true, 'targetClass' => Status::*className*(), 'targetAttribute' => ['id\_status' => 'id']],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'id\_companyes' => 'Id Companyes',  
 'id\_status' => 'Id Status',  
 'id\_cities' => 'Id Cities',  
 'id\_positions' => 'Id Positions',  
 'id\_educations' => 'Id Educations',  
 'salary' => 'Salary',  
 'id\_schedules' => 'Id Schedules',  
 'text' => 'Text',  
 'phone' => 'Phone',  
 'e\_mail' => 'E Mail',  
 'created\_at' => 'Created At',  
 ];  
 }  
  
 */\*\*  
 \* Gets query for [[Cities]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getCities()  
 {  
 return $this->hasOne(Cities::*className*(), ['id' => 'id\_cities']);  
 }  
  
 */\*\*  
 \* Gets query for [[Companyes]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getCompanyes()  
 {  
 return $this->hasOne(Companyes::*className*(), ['id' => 'id\_companyes']);  
 }  
  
 */\*\*  
 \* Gets query for [[Educations]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getEducations()  
 {  
 return $this->hasOne(Educations::*className*(), ['id' => 'id\_educations']);  
 }  
  
 */\*\*  
 \* Gets query for [[Positions]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getPositions()  
 {  
 return $this->hasOne(Position::*className*(), ['id' => 'id\_positions']);  
 }  
  
 */\*\*  
 \* Gets query for [[Schedules]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getSchedules()  
 {  
 return $this->hasOne(Schedule::*className*(), ['id' => 'id\_schedules']);  
 }  
  
 */\*\*  
 \* Gets query for [[Status]].  
 \*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getStatus()  
 {  
 return $this->hasOne(Status::*className*(), ['id' => 'id\_status']);  
 }  
 public function getVacancyes(){  
 return static::*find*()->select()  
 ->from(*vacancy*)  
 ->all();  
 }  
  
 public function activeVacancy($id\_status){  
 $sql = 'SELECT *count*(id) from vacancy where id\_status=:id\_status;';  
 return static::*findBySql*($sql,[':id\_status' => $id\_status] )->column();  
 }  
}

VacancySearch.php:

<?php  
  
namespace app\models;  
  
use yii\base\Model;  
use yii\data\ActiveDataProvider;  
use app\models\Vacancy;  
  
*/\*\*  
 \* VacancySearch represents the model behind the search form of `app\models\Vacancy`.  
 \*/*class VacancySearch extends Vacancy  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['id', 'id\_companyes', 'id\_status', 'id\_cities', 'id\_positions', 'id\_educations', 'salary', 'id\_schedules'], 'integer'],  
 [['text', 'phone', 'e\_mail', 'created\_at'], 'safe'],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function scenarios()  
 {  
 // bypass scenarios() implementation in the parent class  
 return Model::*scenarios*();  
 }  
  
 */\*\*  
 \* Creates data provider instance with search query applied  
 \*  
 \** ***@param*** *array $params  
 \*  
 \** ***@return*** *ActiveDataProvider  
 \*/* public function search($params)  
 {  
 $query = Vacancy::*find*();  
  
 // add conditions that should always apply here  
  
 $dataProvider = new ActiveDataProvider([  
 'query' => $query,  
 ]);  
  
 $this->load($params);  
  
 if (!$this->validate()) {  
 // uncomment the following line if you do not want to return any records when validation fails  
 // $query->where('0=1');  
 return $dataProvider;  
 }  
  
 // grid filtering conditions  
 $query->andFilterWhere([  
 'id' => $this->id,  
 'id\_companyes' => $this->id\_companyes,  
 'id\_status' => $this->id\_status,  
 'id\_cities' => $this->id\_cities,  
 'id\_positions' => $this->id\_positions,  
 'id\_educations' => $this->id\_educations,  
 'salary' => $this->salary,  
 'id\_schedules' => $this->id\_schedules,  
 'created\_at' => $this->created\_at,  
 ]);  
  
 $query->andFilterWhere(['like', 'text', $this->text])  
 ->andFilterWhere(['like', 'phone', $this->phone])  
 ->andFilterWhere(['like', 'e\_mail', $this->e\_mail]);  
  
 return $dataProvider;  
 }  
}